Club de Investigación Tecnológica

Arquitectura Orientada a Servicios

Ing. Jorge Ramírez J., M.Sc. (cand.)

Noviembre 2009

Club de Investigación Tecnológica

Informes publicados

	Informe	Autor	Fecha
1.	Redes de Computadores	Dr. Roberto Sasso	Agosto 1988
2.	Sistemas Expertos	Dr. Claudio Gutiérrez	Enero 1989
3.	Planificación de Sistemas	Dr. René-Pierre Bondu	Abril 1989
4.	Proyectos de Sistemas	Ing. Ignacio Trejos	Setiembre 1989
5.	Bases de Datos	Dr. Carlos González	Diciembre 1989
6.	Escapando de los Sistemas del Ayer	Lic. Pablo Rojas, M.Sc.	Marzo 1990
7.	Aplicaciones Creativas	Dr. Roberto Sasso	Mayo 1990
8.	Calidad de Sistemas	Dr. Ulises Agüero	Octubre 1990
9.	Personal y Organización de Sistemas	KPMG Consultores	Marzo 1991
10.	Sistemas Abiertos	Ing. José Rubinstein, MBA	Octubre 1991
11.	Análisis de la Industria de la TI.	Lic. Roberto Venegas, MBA	Enero 1992
12.	Nuevas Tecnologías de Información	Dr. Roberto Sasso (Editor)	Marzo 1992
13.	Ambientes de Proveedores Múltiples	Lic. Alexis Rodríguez U.	Julio 1992
14.	Planificación y Recuperación de Desastres	Sr. Gerardo Ortuño	Agosto 1992
15.	Diseño de Redes Novell	Ing. David Baruch	Agosto 1993
16.	Minis Vs LANs	Ing. Marvin Campos	Octubre 1993
17.	Intercambio Electrónico de Datos (EDI)	KPMG Consultores	Enero 1995
18.	Sistemas Abiertos de Software	Ing. José Ardón	Abril 1995
19.	Outsourcing de Tecnología de Información	Roxana Murillo, M.Sc.	Julio 1996
20.	Redes Empresariales de Banda Ancha	Ing. Aníbal Mayorga, M.Sc.	Febrero 1997
21.	Comercio Electrónico	Dr. Roberto Sasso Rojas	Abril 1997
22.	Estudio de Opinión Informática	Dr. Freddy Abarca	Julio 1997
23.	Desarrollo de Sistemas Cliente/Servidor	Lic. Édgar Hernández Ing. Luis Martínez	Diciembre 1997
24.	Enfrentando el año 2000. Guía Práctica	Ing. Carlos Gallegos, M.Sc. Dr. Roberto Sasso Ing. Ignacio Trejos, M.Sc.	Mayo 1998
25.	Depósitos de Datos	Beatriz Jiménez, M.Sc. Rafael Avalos, M.Sc.	Noviembre 1998
26	El modelo de objetos: Análisis y Diseño	Ing. Ignacio Trejos, M.Sc. Ing. Antonio Luna	Setiembre 1999
27.	Silicon Valley, 1999	Ing. Mauricio Monge Dr. Roberto Sasso Ing. Ignacio Trejos, M.Sc.	Enero 2000
28.	Calidad de los datos: Un enfoque conceptual	Ing. Lilia Muñoz, M.Sc.	Febrero 2000
29.	El modelo de objetos: Lenguaje de modelaje Unificado (UML)	Ing. Antonio Luna Ing. Ignacio Trejos, M.Sc.	Marzo 2000
30.	Medición de calidad de datos: Un enfoque práctico	Ing. Franco Quirós	Marzo 2000
31.	Seguridad de la información en la era de los negocios digitales	Lic. Édgar Hernández Lic. Marco V. Gámez	Julio 2001
32.	Transformación de aplicaciones legacy	Ing. Declan Good	Agosto 2002
32.	Legacy transformation	Ing. Declan Good	August 2002
33.	Calidad en la especificación de requerimientos	Ing. Javier Rivas	Febrero 2003
34.	Inteligencia de negocios	Lic. José Mayorga	Setiembre 2004
35.	Sistemas colaborativos	Ing. Xinia Robles Lic. Lizette Ramírez, M.Sc.	Octubre 2004
36.	XML: Tecnología y aplicaciones	Dr. José Enrique Araya Ing. Emilia Zeledón	Enero 2005
37.	Procesos de software	Ing. Priscilla Garbanzo, MIS	Setiembre 2005
38.	Patrones de software	Lic. Alan Calderón, M.Sc.	Agosto 2006
39.	Administración del Riesgo en Proyectos Informáticos	Ing. Carlos E. Vargas, CSQE	Noviembre 2007
40.	Arquitectura empresarial	Ing. Gerardo Porras Cedeño, M.Sc.	Setiembre 2008
41.	Arquitecturas orientadas a servicios	Ing. Jorge Ramírez J., M.Sc. (cand.)	Noviembre 2009

Editado y publicado por Rho-Sigma, S.A., a nombre del Club de Investigación Tecnológica.

San José, Costa Rica. Noviembre 2009

Resumen Ejecutivo

Hoy en día la mayoría de las organizaciones apoyan gran cantidad de sus operaciones en tecnologías de información, especialmente en sistemas basados en software, los cuales son desarrollados internamente, por contrataciones externas o creados por terceros. Esta diversidad de los sistemas de software, sumada al cambio constante y rápido del mercado, hacen que los departamentos de TI siempre estén en la búsqueda nuevas opciones que ayuden a mejorar la calidad y el tiempo de entrega de las aplicaciones. Sin embargo, existen aplicaciones de software legadas, que están soportando funcionalidades importantes de negocio que no están diseñadas para cambiar, esto dificulta el avance rápido, ordenado y ágil según los requerimientos del negocio.

Por estas razones, en la última década las empresas y la academia han dirigido sus esfuerzos hacia el diseño e implementación basados en Arquitectura de Software, cuyo principal objetivo es hacer explícita la estructura de un sistema de software, es decir, determinar cuáles son los componentes que la conforman y cómo estos interactúan entre sí. Como resultado de estos esfuerzos, surge la necesidad de crear patrones de software que permitan la repetición y divulgación de buenas prácticas en el diseño de aplicaciones.

A pesar de contar con la forma de cómo documentar, estructurar y transmitir el conocimiento, la diversidad de plataformas para construir software, ha establecido la necesidad de crear y comunicar aplicaciones en ambientes heterogéneos.

Una respuesta que ayuda a satisfacer estos requerimientos es la Arquitectura Orientada a Servicios (SOA), la cual busca como principal objetivo apalancar la optimización de los procesos de negocios por medio de la identificación, definición, almacenamiento y divulgación de servicios, entendiendo como servicios aquellas funcionalidades que provee el negocio por medio de componentes de software o provistas con la intervención humana.

Desde el punto de vista de TI, una Arquitectura Orientada a Servicios permite el diseño aplicaciones de software preparadas para cambiar, ágiles y adaptables, por su bajo acoplamiento, es decir, sin importar las tecnologías con que estas fueron creadas o la plataforma donde vayan a ejecutar.

Por su capacidad de favorecer ambientes heterogéneos, podemos decir que SOA es una estilo arquitectónico, si se quiere, con una mayor orientación al negocio, pues permite la combinación de servicios para crear nuevas funcionalidades o procesos de negocio y así responder a requerimientos inmediatos o futuros.

Este informe orienta al lector sobre los conceptos básicos que están detrás de una Arquitectura Orientada a Servicios, así como los elementos para introducir tal estilo arquitectónico en su organización. Asimismo, se aclara qué esperar y que **no** esperar de SOA.

Del autor
Jorge Ramírez Jiménez es Ingeniero de la Gerencia de Arquitectura de TI en BAC | Credomatic Network. Es Ingeniero en Computación del Instituto Tecnológico de Costa Rica y está terminando una maestría en Computación en el Instituto Tecnológico de Costa Rica.

Agradecimiento
Un agradecimiento sincero a Ignacio Trejos y los compañeros de la Gerencia de Arquitectura de TI en BAC | Credomatic Network por la confianza brindada para la publicación de este informe, el cual es una gran oportunidad para compartir el conocimiento y experiencias adquiridas en el tema de Arquitectura Orientada a Servicios. Además un sincero agradecimiento a los lectores por sus valiosos comentarios, los cuales contribuyeron a enriquecer el informe.

Nota editorial
Este informe fue revisado por el Dr. Richard Mark Soley (OMG y SOA Consortium), el Dr. Roberto Sasso (Club de Investigación Tecnológica), el M.Sc. Fulvio Lizano (Universidad Nacional), el M.Sc. Gerardo Porras (Banco Central de Costa Rica), el Ing. Luis Chavarría (BAC | Credomatic Network) y el M.Sc. Ignacio Trejos (Club de Investigación Tecnológica). La edición final estuvo a cargo de Ignacio Trejos.

Tabla de contenidos

Tabla de figuras

1. Introducción

Para nadie es un secreto que la globalización ha hecho que las organizaciones deban ser altamente competitivas para mantenerse con vida dentro del mercado y para ello tienen que estar alertas al constante cambio que este experimenta, como por ejemplo: alianzas estratégicas, diversificación de la producción, adquirir o prestar servicios a terceros, etc. Además, las futuras necesidades hacen que las organizaciones deban estar preparadas para reaccionar ya sea creando nuevos procesos de negocio o modificando los existentes. Para ello deben ofrecer productos o servicios con la mejor calidad, en el menor tiempo posible y con un valor agregado superior al de la competencia.

Un factor que ha contribuido a agilizar los procesos de negocio es la inclusión de sistemas de información para automatizar o facilitar el manejo de tareas dentro de dichos procesos. Sin embargo, los sistemas de información de la mayoría de las organizaciones han crecido descontroladamente. Las organizaciones cuentan con sistemas que realizan funcionalidades similares a las de otros sistemas, algunos sistemas son totalmente monolíticos (lo cual dificulta su mantenimiento), no se reutilizan componentes de software existentes en la organización y hasta se toma la decisión de adquirir software que realiza la misma funcionalidad que estos componentes. Por otro lado, la diversidad de plataformas sobre las que se ejecutan los sistemas de software aumenta la dificultad para integrar funcionalidades entre sistemas. Estos y otros factores hacen que las modificaciones o nuevos procesos de negocio muchas veces tarden más de lo esperado, ya que en la mayoría de los casos las decisiones de negocio se ven afectadas o restringidas por la tecnología disponible.

De esta problemática empresarial surgen dos visiones, la que es propia del negocio y la visión del área de tecnología de información, generándose una barrera o brecha debido a que cada uno busca satisfacer sus necesidades. Por ejemplo, el negocio busca tener beneficios en costos, valor agregado y organización. Por su parte, TI busca tener proyectos, proveer productos y generar estándares. Sin embargo, ambas visiones no se pueden tratar por separado, pues una es complemento de la otra: el negocio requiere de la tecnología para dar solución a sus procesos, organizarlos y agilizarlos, y por su parte tecnología requiere de estas necesidades para llevar a cabo sus proyectos.

Para solventar esta brecha nace la arquitectura orientada a servicios, conocida por sus siglas en inglés como SOA, cuya filosofía es la exposición de funcionalidades de negocio como servicios, teniendo en cuenta que, para automatizar algunas de estas funcionalidades, debe existir un componente de software que lo respalde, sin importar la plataforma tecnológica en que este se encuentre. Algunos de los beneficios que puede obtener la organización son:

- La reducción de costos y del tiempo en el proceso de desarrollo de aplicaciones de software que respondan a una necesidad de negocio, esto debido a que SOA busca fundamentalmente la reutilización de componentes.

- SOA permite utilizar la mezcla de nuevas funcionalidades y las ya existentes en la organización.

- Estandarización entre aplicaciones que utilizan un mismo servicio; la tecnología contribuye a disminuir la duplicidad de componentes de software.

