# ESTADO DEL ARTE DE SOA, PRINCIPIOS, TÉCNICAS, TECNOLOGÍAS, PATRONES Y METODOLOGÍAS

**KAREN ELENA ALVAREZ BALLESTAS**
**LUIS ALFONSO RODELO CALVO**

**UNIVERSIDAD TECNOLOGICA DE BOLIVAR**
**FACULTAD DE CIENCIAS DE INGENIERIA**
**PROGRAMA DE INGENIERIA DE SISTEMAS**
**CARTAGENA D. T. y C.**
**2008**

# ESTADO DEL ARTE DE SOA, PRINCIPIOS, TÉCNICAS, TECNOLOGÍAS, PATRONES Y METODOLOGÍAS

**KAREN ELENA ALVAREZ BALLESTAS**
**LUIS ALFONSO RODELO CALVO**

**Monografía para optar al título de Ingeniero de Sistemas**

**Director del Programa**
**Giovanny Vásquez**

**UNIVERSIDAD TECNOLOGICA DE BOLIVAR**
**FACULTAD DE CIENCIAS DE INGENIERIA**
**PROGRAMA DE INGENIERIA DE SISTEMAS**
**CARTAGENA D. T. y C.**
**2008**

# TABLA DE CONTENIDO

Pág.

# TABLA DE FIGURAS

Pág.

# INTRODUCCIÓN

SOA representa la integración de varias tecnologías que aunque diferentes presentan un ambiente de integración sin exclusiones. La arquitectura transciende el concepto de una tecnología en particular, Web Services por ejemplo, y es independiente de ellas. Una arquitectura como SOA esta en capacidad de involucrar diferentes tecnologías y representa mejor la integración de las mismas.

SOA define las funciones como servicios independientes, que poseen interfaces bien definidas. Los componentes de está definición son:

- Todas las funciones son definidas como servicios. Estas están compuestas por tres tipos de funciones que son las funciones puramente de negocios (Ej. Crear una Orden), las transacciones de negocios compuestas de funciones de más bajo nivel (Ej. Obtener historia crediticia) y las funciones de servicios del sistema (Ej. Validar identificación).
- Todos los servicios son independientes. Lo único que interesa entre componentes es