El principal objetivo de este informe es que el lector conozca en qué consiste el concepto de arquitectura orientada a servicios. Para esto se responderán preguntas como:

- ¿Qué es arquitectura de software?
- ¿En qué consiste un servicio?
- ¿Cómo está conformada una arquitectura orientada a servicios?
- ¿Cuáles son las características de un servicio en SOA?
- ¿Cuáles son los beneficios que trae SOA a mi organización?
- ¿Qué aspectos deben tomarse en cuenta en la organización para poner en práctica SOA?
- ¿Dónde está mi organización con respecto de SOA?

2. ¿Qué es 'arquitectura de software'?

2.1. Definiciones

En [IBM 2008][1] se recopilan varias definiciones sobre arquitectura de software. A continuación se presentan las más relevantes.

- "*La arquitectura de software de un programa o sistema computacional es la estructura o estructuras del sistema, con las cuales comprendemos los elementos de software, las propiedades visibles externamente de estos elementos, y la relación que existe entre ellos*" [Bass 2003].

  En [IBM 2008] se presenta una explicación de algunos conceptos que abarca esta definición, por ejemplo cuando se habla de propiedades visibles externamente está refiriéndose a aquellos supuestos que otros elementos pueden hacer de un elemento, tales como: si provee servicios, características de rendimiento, manejo de excepciones, recursos compartidos utilizados, etc.

  Por otro lado, si se analiza detenidamente la definición se pueden encontrar varios conceptos implícitos como:

  1. La arquitectura define elementos. La arquitectura plasma la información acerca de cómo los elementos se relacionan unos con otros. Esto significa que la arquitectura específicamente omite alguna información de los elementos, si esta no pertenece a la relación que existe entre ellos. De esta manera, una arquitectura es principalmente una abstracción de un sistema, que oculta detalles de los elementos que no tienen que ver con: cómo estos son utilizados, cómo se relacionan o cómo interactúan con otros elementos.
  2. La definición deja claro que los sistemas pueden constar de más de una estructura, lo cual nos lleva a la existencia de una arquitectura.
  3. La definición implica que cada sistema de software tiene una arquitectura porque cada sistema puede estar compuesto de elementos y relaciones entre ellos. En el más trivial de los casos, un sistema de un solo elemento, puede no ser interesante y probablemente una arquitectura inútil, pero es una arquitectura al fin.
  4. El funcionamiento de un elemento es parte de una arquitectura en tanto este pueda ser observado o percibido desde el punto de vista de otro elemento. Esta característica es la que permite que los elementos puedan interactuar unos con otros, lo cual es claramente parte de una arquitectura.
  5. La definición es indiferente a si la arquitectura es buena o mala para un sistema.

- "*Una arquitectura es el conjunto mínimo de reglas que organizan los acuerdos, iteraciones e interdependencias de las partes o elemento que juntos pueden ser utilizados para formar un sistema de información. Su propósito es asegurar que el sistema conformado satisfaga un conjunto de requisitos especificados*" [ATA 1996].

[1] Todas las referencias aparecen al final del informe.

Las definiciones anteriores nos muestran el concepto de arquitectura de software desde un punto de vista estructural. Según Cuesta [Cuesta 2002], generalmente cuando se trata de definir el concepto arquitectura de software se tiende a relacionar la noción de estructura con la concepción y la especificación. Sin embargo, el concepto no sólo se debe utilizar en ese sentido, ya que conforme avanza la disciplina se han podido identificar al menos cuatro significados:

- Arquitectura como *producto*: en este sentido, la arquitectura es el conjunto formado por la organización, la estructura y la infraestructura de un sistema de software. Desde esta perspectiva, interesa ante todo su descripción, evaluación y análisis.

- Arquitectura como *proceso*: desde este punto de vista se hace referencia al método por seguir para inferir la arquitectura de un sistema, como a la elaboración de una metodología de desarrollo que considere explícitamente el impacto de la arquitectura de software.

- Arquitectura como *campo de estudio*: desde esta perspectiva, se deben considerar tanto los estudios específicos existentes, como una amplia variedad de aspectos fuertemente relacionados, que abarcan desde los métodos formales constituidos en herramienta básica, hasta los patrones de diseño.

- Arquitectura como *profesión*: con frecuencia aparece la figura del arquitecto de software, definido como el encargado de diseñar y mantener la descripción de la arquitectura, por lo que implícitamente es el que proporciona la visión integral del producto.

Finalmente con base en la información que nos brinda Cuesta en [Cuesta 2002], podemos ver el concepto de arquitectura de software como una disciplina de la computación que identifica, organiza, interactúa y documenta los componentes que participan en el funcionamiento de una aplicación de software, así como la metodología que se debe seguir para construir, modificar o reutilizar los componentes pertenecientes a dicha aplicación.

2.2. Roles de la arquitectura de software

El principal rol que típicamente desempeña la arquitectura de software, es el de puente entre los requerimientos y la implementación. Sin embargo, en [Garlan 2000] se presentan por lo menos seis aspectos donde la arquitectura de software es importante para el desarrollo de software:

1. *Entendimiento*: la arquitectura de software simplifica la capacidad de comprender grandes sistemas, presentándolos en un nivel alto de abstracción que permite su comprensión con mayor facilidad.

2. *Reutilización*: la arquitectura de software permite el diseño basado en componentes con los cuales se pueden crear marcos de trabajo (*frameworks*) que permitan y fomenten su reutilización en conjunto o de componentes individuales por medio de patrones.

3. *Construcción*: la arquitectura de software provee descripciones que muestran cuáles son los componentes que la conforman y la interacción entre ellos; estas descripciones facilitan el desarrollo de software.

4. *Evolución*: la arquitectura de software, al tener identificados cada uno de sus componentes y sus interacciones, permite identificar posibles modificaciones o mejoras a la arquitectura en temas como: desempeño, interoperabilidad, prototipos y reutilización.

5. *Análisis*: la arquitectura de software provee oportunidades para realizar análisis de verificación de consistencia, atributos de calidad, dependencias etc.

6. *Administración*: la arquitectura de software facilita la administración de un proyecto de software debido a que clarifica la compresión de requerimientos, estrategias de implementación y riesgos potenciales.

2.3. Beneficios de la arquitectura de software

Todo proyecto o aplicación de software incluye implícitamente el desarrollo de una arquitectura de software, ya que en una aplicación de software interactúan diferentes componentes con el fin de proporcionar una funcionalidad que se derivó de un requerimiento de un usuario. Sin embargo, la práctica de contar con una arquitectura de software debería ser totalmente explicita debido a los beneficios que trae consigo. Mollineda cita algunos de estos beneficios [Mollineda 2005]:

- Evaluación de la solución (diseño + implementación) mediante demostraciones tangibles de sus capacidades desde fases muy tempranas de desarrollo.

- Atención temprana de riesgos relacionados con la arquitectura que, generalmente, coinciden con aquellos que pueden conducir a mayores daños.

- Propicia la construcción incremental del software (integración temprana y planificada) y las correspondientes actividades de verificación.

- Propicia el desarrollo orientado a demostraciones periódicas de productos funcionales.

- Interfaces correctas permiten una cooperación eficiente entre diseñadores e implementadores.

- Facilita la detección e identificación temprana de errores.

Por estos beneficios, es que el proceso de diseño y construcción de software debe involucrar de manera fuerte y comprometida el desarrollo de una arquitectura, a fin de contar con un esqueleto de las aplicaciones que permita acelerar el desarrollo de software.

Además, es importante empezar a invertir tiempo y recursos en adoptar la arquitectura de software como parte del proceso de desarrollo, ya que este en los últimos años se ha convertido en un campo de desarrollo económico para la sociedad como muy bien se plantea en [Lizano 2000]:

- *"La arquitectura de software nos permite "acelerar el paso" para generar un buen nivel de disciplina a la hora de desarrollar software, lo cual hace a la industria de software más competitiva, no sólo en los réditos económicos si no también en la eficiencia del procesos de desarrollo".*

- *"De qué vale tener las mejores condiciones o elementos fundamentales para desarrollar software de primer orden, si no podemos estructurar nuestros procesos de diseño de forma tal que todo ese conocimiento acumulado a lo largo de múltiples proyectos de desarrollo, esté resguardado de la trampa del olvido y la improvisación".*

En este punto es importante responder la pregunta *¿Por qué es importe conocer el concepto de arquitectura de software para comprender SOA?*

La respuesta es sencilla, es importante debido a que una de las típicas ideas equivocadas acerca de SOA es *pensar que SOA provee una arquitectura completa para un sistema* [Lewis 2007].

Como se menciona en [Lewis 2007], SOA no es una arquitectura, es un *estilo arquitectónico* del cual se pueden derivar muchas arquitecturas. Básicamente, un patrón arquitectónico es aquel que provee una guía acerca de los tipos de elementos que posee un sistema y cómo estos interactúan entre sí [Calderón 2006]. Como se mencionó anteriormente, esta característica es apenas es uno de varios roles que desempeña una arquitectura de software.

Otro problema que acarrea tener esta idea equivocada, es que puede llevar a los clientes que no pertenecen a TI a pensar que SOA es un producto que se toma de un estante, se conecta, se enciende y funciona. Esto es totalmente errado, ya que cada empresa u organización utilizará el patrón arquitectónico de acuerdo con sus necesidades y con el nivel de madurez en que se encuentre.

3. ¿Qué es 'arquitectura orientada a servicios'?

3.1. Definición de servicio

En general podemos definir un *servicio* como un conjunto de actividades, posiblemente automatizadas, que buscan responder a una o más necesidades de un cliente [Wikipedia 2008]. Según [Plummer 2005] el concepto de servicio se puede definir según la semántica de la palabra:

- *Cuando se utiliza como sustantivo*, la palabra servicio se refiere a los servicios que son activados. Estos módulos abarcan la implementación técnica de alguna unidad lógica de trabajo, como por ejemplo la lectura del ID de un empleado o el retorno del cálculo de la inversión para una transacción financiera. Los servicios pueden ser atómicos (esto es, que los servicios ofrecen solamente una función básica, como por ejemplo actualizar la existencia de un ítem en un inventario) o compuestos (esto es, que los servicios proveen más de una funcionalidad, por ejemplo el cálculo del precio de un ítem del inventario y después aplicar descuentos y devoluciones).

- *Cuando se utiliza como verbo*[2], el término (dar) servicio se refiere al resultado entregado derivado de la utilización de un servicio. Cuando se utiliza como verbo, (dar) servicio es sinónimo de entrega.

En [Channabasavaiah 2003] se define un servicio como una simple capacidad de negocio (ej. la cotización de una acción), una transacción más compleja (ej. comprometa el inventario) o un servicio del sistema (ej. registro del mensaje entrante).

Además, en [Channabasavaiah 2003] se hace la diferenciación de los servicios por su granularidad (fina o gruesa) cuya diferencia se puede ver en términos de desempeño, facilidad de mantenimiento y reutilización. El nivel de granularidad es una expresión de la riqueza funcional del servicio, en términos de la facilidad con que éste puede combinarse con otros servicios para la creación de una nueva funcionalidad de negocio. Los servicios típicamente son funciones de negocio de grano grueso, como abrir una cuenta, puesto que la operación puede resultar de la ejecución de múltiples servicios u operaciones de grano fino, como verificar identificación y crear cuenta.

De estas definiciones se puede decir que un servicio es el diseño o implementación de la solución a una necesidad de negocio o tecnológica, y que estos pueden realizar una única funcionalidad o pueden estar compuestos por un conjunto de servicios que satisfacen una nueva funcionalidad. Además, un servicio no tendría utilidad alguna si no existe un consumidor que espere un resultado o una entrega del servicio particular que está consumiendo.

Para complementar estás definiciones en [Barco 2006b] se presenta un ejemplo con los siguientes elementos:

- Existen tres individuos, donde cada uno ofrece un servicio distinto. Los vamos a identificar de la siguiente forma:

    1. *El despachador*: encargado de tomar llamadas y organizar entregas.

[2] N.Ed. En español "to service" sería "dar servicio".

2. *Conductor*: encargado de hacer las entregas.

3. *Contador*: encargado de la contabilidad.

Cada uno de estos individuos presta servicios atómicos, gráficamente se vería así:

Figura 1: Ejemplo de servicios atómicos. Tomada de [Erl 2007]

- Una compañía que ofrece el servicio de entregas, requiere el conjunto de los servicios que prestan individualmente las personas del punto anterior. Representado de la siguiente manera

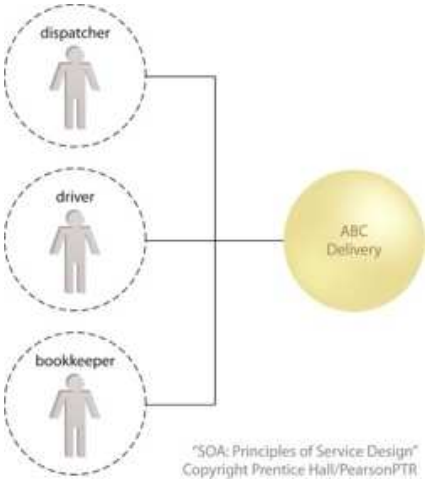

Figura 2: Ejemplo de servicios compuestos. Tomada de [Erl 2007]

Por lo tanto, existe un determinado requerimiento que permite que un grupo de proveedores de servicios individuales colaboren colectivamente en prestar un servicio más grande.

Desde el punto de vista de SOA en [Plummer 2005], un servicio es considerado como la interacción de dos piezas – proveedor y consumidor – donde el proveedor tiene con una interfaz bien descrita, la cual permite que el consumidor sea capaz de utilizar las funcionalidades que el proveedor ofrece.

Un aspecto importante es que el consumidor no necesita entender los detalles de la implementación que ofrece el proveedor del servicio. Esto implica el concepto de "pérdida de acoplamiento", es decir, que el servicio proveedor puede cambiar sin obligar al consumidor a cambiar. Además, un servicio es considerado una colección de habilidades [Barco 2006b], donde cada una puede ser utilizada individualmente, pero están agrupadas dentro de un mismo contexto que es propiamente el servicio que conforman. Por ejemplo, el individuo que ofrece el servicio de conductor, podría tener las habilidades de manejar, llenar hojas de ruta y recolectar pagos. Además, el proveedor del servicio no necesita saber quiénes son sus consumidores.

Figura 3: Servicios como colección de habilidades. Tomada de [Erl 2007]

Con base en la definición de servicio presentada no se debe caer en las siguientes ideas equivocadas acerca de SOA relacionadas con el concepto de servicio [Lewis 2007]:

- *"El desarrollo de aplicaciones es sencillo basándose en servicios"*. El desarrollo de aplicaciones basadas en servicios no es tan simple como invocar algunos de estos servicios y listo. Para lograr el éxito se debe trabajar en temas como:

    - *Publicación de los servicios existentes*, esto se refiere a un repositorio que contenga información relacionada a las características de los servicios, cuál es la interfaz para consumirlo etc.

    - *Composición de servicios:* esto se enfoca en los mecanismos o estándares con los que debe contar la organización para lograr que dos servicios con responsabilidades diferentes logren comunicarse.

    - *Utilización de los servicios:* con respecto de esto se deben buscar los mecanismos y estándares para garantizar el buen funcionamiento de los servicios que se ofrecen.

- *"Es sencillo desarrollar servicios que cualquiera pueda utilizar"*: esta idea errónea no es del todo falsa, sin embargo, su validez está muy ligada al estado actual de las aplicaciones existentes en la organización, a la madurez en los procesos de desarrollo y a la utilización de las herramientas en el desarrollo de los posibles servicios.

- *"Los servicios están solamente ligados a necesidades de negocio"*: dentro del portafolio de servicios podemos contar con servicios de negocio y servicios de infraestructura o de tecnología. Estos últimos forman parte de la implementación de los servicios de negocio, proveen capacidades reutilizables de infraestructura a múltiples servicios o facilitan la administración de los servicios. Por ejemplo, consideremos el servicio de negocio del

procesamiento de planillas; parte de las funcionalidades que presta este servicio es la carga del archivo con la planilla. Para resolver esta prestación es necesario consumir un servicio de infraestructura que permite tomar el archivo, trasmitirlo y, una vez que finalizó la trasmisión, tomarlo para iniciar el procesamiento.

3.2. Definición de SOA

Como bien mencionan [Evdemon 2005] y [Gutiérrez 2005], esencialmente en SOA se resumen muchos de los esfuerzos de múltiples tecnologías previas, que tenían como objetivo la comunicación entre distintas aplicaciones independientemente de la ubicación de estas. Algunos ejemplos de estas tecnologías son:

- La plataforma J2SE con RMI (Remote Method Invocation).
- CORBA (Common Object-Request Broker Architecture).
- Otra alternativa era utilizar un protocolo diseñado exclusivamente para la comunicación entre pares de aplicaciones.

Además, en lo que respecta al diseño de los servicios propiamente, SOA promueve el encapsulamiento, la abstracción y las interfaces bien definidas; estos conceptos tienen una línea ancestral en los tipos abstractos de datos, la orientación a objetos y los modelos de componentes [Sasso 1992, Trejos 1999, Luna 2000].

Las siguientes son definiciones de SOA:

- *"SOA es un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios"* [Microsoft 2006].

- *"Es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario"* [García 2007].

- *"SOA es una forma de organizar el software de tal manera que las compañías puedan responder rápidamente al cambio de requerimientos del mercado. Se basa en servicios, los cuales son unidades de software que se ejecutan en una red"* [Margolis 2007].

- *"SOA es una arquitectura de software que propone la construcción de aplicaciones mediante el ensamblado de bloques reusables, débilmente acoplados y altamente interoperables, cada uno de los cuales es representado como un servicio. Los mismos pueden encontrarse distribuidos y pertenecer potencialmente a diferentes propietarios"* [Canto 2006].

Las definiciones anteriores nos demuestran que el criterio de definición de SOA es muy diverso. Sin embargo, como se menciona [Rotem 2007] la definición de SOA se debe realizar desde los puntos de vista de negocio y tecnología, ya que cada uno lo enfoca de acuerdo con sus necesidades y objetivos.

El *negocio* enfoca su definición basándose en el concepto de *orientación a servicios* (***SO***A), es decir, en la búsqueda e identificación de áreas de negocio, seguido por la definición de servicios que representen las capacidades de estas áreas con el fin de orquestar con estos servicios un proceso de negocio.

Por otro lado, el punto de vista de *tecnología* está basado en la arquitectura (SO***A***), es decir, cómo soportar estos servicios con un componente de software y cómo comunicar de una manera ágil y sencilla estos componentes.

Las definiciones anteriores y los puntos de vista presentados nos llevan a decir que el objetivo primordial de SOA es: promover la reutilización y componentización de funcionalidades (automatizadas o no) existentes en la compañía, denominados servicios, los cuales se encuentran en diferentes lugares y además, están construidos en diferentes lenguajes y plataformas. Esto con el único fin de agilizar la definición, construcción y divulgación de procesos de negocio que respondan al constante cambio de los requerimientos del mercado.

A partir de esta información es importante aclarar la idea errónea: *"SOA puede ser implementada rápidamente"* [Lewis 2007]. Esta es una idea equivocada por varias razones:

- Como lo hemos mencionado, existen puntos de vista diferentes que deben ser llevados a un punto de conciliación. Esto la mayoría de las veces implica un cambio cultural fuerte en la organización.
- Las empresas no pueden pensar en una estrategia de migración estilo "big bang" ya que podrían fracasar fácilmente. Primero se debe asegurar el mínimo impacto a los usuarios finales de los procesos de la empresa. Además, desde el punto de vista de TI, el proceso de construcción de los servicios es diferente al proceso tradicional que se sigue en la construcción de aplicaciones de software.
- La organización debe entender cuáles son los beneficios que SOA le trae en el contexto de su negocio.
- Es importante ejecutar planes pilotos con el fin de disminuir los riesgos de poner en práctica SOA en el negocio. Además, con la experiencia obtenida en estos pilotos se deben refinar los procesos de desarrollo utilizados para la adopción de SOA.

3.3. Componentes de una arquitectura orientada a servicios

En [Gutiérrez 2005] se enumeran tres componentes que participan en una arquitectura orientada a servicios:

- **El servicio**: una arquitectura orientada a servicios puede estar compuesta por más de un servicio, donde cada servicio tiene una funcionalidad específica y, además, este puede ser utilizado por el resto de los servicios que pertenecen a la arquitectura.
- **El directorio o catálogo de servicios**: el directorio es el responsable de mantener centralizada la información de cada uno de los servicios, por ejemplo: la funcionalidad del servicio, cómo se debe utilizar el servicio, etc.
- **El cliente o consumidor**: este componente es el que utiliza los servicios que están publicados a través del directorio. Un cliente puede ser una aplicación u otro servicio perteneciente a la arquitectura.

Además, en [Gutiérrez 2005] se listan las relaciones entre estos componentes:

- **Localización de servicios**: los posibles clientes de un servicio utilizan el directorio para obtener la información del servicio que desean consumir.
- **Publicación de servicios**: los servicios se publican por medio del directorio para que los clientes tengan disponible la información que respecta a estos.

- **La comunicación entre los servicios y el cliente**: el cliente realiza las peticiones al servicio, por medio del protocolo de comunicación establecido previamente para la arquitectura, posteriormente el servicio retorna la respuesta al cliente utilizando el mismo protocolo.

En ambientes heterogéneos estas relaciones se podrían llevar a cabo por medio de un cuarto componente denominado *Bus de Servicios Empresarial* [Gómez 2009, Linthicum 2006], más conocido como ESB por sus siglas en inglés.

3.4. Bus de Servicios Empresarial

Un ESB consiste en un software mediador cuyo objetivo principal es permitir la comunicación entre los participantes de una arquitectura SOA (clientes, directorio y servicios); esta comunicación se realiza por medio de un protocolo de mensajería (XML, trama plana, petición HTTP, etc). Además, este soporte debe ser heterogéneo porque en la mayoría de las ocasiones los participantes se encuentran implementados en diferentes tecnologías y los paradigmas de comunicación entre ellos también son heterogéneos pues pueden ser síncronos o asíncronos [Otero 2007].

En [Novegil 2008] se presentan algunas de las principales características de un ESB:

- **Enrutamiento basado en el contenido**: con base en la información del mensaje, el ESB debe determinar a qué servicio o servicios ha de invocar.
- **Transformación de mensajes**: en ocasiones es necesaria la transformación de los mensajes entre participantes de una arquitectura orientada a servicios. Por ejemplo, algún participante que solamente "habla" XML requiere comunicarse con otro que solamente "habla" trama plana. Para esto un ESB debe proveer motores de transformación y "parseadores"[3] que permitan exitosamente esta comunicación.
- **Proxy de servicios**: el ESB funciona como un puente entre el cliente y el servicio; es decir, el cliente no tiene por qué conocer la implementación del servicio.
- **Conversión de protocolos**: esto debido a que una arquitectura orientada a servicios se desenvuelve en un ambiente heterogéneo donde los protocolos de comunicación para muchas aplicaciones y servicios no son los mismos.
- **Auditorías y bitácoras de mensajes**: el ESB, al ser el punto neurálgico de la arquitectura orientada a servicios, permite controlar en un solo lugar las bitácoras y la información de auditoría.
- **Manejo de excepciones**: el ESB, al centralizar las funcionalidades, permite gestionar las excepciones de una manera estandarizada.
- **Seguridad de los servicios**: por medio del ESB se pueden definir políticas de seguridad para los servicios y establecer requisitos para su invocación o ejecución.
- **Acceso a repositorios de servicios**: es dentro del ESB donde se implementa el directorio o catálogos de servicios mencionado en [Gutiérrez 2005].
- **Validación, enriquecimiento, transformación y operación de los mensajes**: se puede decir que estas son las 4 características estrellas de un ESB.
    - *Validación de la información,* de los mensajes entrantes, sobre los que se pueden aplicar distintas condiciones y verificaciones que hagan cumplir los requisitos establecidos.

[3] Analizadores de la estructura sintáctica.

- *Enriquecimiento de los mensajes*, posibilidad de añadir información o meta-información adicional para cumplir las necesidades de integración en cada momento.
- *Transformación de los mensajes*, normalmente entre los diversos modelos de datos del conjunto de servicios y aplicaciones a integrar.
- *Operación* de los mensajes o enrutamiento en función de directivas preestablecidas o a partir de meta-información incluida en la propia comunicación.

Es importante destacar que algunas veces no es necesario que dentro de una arquitectura orientada a servicios exista un ESB. Cuando el ambiente donde se desenvuelven las funcionalidades de negocio es homogéneo, no se requiere de una pieza de software tan compleja como un ESB. También podría desearse no contar con todas las funcionalidades que un ESB provee como producto de software. Sin embargo, si se desea prescindir del ESB es necesario que existan los componentes necesarios para: enrutar los mensajes e interpretar la información que estos mensajes contengan, según el protocolo establecido.

Es importante conocer esto porque como se menciona en [Gómez 2009] y en [Linthicum 2006], *"ESB se centra en la integración de sistemas, específicamente hablando de los protocolos, plataformas y formas de acceso como http, ftp, sockets, etc. Tengamos en cuenta que la integración es una parte importante, pero SOA es mucho más que eso. Incluso si utilizamos un ESB eso no hace que tengamos SOA; lograr SOA va a depender de la forma en que modelamos y concebimos a nuestros procesos de negocio"*.

El ESB es un concepto afín al de *Object Request Broker* y se sitúa en una línea evolutiva arquitectónica y tecnológica que ha pasado por CORBA y otros elementos mediadores de software ("*middleware*"); véase [OMG-CORBA 2009]. Una línea reciente es hacer interoperabilidad basada en estándares definidos por modelos para industrias específicas; véase [OMG-MDMI 2009] como una demostración de concepto sobre el futuro previsible en la interoperabilidad en el sector de servicios financieros.

3.5. Organización de una aplicación orientada a servicios

Enfocándonos en el concepto de servicio, SOA propone varias capas que exponen funcionalidad, fundamentalmente de negocio, que permiten la composición de aplicaciones a partir de estos servicios. En [Evdemon 2005], [Barco 2006b], [Margolis 2007] se proponen tres tipos de servicios:

1. *Servicios controladores*: reciben las peticiones de los clientes y además conocen cómo están conformados los servicios, esto le permite coordinar las llamadas a cada uno de los servicios principalmente en el caso de los servicios compuestos, es decir, los servicios controladores conocen el orden en que se deben invocar cada uno de los servicios. Generalmente estos servicios se exponen a las aplicaciones consumidoras a través de un ESB.

2. *Servicios de negocio*: estos servicios están ligados a un proceso de negocio lo que los hace poco reutilizables ya que resuelven un problema específico. Generalmente son invocados por los servicios controladores.

3. *Servicios reutilizables*: como su nombre lo dice pueden ser utilizados en solución de varios problemas o requerimientos. Existen dos tipos, los *servicios orientados al negocio* que representan una tarea de negocio altamente reutilizable entre aplicaciones y los servicios tecnológicos encargados de encapsular una determinada tecnología, como por ejemplo servicios de acceso a bases de datos relacionales.

Gráficamente una arquitectura de software de una aplicación orientada a servicios se puede ver como en la siguiente figura:

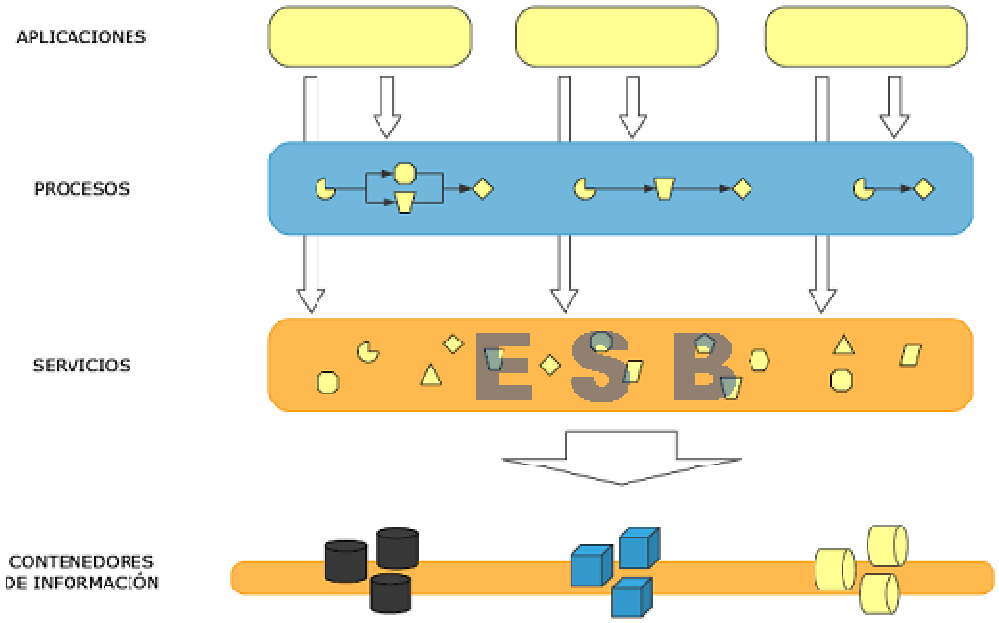

Figura 4: Arquitectura Orientada a Servicios. Tomada de [Recena 2007]

En esta figura se puede observar cómo los procesos invocan los diferentes tipos de servicios por medio del ESB en el caso de que se requiera contar con un componente como este [Gómez 2009, Linthicum 2006].

3.6. Principios SOA

En esta sección estudiaremos cuáles deben ser los principios con lo que debe cumplir un servicio que pertenezca a una arquitectura orientada a servicios. Estos principios giran en torno al concepto de reutilización, por lo que es importante conocer acerca de él.

3.6.1. Reutilización de software

Definición

La idea de contar con elementos reutilizables de software es de larga data en el mundo informático. En 1968, Douglas McIlroy enunciaba el concepto de 'componente de software' en estos términos: "*La característica más importante de una industria de componentes de software es que ofrecerá familias de rutinas para cualquier tarea. ...El comprador de un componente de una familia escogerá aquel que satisfaga sus necesidades exactas. Consultará un catálogo que*

ofrezca rutinas en variados grados de precisión, robustez, rendimiento espacio-tiempo y generalidad. Tendrá la confianza de que cada rutina de la familia es de alta calidad – confiable y eficiente. Esperará que la rutina sea inteligible, sin duda expresada en un lenguaje de alto nivel apropiado para el propósito del componente, aunque no necesariamente compilable instantáneamente en cualquier procesador. Esperará que se construyan racionalmente familias de rutinas de manera que las familias calcen como bloques de construcción. En resumen: él debería ser capaz de ver con seguridad a los componentes como cajas negras."

En lo que respecta a *reutilización* de software existen varios puntos de vista, con definiciones como las que se presentan en [Sametinger 1997]:

- "*Reutilización es el uso de cualquier información que un desarrollador necesite en el proceso de creación de software*" [Freeman 1987].
- "*Reutilización es todo lo relacionado con un proyecto de software incluyendo el conocimiento*" [Basili 1988].
- "*Reutilización es la utilización de componentes de software existentes en un nuevo contexto, ya sea en el mismo proyecto para el cual fueron creados o en un proyecto nuevo*" [Braun 1994].
- "*Reutilización es la capacidad que tiene un componente de software desarrollado previamente de ser utilizado nuevamente o repetidamente sin modificaciones*" [Cooper 1994].

Con base en estas definiciones podemos decir que reutilización de software es la posibilidad que tienen los desarrolladores de incorporar o adaptar en su proceso de desarrollo activos de software previamente generados o construidos en el ciclo de vida de desarrollo de software de un proyecto anterior. Los activos pueden ser: conocimiento generado, documentación o componentes de software [Torralba 2004].

Desde el punto de vista de SOA, el concepto de reutilización lo podemos ver como la incorporación de los servicios *existentes*, automatizados o no, en uno o varios procesos de negocios.

Beneficios de la reutilización

Los beneficios que se pueden obtener de la reutilización se clasifican muy bien en [Sametinger 1997] así:

- **Mejoras en calidad**: esta categoría abarca mejoras en calidad, productividad, desempeño, confiabilidad e interoperabilidad.
  - *Calidad*: las mejoras a los activos se acumulan conforme estos se reutilizan. Todos los proyectos que los utilicen se verán beneficiados con dichas mejoras.
  - *Productividad*: las mejoras que trae la reutilización a la productividad por ejemplo se ve reflejada en la disminución de líneas de código que se deben digitar, disminución en los esfuerzos en la etapa de pruebas y en el ahorro en la etapa de análisis y diseño.
  - *Desempeño*: es más sencillo optimizar el funcionamiento de un solo componente que optimizar el funcionamiento de "n" aplicaciones.
  - *Confiabilidad*: esto se refleja en la confianza que adquieren los activos que se van a reutilizar cuando han sido sometidos a una etapa rigurosa de pruebas.
  - *Interoperabilidad*: se refiere a los beneficios de reutilizar los mismos componentes para implementar la comunicación de las interfaces de cada uno de los componentes que conforman un sistema heterogéneo.

- **Mejoras en la reducción de esfuerzos**: esta categoría abarca mejoras en trabajo redundante, tiempo de entrega, documentación, costos de mantenimiento, costos de capacitación y tamaño de los equipos de trabajo.
  - *Trabajo redundante y tiempo de desarrollo*: cuando todos los proyectos son desarrollados desde cero se incrementa las posibilidades de un desarrollo redundante en muchas partes del sistema como interfaces, comunicaciones, algoritmos básicos etc. Esto se puede evitar con la reutilización.
  - *Tiempo de entrega*: al evitar el trabajo redundante y disminuir el tiempo de desarrollo se da un impacto beneficioso en el tiempo de entrega del proyecto.
  - *Documentación*: la reutilización de código disminuye la cantidad de documentación en la creación de un sistema. Además, esta es compartida, trasmitida y mejorada entre cada uno de los desarrolladores que reutilice el activo.
  - *Costos de mantenimiento*: todos los proyectos que utilicen un activo reutilizable se benefician cuando este es mejorado. Con esto se evita dar mantenimiento cada aplicación por separado.
  - *Tiempo de capacitación*: cuanto más se utilicen activos reutilizables estos van a ser aún más familiares para los desarrolladores lo que implica una reducción en el tiempo de aprendizaje del componente reutilizable cuando este vuelva a ser utilizado por el mismo desarrollador.
  - *Tamaño del equipo de trabajo*: al tener que desarrollar menos, el trabajo de los equipos y su número de integrantes disminuyen, mejorando la comunicación entre sus integrantes y por ende la productividad del equipo.

- **Otros beneficios** que se pueden obtener de la reutilización son:
  - *Construcción rápida de prototipos*: utilizar activos reutilizables da la posibilidad de construir prototipos rápidamente. Con esto se puede obtener retroalimentación temprana del cliente del sistema, lo que permite muchas veces descubrir o ver requerimientos que no se analizaron en la etapa de definición.
  - *Transmisión de la experiencia obtenida*: a partir de la reutilización de software se puede obtener estándares y patrones para el desarrollo de nuevos componentes, siempre y cuando los componentes reutilizados estén bien diseñados.

Estos beneficios que se obtienen de la reutilización también se pueden justificar desde el punto de vista económico, según se presenta en [Favaro 1998].

- *Beneficios operacionales*: estos consisten en mejorar la calidad y la productividad, así como reducir el costo de mantenimiento.
- *Beneficios estratégicos*: son los que otorgan la posibilidad de entrar en nuevos mercados o los que dan flexibilidad para responder a la competencia y cambiar las condiciones del mercado. Por ejemplo beneficios en la calidad de los sistemas, productividad, construcción rápida de prototipos.

Obstáculos en la construcción de software reutilizable

La reutilización de software plantea un reto importante para la organización que desee implantar este proceso ya que se deben evadir una serie de obstáculos como los que se presentan en [Sametinger 1997].

Obstáculos gerenciales y organizacionales

- Falta de apoyo gerencial: como todo proyecto, la reutilización de software requiere del patrocinio de los altos mandos de la organización ya que ellos deben estar claros de la inversión que requiere y los beneficios que se obtienen.
- Administración de proyectos: la reutilización plantea cambios en la forma que se administran tradicionalmente los proyectos.
- Falta de procedimientos explícitos: cuando se implante el proceso de reutilización de software es necesario que previamente esté delimitado el alcance que este tiene dentro del ciclo de vida de desarrollo, por ejemplo en las etapas de diseño, planeamiento y estimación.
- Estructuras organizacionales inadecuadas: si la organización toma la decisión de adoptar el proceso de reutilización, todas las áreas que la conformen deben adoptarla también. Por ejemplo no se puede pensar en un solo equipo que se encargue de proveer, mantener y administrar los activos reutilizables, toda la organización debe estar comprometida.
- Síndrome del "no fue inventado aquí": muchas veces las organizaciones prefieren desarrollar sus propias soluciones por temor a utilizar soluciones creadas por terceros.

Obstáculos económicos

Estos obstáculos giran en torno a tres ideas principales: el costo de hacer algo reutilizable, el costo de reutilizar el activo y el costo de definir e implementar el proceso de reutilización. Un aspecto importante que se debe tener claro con respecto de la reutilización es que los beneficios por obtener *no* pueden ser a corto plazo, ya que primero se deben crear los activos reutilizables e implantar todo un proceso organizacional para utilizarlos; esto no conlleva erogaciones, peor es el cambio cultural en la organización sí es costoso.

Obstáculos técnicos

Dentro de este tipo de obstáculos están:

- Dificultad para encontrar componentes reutilizables, ya que los componentes existentes no están bien clasificados o no existe un repositorio para organizar este conocimiento[4].
- Los componentes encontrados después de un análisis minucioso no son candidatos a ser reutilizados, pues su adaptación sería más costosa que hacer componentes nuevos.
- Muchas veces se piensa de manera errónea que para implantar un proceso de reutilización es necesario tener una tecnología orientada a objetos.
- Frecuentemente los componentes disponibles requieren modificaciones, pero debe hacerse un cuidadoso análisis del impacto que estos cambios pueden traer.
- Cuando se crean componentes reutilizables siempre se deben pensar en que estos requieren ser integrados en diferentes sistemas. Algunas veces la integración falla por un mal diseño del componente.

Personalmente creo que la mayoría de los obstáculos que se vayan a presentar son parte de lo que normalmente deberá enfrentar una organización que decida implantar un proceso de reutilización; solamente con la experiencia que se adquiere en este proceso se pueden sortear tales obstáculos.

[4] Hay una dicotomía entre la *granularidad* (del componente o servicio) y *administrabilidad*, los componentes de grano más fino son más fáciles de reutilizar, pero se hacen más difíciles de administrar.

3.6.2. Principios de diseño de servicios

En [Barco 2006a] y [Erl 2007] se listan varios principios que deben cumplir los servicios que participen en una arquitectura orientada a servicios; estos son:

- *Los servicios deben ser reutilizables*: los servicios deben ser diseñados pensando en que varias aplicaciones pueden requerir su utilización. Es decir, hacer que las capacidades del servicio sean útiles para más de un propósito.
- *Los servicios deben proporcionar un contrato*: todo servicio desarrollado debe proporcionar un contrato, con las siguientes características como mínimo: el nombre del servicio, su forma de acceso, funcionalidades que ofrece, los datos de entrada de cada una de estas y los datos de salida.
- *Los servicios deben tener bajo acoplamiento*: los servicios tienen que ser independientes los unos de los otros. El cambio en un servicio no tiene porqué obligar a otros servicios a cambiar.
- *Los servicios deben permitir la composición*: todo servicio debe ser construido de tal manera que pueda ser utilizado (como elemento) para construir servicios genéricos de más alto nivel.
- *Los servicios deben ser autónomos*: todo servicio debe tener su propio entorno de ejecución. De esta manera el servicio es totalmente independiente y se asegura que así podrá ser reutilizable desde la perspectiva de la plataforma de ejecución.
- *Los servicios no deben tener estado*: un servicio no debe guardar ningún tipo de información, es decir, debe tener solamente una responsabilidad.
- *Los servicios deben poder ser descubiertos*: todo servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades. Esto se logra por medio del desarrollo de un directorio o catálogo de servicios.

Estos principios aclaran la idea errónea: "*Los sistemas legados pueden ser integrados fácilmente en un ambiente SOA"* [Lewis 2007]. Muchas de las aplicaciones legadas son aplicaciones monolíticas, que no permiten que el diseño de servicios a partir de sus funcionalidades sea sencillo ya que caen dentro de los denominados "olores del software" [Martin 2003]:

- *Rigidez*: es la dureza que tiene el software para ser cambiado, es decir, realizar una modificación sencilla implica una cascada de cambios en toda la aplicación.
- *Fragilidad*: es la tendencia que posee que una aplicación de romperse en muchos lugares cuando se realiza una modificación.
- *Inmovilidad*: este síntoma se refiere a que existen partes de una aplicación que pueden ser reutilizadas en otros sistemas, pero el esfuerzo y riesgo de separarlos del sistema original son muy grandes.
- *Viscosidad*: cuando un desarrollador se enfrenta a un cambio existen varias formas de hacerlo: las que preservan el diseño y las que no, es decir, la forma correcta y la incorrecta. La viscosidad de una aplicación o un de un diseño es cuando implementar el cambio de la forma correcta es más complicado que hacerlo de la forma incorrecta.
- *Complejidad innecesaria*: se refiere a las funcionalidades implementadas y que no son utilizadas. Estos son los cambios que son detectados anticipadamente y que sería complicado implantarlos posteriormente.
- *Repetición innecesaria*: este olor se refiere a estructuras repetitivas que podrían unificarse en una sola.

- *Opacidad*: es la tendencia que tienen las aplicaciones a ser difíciles de leer y entender.

Esta problemática nos lleva a identificar las razones por las que una aplicación monolítica escasamente brinda valor agregado en una organización:

1. Las aplicaciones monolíticas han sufrido gran cantidad de cambios a lo largo del tiempo, y estos cambios han sido realizados por muchas personas, que inclusive ya no laboran para la organización.
2. Las aplicaciones actuales son muy grandes y complejas, por lo que es difícil y peligroso imaginarse que estas sean implementadas con una estructura monolítica ya que un cambio en una de las funcionalidades podría afectar el funcionamiento total de la aplicación.
3. Los equipos que desarrollan y mantienen las aplicaciones tendrían que estar muy compenetrados, ya que toda la implementación está concentrada en un solo componente, no pueden dejar escapar un detalle de cómo interaccionan cada una de las capas que conforman la aplicación.
4. El desarrollo de componentes en diferentes lenguajes o plataformas queda prácticamente descartado, por el fuerte acoplamiento que existe entre todas las capas de la estructura de la aplicación.
5. El mantenimiento y la depuración se vuelven complicados debido a que la tarea de encontrar el punto de falla o de modificación es muy difícil al estar toda la implementación en un solo componente.
6. Muchas veces reutilizar funcionalidades de una aplicación monolítica en otro sistema es prácticamente imposible, por el trabajo que conlleva separar esta funcionalidad de la estructura de la aplicación.

Por lo tanto, una organización típica que desea obtener beneficios a largo plazo al implantar SOA, debe dividir las aplicaciones monolíticas en componentes con una granularidad más fina y de esta manera obtener un mayor valor agregado de los beneficios que otorgan la componentización y la reutilización.

4. ¿Qué beneficios aporta SOA?

Esta sección se desarrolla desde la perspectiva de los beneficios que puede obtener una organización con la implementación de una arquitectura orientada a servicios. En [Tam 2006] se plantean interrogantes típicas de una organización que busca mayor rapidez y eficiencia; estas razones hace que SOA en la actualidad se esté convirtiendo en una de las opciones más robustas para cumplir con los siguientes requisitos:

- Lanzamiento de nuevos servicios de negocio a clientes, y facilidad de integración de los proveedores a ellos.
- Mejoramiento de la agilidad del negocio y flexibilidad en los procesos.
- Reducción en el número de interfaces entre sistemas y maximizar la reutilización.

Con base en los tres puntos anteriores, los beneficios de SOA se pueden ver desde dos perspectivas diferentes: desde el punto de vista de negocio y desde el de la tecnología [Microsoft 2006]. Los beneficios que obtiene el negocio son:

- Mejora la toma de decisiones, ya que SOA ayuda a documentar el modelo de negocio el cual se puede utilizar en la integración y la respuesta a las nuevas necesidades del dinámico mundo de los negocios.
- Mejora la productividad gracias a la facilidad que provee para adaptarse rápidamente a las nuevas reglas de negocio basadas en los servicios disponibles.
- Potencia la relación con los clientes y proveedores debido al incremento en la velocidad para crear nuevos servicios que permitan la integración entre proveedores o la adquisición de nuevos clientes, además de ofrecer una comunicación ordenada entre ellos, como lo presenta la siguiente figura:

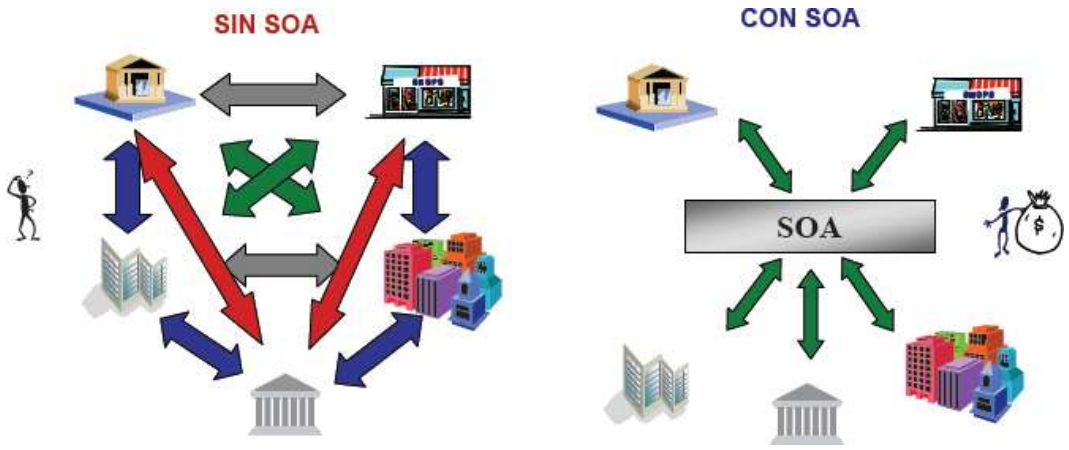

Figura 5: Flexibilidad de SOA para la integración ordenada. Tomado de [Tolosa 2009]

En lo que respecta TI, los principales beneficios de SOA son:

- Aplicaciones más productivas y flexibles: SOA contribuye a eliminar el espagueti de interfaces entre sistemas. Las aplicaciones que requieren conectarse unas con otras, no lo hacen directamente si no a través de algún componente responsable de la integración, por ejemplo un ESB, como lo muestra la siguiente figura:

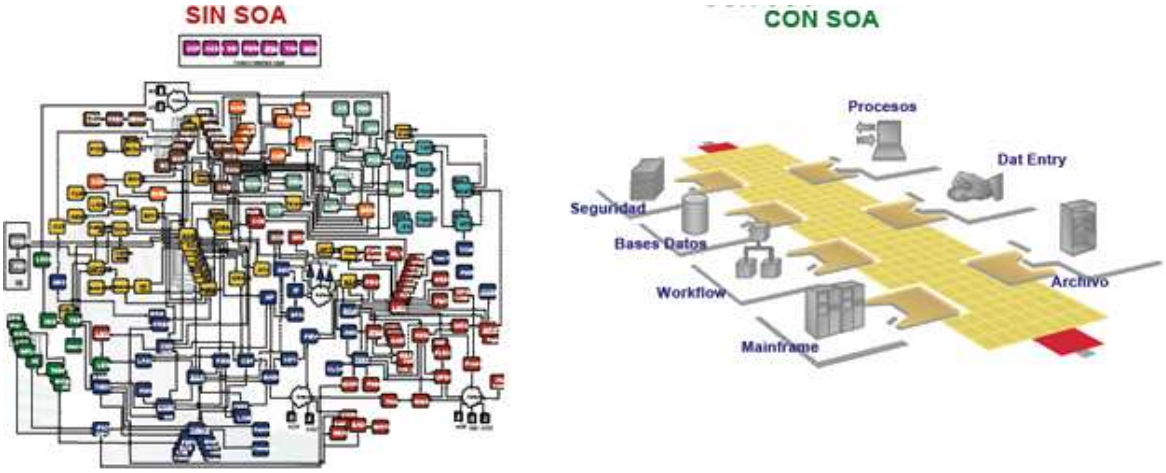

Figura 6: SOA ayuda a eliminar el espagueti de aplicaciones. Tomado de [Tolosa 2009]

- Eliminar el desorden de conexiones entre aplicaciones, hace el desarrollo de estas más rápido y económico.
- La reutilización de servicios en distintas aplicaciones disminuye el riesgo en la implementación de los proyectos ya que se aprovechan los beneficios de los componentes ya implementados.
- Los servicios que se prestan a los clientes incrementan en calidad, debido a que una mejora en uno de los servicios trae consigo mejora a todos los procesos que lo utilizan.

A pesar de los beneficios que se puedan obtener desde ambos puntos de vista, el mayor logro que puede tener una empresa de la implementación Arquitectura Orientada a Servicios, es la colaboración que esta genera entre TI y negocio [OMG 2007], es decir, las organizaciones deben evitar pensar en estrategia de TI y estrategia de negocio por separado. Las siguientes figuras muestran cómo ha sido tradicionalmente la relación negocio-TI y cómo debería ser, desde el punto de vista SOA.

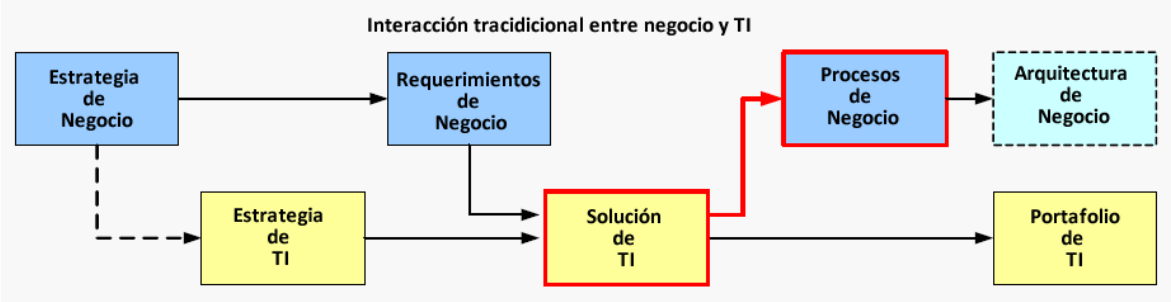

Figura 7: Interacción tradicional negocio-TI. Tomada de [OMG 2007]

La figura anterior nos muestra que como resultado de las estrategias de TI y las interacciones negocio-TI, se han obtenido soluciones de TI que *definen o dirigen* los procesos y la arquitectura de negocio, en lugar de ser lo contrario: el negocio marcando la pauta de las soluciones de TI que se entregan.

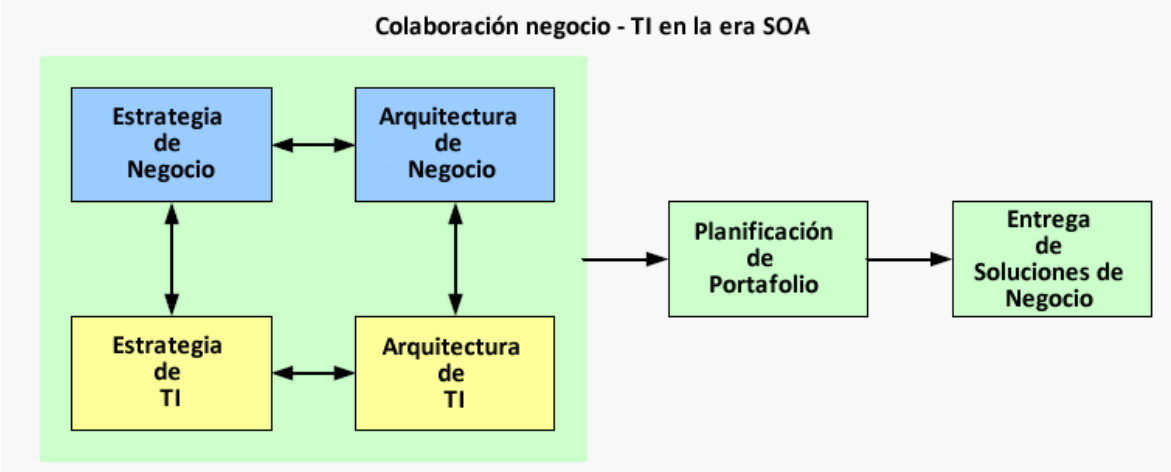

Figura 8: Colaboración negocio - TI en la era SOA. Tomada de [OMG 2007]

Esta figura nos muestra que SOA busca que exista un alineamiento entre las estrategias de negocio y TI; si esta comunicación se realiza exitosamente, la arquitectura de TI contribuye a hacer realidad la arquitectura de negocio. Esta colaboración en estrategia y arquitectura, entre TI y negocio, se ve reflejada en un portafolio (cartera) de soluciones que pueden ser entregadas en beneficio de la organización.

En [Michelson 2009] se presenta una propuesta de hacia donde deberían ir encaminada la colaboración negocio-TI en la era SOA, lo cual se resume en la siguiente figura.

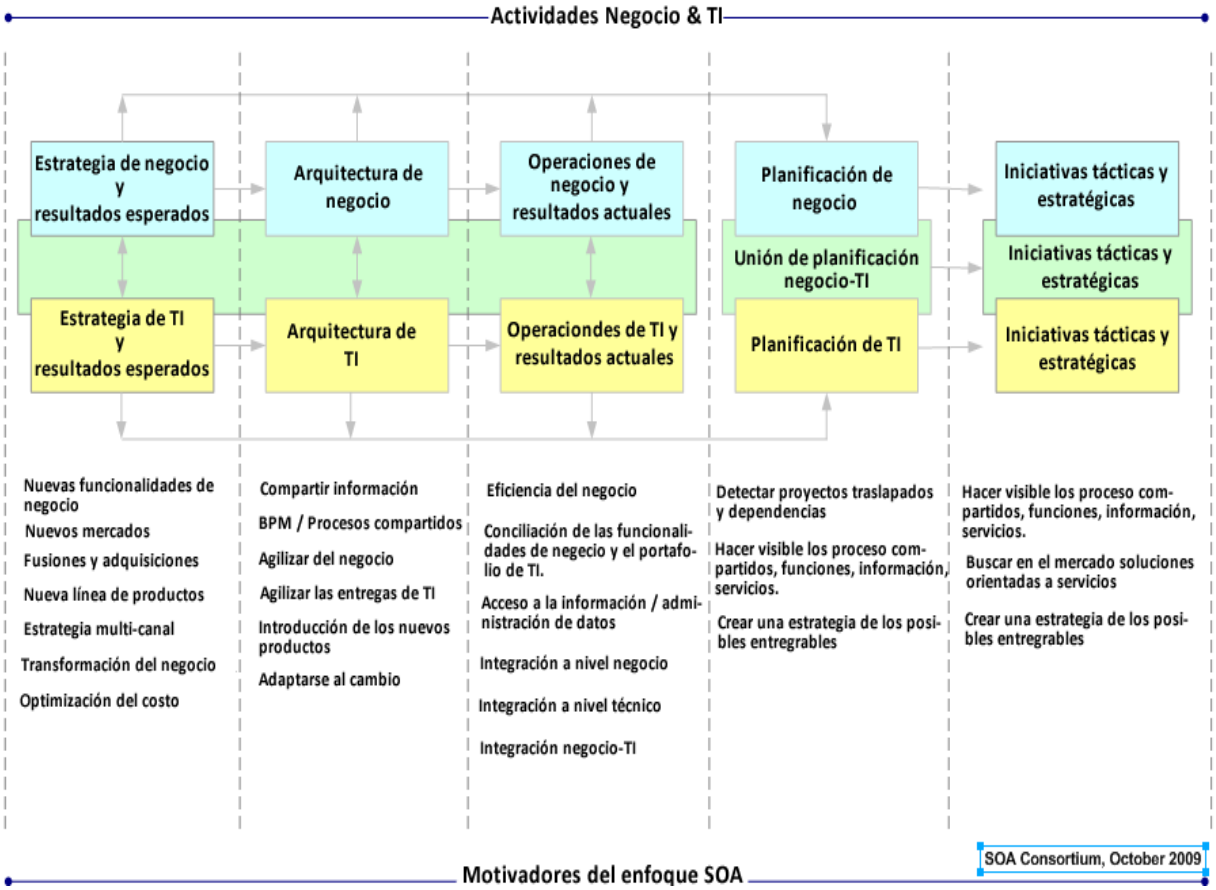

Figura 9: Motivadores del enfoque SOA, asociados a las actividades de negocio y TI

Para lograr alcanzar estos beneficios es importante no cometer en la implementación de SOA los siguientes errores [Infobae Profesional 2008]:

- *Adoptar una política de "big bang"*: en el camino hacia SOA muchas de las organizaciones buscan transformar todos los sistemas a servicios, sin tomar en cuenta cuáles se utilizan realmente. Convertir todos los sistemas no es rentable y muchas veces no es importante para el negocio.
- *Fallar en la inclusión de los analistas de negocios*: muchas veces las organizaciones se concentran en la implementación del servicio, dejando de lado el punto de vista del experto de negocio.
- *Invertir más tiempo en productos SOA que en planificación SOA*: en primera instancia, las iniciativas deberían estar enfocadas al planeamiento y la preparación de los proyectos hacia SOA, y solo una vez que exista esta planificación se debería pensar cuáles son los productos que ayudaran a soportarla. Sin embargo, en muchos casos la estrategia se da en sentido contrario.
- *Abordar los proyectos más grandes en primer lugar*: la mejor práctica con respecto de SOA es empezar con proyectos pequeños pero útiles, poco visibles y de bajo riesgo; con esto se crea una base de conocimiento y experiencia que permite incrementar la complejidad de los siguientes proyectos.
- *Olvidarse de que SOA es un tema de negocios*: cuando el campo tecnológico es el que toma predominancia en el proceso SOA, el riesgo de salirse de curso es grande, ya que el aspecto tecnológico es el que da soporte a los procesos de negocio que al fin y al cabo es el aspecto más importante.
- *Comprar nuevos productos cuando las inversiones actuales son suficientes*: muchas veces se adquiere hardware, software y productos específicos para la implementación de SOA, cuando los recursos disponibles son suficientes para iniciar el proceso. Esto sucede debido a que primero se piensa en los productos en lugar de crear la estrategia y planificación de los proyectos SOA.
- *No comprender el rol de los jugadores clave dentro de la empresa*: es importante conocer cuáles serán los roles y responsabilidades de los participantes y afectados dentro del proceso SOA.
- *Esperar que el proceso SOA se propague rápidamente*: muchas organizaciones esperan que SOA se propague de una unidad de negocio a otra rápidamente y se frustran cuando esto no sucede. Hay que comprender que esta rapidez depende del nivel de madurez en que la organización se encuentre con respecto de SOA; ver más adelante y [Sonic 2005a, Sonic 2005b].

5. SOA en el Negocio

Según [Duré 2005], si una organización planea implementar una arquitectura basada en SOA en necesario que esta avance en aspectos organizacionales, culturales, de desarrollo, de presupuesto y de alineación al negocio; de lo contrario no obtendrá beneficios de la solución SOA. La siguiente figura nos presenta el modelo de dominios de SOA propuesto en [Duré 2005]; este modelo describe tres dominios fundamentalmente técnicos (arquitectura, proyectos y aplicaciones) y tres dominios no técnicos (estrategia de negocio y procesos, costos y beneficios, organización y gobierno).

Figura 10: Modelo de dominios SOA. Tomada de [Duré 2005]

- **Arquitectura:** es necesario que la organización invierta tiempo y recursos en la implantación de una arquitectura de software de referencia que permita crear estándares en temas de: definición de procesos de negocio, desarrollo, implementación, administración, seguridad, integración y monitoreo de aplicaciones de software.

  Para esto es necesario desde el punto de vista de [Geeks 2007]:

  - *Comprender los objetivos de negocio*, es decir, qué información es necesaria, cómo se intercambia la información y qué se hace con dicha información.
  - *Definir el dominio que encierran el o los problemas*, esto es, analizar si es necesario que todas las aplicaciones adopten una arquitectura de referencia o si se puede migrarlas paulatinamente.
  - *Seleccionar la(s) tecnología(s) que se va(n) a utilizar*. Que probablemente será una mezcla de productos y elementos que satisfacen adecuadamente las necesidades en términos de SOA.

- **Bloques básicos**: definir los servicios básicos con los que va contar la organización. Esto con base en los procesos de negocio que se hacen recurrentemente en las diferentes unidades de negocio.

Para esto es necesario, según [Geeks 2007]:

- *Comprender todos los servicios disponibles en el dominio:* hay que responder a las preguntas: ¿Cuántos servicios tengo? ¿Dónde están estos servicios? ¿Cuál es el objetivo de cada servicio? ¿Qué información está vinculada con cada servicio? ¿Existen dependencias? ¿Qué problemas de seguridad hay? Por tanto, la idea es crear un directorio de servicios que tenga plasmada toda esta información [Gutiérrez 2005].
- *Comprender todas las fuentes de información disponibles en el dominio*: identificar aspectos relativos a: la ubicación de estas fuentes, la estructura de la información almacenada, restricciones de integridad, dependencias, problemas de seguridad, etc.
- *Comprender todos los procesos del dominio*: definir y enumerar todos los procesos de negocio que existen en el dominio del problema (automatizados o no). De esta forma, se podrán implementar mecanismos de alto nivel que faciliten el modelado de dichos procesos.
- *Identificar y clasificar todas las interfaces al exterior del dominio que puedan ser necesarias*: por ejemplo, es necesario tomar en cuenta comunicaciones con proveedores o posibles consumidores externos de los servicios de la organización.
- *Definir nuevos servicios y los límites de la información para esos servicios.*

- **Proyectos y aplicaciones**: este punto se refiere al cambio que debe darse en la organización con respecto de la administración de los proyectos y por ende de las funcionalidades que se encuentran implementadas, ya que muchas veces la descoordinación entre las distintas áreas involucradas en la creación y desarrollo de nuevos requerimientos provoca el re-trabajo y la duplicidad de funcionalidades en los sistemas. Para esto es necesario la definición de un gobierno SOA (ver la sección de Organización y Gobierno).

- **Estrategia de negocio y procesos**: este dominio consiste en lograr que las personas involucradas en el negocio y las involucradas en TI puedan comunicarse en un mismo lenguaje con el fin de satisfacer los requerimientos de negocio exitosamente.

  Los desarrollos deben ser impulsados por necesidades de negocio y de procesos más que centrarse en aplicaciones y sistemas. Pero no sólo los servicios desde TI deben ser implementados según las necesidades del negocio y de los procesos sino que - por otro lado - las líneas de negocio deben conocer qué servicios están disponibles para poder expresar sus necesidades en términos de dichos servicios, y definir procesos nuevos basados en procesos y servicios existentes.

  Esto se puede lograr con la combinación de SOA y BPM ya que este último soporta el modelado visual de procesos de negocio, lo cual permite el entendimiento conceptual de cómo el flujo de trabajo de estos procesos fluye entre los distintos servicios. Además, el mercado provee herramientas que automatizan la creación de estos procesos, al punto que analistas de negocios pueden participar de la creación de estas tareas e incluso les da la capacidad de incorporar cambios a procesos existentes.

- **Costos y beneficios**: este dominio se refiere a la necesidad de recolectar la información necesaria para justificar las inversiones realizadas en la creación y mantenimiento de servicios. El retorno de inversión de los servicios se puede medir basado en dos propiedades fundamentales: reutilización y agilidad para cambiar [Linthicum 2005]. El primero se refiere a utilizar un servicio en más de un proyecto o necesidad de negocio y el segundo nos indica si la arquitectura de TI se adaptó rápidamente a los cambios del negocio.

  Según [Linthicum 2005] para obtener el retorno a la inversión (ROI) de la reutilización se deben analizar factores como:
  - *El número de servicios reutilizables*: esto se refiere a la cantidad de servicios disponibles para reutilizar.
  - *La complejidad de los servicios*: número de funciones que componen el servicio. En ocasiones un servicio puede proveer diferentes comportamientos.
  - *El grado de reutilización de sistema a sistema*: se refiere a la cantidad de veces que se ha reutilizado un servicio en diferentes sistemas.

  En lo que respecta a la agilidad se podría evaluar:

  - *El grado de cambio durante el tiempo*: esto se refiere a la cantidad de veces que se han cambiado procesos de negocio en un periodo particular.
  - *Habilidad para adaptarse al cambio*: grado en que la arquitectura de TI contribuyó a mejorar la adaptación al cambio.
  - *Valor relativo del cambio*: se refiere a la cantidad de dinero generado por el cambio en el proceso.

Como se puede notar, a ninguno de los criterios se les ha dado un peso específico pues este varía de organización en organización; esto depende única y exclusivamente de la importancia y priorización que cada empresa le dé a sus procesos de negocio. También depende del nivel de madurez que la organización posea con respecto de SOA.

Con respecto del *nivel de madurez*, la organización también puede medir la inversión y los beneficios de una arquitectura SOA, según un modelo de madurez como el que presenta [Chavarría 2007]:

Figura 11: Modelo de Madurez SOA. Tomado de [Chavarría 2007]

Otro modelo de madurez interesante es el presentado por [Sonic 2005a], el cual proporciona objetivos y guías de cómo SOA puede ir generando poco a poco un impacto positivo en la empresa, donde el principal objetivo es alcanzar la optimización de los servicios de negocio, pasando a través de distintos niveles de madurez, como lo muestra la siguiente figura:

Figura 12: Modelo de Madurez SOA. Tomado de [Sonic 2005b]

El modelo de madurez completo se encuentra descrito en [Sonic 2005a] y en [Sonic 2005b], donde podrá encontrar una referencia rápida de cuáles son los beneficios que se obtienen de aplicar el modelo, cuál es el alcance de cada una de las etapas, cuáles son los factores críticos de cada nivel desde las perspectivas tecnológica, de recursos humanos y organizacional, así como los estándares recomendados en cada fase. Además, la referencia rápida determina los objetivos principales de cada etapa y las prácticas por seguir para alcanzarlos.

- **Organización y gobierno:** en un proceso de implementación SOA van a surgir preguntas como: ¿Qué servicios tenemos?, ¿Cómo se invocan los servicios?, ¿Quién garantiza la exactitud de los datos que retornan los servicios?, ¿Quién financia los servicios?, ¿Cuánto cuesta la operación de cada servicio?, ¿Qué aplicaciones utilizan los servicios?, etc.

  Estas interrogantes son las que hacen necesario que exista un gobierno de SOA, el cual es el encargado de definir qué es lo que hay que hacer, cómo hay que hacerlo y quién tiene la responsabilidad de realizarlo [Bastida 2008]. En este dominio surge la necesidad de la formación de un "Comité de SOA" que se responsabilice de:
  - Priorizar implementaciones.
  - Auditar el nivel de disponibilidad de los servicios.
  - Auditar el cumplimiento de los contratos, tanto por parte de los productores como por parte de los consumidores de los servicios.
  - Resolver cuestiones presupuestarias (qué área financia la implementación de un servicio que va a ser reutilizado por muchas otras áreas en producción, por ejemplo).
  - Asegurar e imponer la reutilización y evitar la duplicación de esfuerzos.
  - Determinar la exposición o no de servicios sensibles hacia el exterior de la organización.
  - Determinar procesos operacionales, herramientas y buenas prácticas.
  - Definir los mecanismos de versionamiento y control de cambios.
  - Asegurar la adhesión a la arquitectura de referencia.
  - Seguir métricas, costos y evaluar y reportar beneficios.
  - Implementar procesos de planeamiento y revisión periódicos.
  - Velar por el proceso de documentación y posterior mantenimiento de la arquitectura. Esta responsabilidad implicaría no solo "normar" el proceso de documentación, sino también auditarlo.

  En [Tibco 2008] se realiza una recomendación respecto de las personas que pueden integrar dicho comité:
  - Un gestor (director) de proyecto que disponga de una visión amplia de todo el proceso de negocio y gestione todo lo relacionado con su creación o modificación.
  - Un arquitecto de procesos de negocio que determine los cambios de proceso de negocio necesarios para alcanzar los beneficios esperados.
  - Un arquitecto de sistemas que establezca la tecnología y servicios necesarios para respaldar el proceso de negocio y sus objetivos.
  - Un responsable de TI que se ocupe de garantizar la cooperación activa de todos los responsables dentro del grupo de TI.
  - Un responsable de negocio que garantice la cooperación de todas las unidades de negocio afectadas por el proceso.

Un aporte importante a la introducción de SOA en el negocio es conocer cómo otras organizaciones lo han hecho, cuáles han sido sus experiencias y los resultados obtenidos. En [SOA Consortium 2008] se puede encontrar un conjunto de casos de estudio que resumen la experiencia de organizaciones que trabajan en campos como: servicios financieros, manufactura, servicios farmacéuticos, telecomunicaciones, ventas, combustibles, transporte y seguridad, entre otros. Los casos de estudio evalúan tres aspectos generales: cuál fue la necesidad de negocio para adoptar SOA, cuáles fueron las razones en la organización que llevaron a la adopción de SOA y cuáles fueron los beneficios entregados por la arquitectura orientada a servicios. Los casos están divididos en cuatro categorías:

1. *Pruebas de concepto*: estos proyectos cuentan con mínimo patrocinio de negocio y el nivel complejidad de SOA es bajo.
2. *Guiados por TI*: estos cuentan con mínimo patrocinio de negocio y el nivel de complejidad de la arquitectura orientada a servicios es alta.
3. *Guiados por negocio*: proyecto que cuentan con el patrocinio de negocio y el nivel de complejidad de SOA es alto.
4. *Los denominados "mega":* proyectos que cuentan con el patrocinio de negocio y el nivel de complejidad de SOA es alto. Además, los cambios que se persiguen con este tipo de proyectos son fuertes a nivel organizacional.

6. Conclusiones y recomendaciones

- Antes de iniciar con la implementación de SOA se deben conocer cuáles son las necesidades de la empresa, ya que cada organización posee su propio ambiente. Además, SOA debería estar ligado completamente con la estrategia de la empresa y cada empresa define su estrategia con respecto de su ámbito de negocios.
- La adopción de una Arquitectura Orientada a Servicios requiere de un cambio cultural importante. La organización debe dejar de pensar en estrategias de negocio y TI por separado y cambiar por un modelo colaborativo, especialmente en temas estratégicos. Para esto, la comunicación negocio-TI debe ser ágil y efectiva, con lo que evitará la entrega de servicios poco útiles para los procesos de negocio. El desarrollo de una Arquitectura Empresarial favorece la armonización de visiones y la consistencia de las estrategias [Porras 2008].
- El motivador principal para la creación de servicios en SOA deben ser las funcionalidades que el negocio requiere u ofrece. Debe evitarse proceder hacia SOA sin tener en mente un resultado relevante para el negocio [Michelson 2009].
- La participación de las áreas de negocio en la implementación de una arquitectura orientada a servicios es fundamental, porque es del negocio que se obtienen los insumos para el desarrollo de cada uno de los servicios que se van exponer. Además, es el negocio el que conoce los procesos y las necesidades de la organización.
- Para la implementación de SOA no se puede dar una receta universal. La implementación depende del nivel de madurez en que se encuentre la organización con respecto de SOA. Además, cada organización tiene su propia cultura.
- El nivel de madurez de la empresa con respecto de SOA va a definir el tipo de inversión que SOA significa para la organización en los plazos corto, mediano o largo.
- Si no existe reutilización, SOA proporciona pocos beneficios para la organización que la implemente y esto afectará directamente la estandarización de los procesos de negocio. La mayoría de los beneficios no se van alcanzar con solo implementar SOA: es importante hacer uso de los principios de arquitectura de software y reutilización previo a la adopción de SOA.
- SOA no es: un producto, una solución, una tecnología o un "Web service", sino es un medio para habilitar los procesos y reaccionar ante las oportunidades que se presentan al negocio.
- Hay una complementariedad y sinergia natural entre SOA y BPM (Business Process Management). Vale la pena considerar la adopción de SOA en conjunto con una orientación hacia procesos de negocio apoyados por tecnología, y usar estándares para definir las arquitecturas y tales procesos de negocios.
- Es importante establecer, posterior a la implementación de un plan piloto, lo que se denomina "Gobierno SOA", con el fin de mantener el control de la implementación de la arquitectura de referencia. Hacerlo en etapas tempranas podría confundir la asimilación de los conceptos.
- Para la implementación de SOA es importante aplicar estándares de comunicación que ayuden con la publicación y descubrimiento de los servicios; un posible inicio es basarse en: XML, SOAP y WSDL.

7. Referencias

[ATA 1996] Army Digitization Master Plan '96. Tomado de: http://www.globalsecurity.org/military/library/report/1996/army_digit_m-plan96-04arch.htm. Visitado: 23-08-2008.

[Barco 2006a] Barco A. Principios de la Orientación a Servicios. Blog Arquitectura Orientada a Servicios. 2006. Tomado de: http://arquitecturaorientadaaservicios.blogspot.com/2006/06/principios-de-la-orientacin-servicios.html

[Barco 2006b] Barco A. Modelado de servicios. Blog Arquitectura Orientada a Servicios. 2006. Tomado de: http://arquitecturaorientadaaservicios.blogspot.com/2006/06/principios-de-la-orientacin-servicios.html. Visitado: 24-11-2007.

[Basili 1988] Basili V., Rombach H. Towards a comprehensive framework for reuse: A reuse-enabling software evolution environment. Technical Report CS-TR-2158, University of Maryland, 1988.

[Bass 2003] Bass L, Clements P , Kazman R; Software Architecture in Practice (2nd. edition), Addison-Wesley, 2003.

[Bastida 2008] Bastida L. Gobierno SOA: Elemento Clave en la Integración de Negocio y Tecnología. European Software Institute. Vizcaya. España, 2008.

[Braun 1994] Braun C. L. Reuse. En Marciniak (ed.) Encyclopedia of Software Engineering, pp. 1055-1069. John Wiley and Sons, 1994.

[Calderón 2006] Calderón, Alan. Patrones de software. Club de Investigación Tecnológica, 2006.

[Canto 2006] Canto M., Pareda D., Segurola A. Service Oriented Architecture (SOA). Facultad de Ingeniería, Universidad de la República. Uruguay, 2006.

[Channabasavaiah 2003] Channabasavaiah K., Kerry H., Edward T. Migrating to a service oriented architecture, 2003. Tomado de: http://www-128.ibm.com/developerworks /webservices/library/wsmigratesoa2/index.html. Visitado: 02-02-2008

[Chavarría 2007] Chavarria L., Gutiérrez J. Modelo de Madurez Arquitectura Orientada a Servicios. GATI.2007.

[Cooper 1994] Cooper J. Reuse-the business implications. En Marciniak (ed.) Encyclopedia of Software Engineering, pp. 1071-1077. John Wiley & Sons, 1994.

[Cuesta 2002] Cuesta C. Arquitectura de software dinámica basada en reflexión. Universidad de Valladolid. España,. 2002.

[Duré 2005] Duré O. SOA: Un Modelo de Dominios que Ataca Todos los Aspectos de una Organización. http://www.willydev.net/descargas/SOA/WillyDev_Nota_Revista_Code.pdf. Visitado: 24-11-2007

[Erl 2007] Erl T. SOA Principles of Service Design. Prentice Hall, 2007.

[Evdemon 2005] Evdemon J. Principios de diseño de servicios: patrones y Antipatrones de Servicios. Microsoft Corporation, Architecture Strategy, 2005. Tomado de: http://www.microsoft.com/spanish/msdn/articulos/archivo/121205/voices/SOADesign.mspx

[Favaro 1998] Favaro J., Favaro K., Favaro P. Value-Based Software Reuse Investment. Annals of Software Engineering 5, 1998.

[García 2007] García F., López F. SOA Desarrollos prácticos y bases tecnológicas, 2007.

[Garlan 2000] Garlan D. Software Architecture: a Roadmap. International Conference on Software Engineering, 2000.

[Geeks 2007] Service Oriented Architecture (SOA): ¿Por dónde empezar?. Tomado de: http://geeks.ms/blogs/ciin/archive/2007/10/15/service-oriented-architecture-soa-191-por-d-243-nde-empezar.aspx Visitado: 08-02-2008

[Gómez 2009] Gómez D., El papel del ESB en una solución SOA.2009. Tomado de: http://www.dosideas.com/java/498-el-papel-del-esb-en-una-solucion-soa.html. Visitado: 22-11-2009.

[Gutiérrez 2005] Gutiérrez I., Otón S. Service Oriented Architecture. Departamento de Ciencias de la Computación. Universidad de Alcalá. España, 2005. Tomado de: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-132/paper09.pdf. Visitado: 24-11-2007.

[IBM 2008] Published Software Architecture Definitions. Tomado de: http://www.sei.cmu.edu/architecture/published_definitions.html#Modern. Visitado: 19-01-2008.

[Infobae Profesional 2008] Cuáles son los errores que comenten las empresas al implementar SOA, 2008. Tomado de: http://www.infobaeprofesional.com/notas/65038-Cuales-son-los-errores-que-cometen-las-empresas-al-implementar-SOA.html?cookie Visitado: 31-05-2009

[Lewis 2007] Lewis G.A., Morries E., Simanta S.,Wrage L. Common Misconceptions about Service-Oriented Architecture. Commercial-off-the-Shelf (COTS)-Based Software Systems, 2007. ICCBSS '07. Sixth International IEEE Conference, 2007.

[Linthicum 2005] Linthicum D. The ROI of your SOA. Tomado de: http://www.ebizq.net/topics/soa/features/6092.html?page=2. Visitado: 07-06-2009.

[Linthicum 2006] Linthicum D. 8 Things Most People Misunderstand About SOAs. Tomado de: http://www.soainstitute.org/articles/article/article/8-things-most-people-misunderstand-about-soas/news-browse/6.html. Visitado

[Lizano 2000] Lizano F. Arquitectura de Software: La impostergable disciplina, 2002. Tomado de: lizano.info/Files/RP/ensayo002-2002.pdf. Visitado 19-01-2008.

[Luna 2000] Luna, A.; Trejos, I. El modelo de objetos: Lenguaje de modelaje Unificado (UML). Club de Investigación Tecnológica, 2000.

[Margolis 2007] Margolis B. SOA for the bussiness developer. 1st edition.MC Press, 2007.

[Martin 2003] Martin R.; Newkirk W.; Koss R. Agile Software Development Principles, Patterns and Practices. Prentice Hall, 2003.

[Michelson 2009] Michelson B. What are real people doing with SOA?. SOA Consortium.2009. Tomado de: http://blog.soa-consortium.org/soa_consortium_insights/2009/10/what-are-real-people-doing-with-soa.html. Visitado 22-11-2009.

[Microsoft 2006] Microsoft Corporation. La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real, 2006.

[Mollineda 2005] Mollineda R. Arquitectura del Software: arte y oficio. Actualidad TIC. Universidad Politécnica de Valencia, 2005. Tomada de: http://web.iti.upv.es/actualidadtic/2005/02/2005-02-arquitectura.pdf. Visitado: 19-01-2008

[Novegil 2008] Novegil A., Lama M., Sánchez E. Descripción del entorno de ejecución de diseños de aprendizaje enriquecido con técnicas de razonamiento basado en ontologías. Proyecto SUMA elearning multimodal y adaptativo, 2008.

[OMG 2007] SOA Consortium.SOA Executive Insight Report. 2007. Tomado de: http://www.soa-consortium.org/Executive_Insight_whitepaper-2007.pdf. Visitado: 22-11-2009.

[OMG-CORBA 2009] Object Management Group. CORBA® Basics. http://www.omg.org/gettingstarted/corbafaq.htm Visitado: 25-11-2009.

[OMG-MDMI 2009] Object Management Group / Model Driven Message Interoperability Consortium. Model Driven Message Interoperability literature. http://www.mdmi-consortium.org/literature.htm Visitado 25-11-2009.

[Otero 2007] Otero A. Arquitecturas Empresariales. Orientación a Servicios (SOA) y Gestión de Procesos de Negocio (BPM). Everis. Universidad Politécnica de Catalunya, 2007.

[Plummer 2005] Plummer D. Defining 'Service' is Key to Implementing a Service Oriented Architecture. Gartner Research, 2005.

[Porras 2008] Porras, G. Arquitectura Empresarial. Club de Investigación Tecnológica, 2008.

[Recena 2007] Recena M. SOA una perspectiva. Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla. España. 2007. Tomado de: www.slideshare.net/recena/soa-una-perspectiva. Visitado: 24-11-2007.

[Rotem 2007] Rotem A. What is SOA anyway?. Tomado de: http://www.rgoarchitects.com/files/SOADefined.pdf . Visitado: 23-05-2009

[Sametinger 1997] Sametinger J. Software Engineering with Reusable Components. Springer, 1997.

[Sasso 1992] Sasso, R. (Editor). Nuevas Tecnologías de Información. Club de Investigación Tecnológica, 1992.

[SOA Consortium 2008] SOA Industry Case Studies. SOA Consortium. 2008. Tomado de: http://www.soa-consortium.org/case-study.htm. Visitado 22-11-2009

[Sonic 2005a] Sonic Software Corporation. A new SOA Maturity Model. 2005. Tomado de: http://soa.omg.org/Uploaded Docs/SOA/SOA_Maturity.pdf. Visitado: 22-11-2009.

[Sonic 2005b] Sonic Software Corporation. A new SOA Maturity Model Quick Reference. 2005. Tomado de: http://soa.omg.org/Uploaded%20Docs/SOA/soamm_quick_reference.pdf. Visitado: 22-11-2009.

[Tam 2006] Tam J. ¿Cómo hacer mi modelo de negocios más eficiente? SOA da la solución. Top Management Alta Gerencia en la red, 2006. Tomado de: http://www.topmanagement.com.mx/modules.php?name=Noticias&file=show&clave=52757 Visitado: 13-02-2008

[Tibco 2008] Las siete claves del éxito SOA. Tomado de www.tibco.com/international/spain/resources/es_soa_together_adv.pdf. Visitado: 24-11-2007

[Tolosa 2009] Tolasa R. SOA Tutorial. Tomado de: http://www.cicomra.org.ar/cicomra2/asp/Present.%20R.%20Tolosa%20-%20CUBECORP%20-%20SOA.pdf Visitado: 31-05-2009

[Torralba 2004] Torralba J. Reutilización del conocimiento del diseño de software. Consideración en la determinación del precio de oferta al cliente. VIII Congreso de Ingeniería de Organización, 2004.

[Trejos 1999] Trejos, I.; Luna, A. El modelo de objetos: Análisis y Diseño. Club de Investigación Tecnológica, 1999.

[Wikipedia 2008] Definición de Servicios. Tomado de: http://es.wikipedia.org/wiki/Servicio. Visitado: 02-02-2008.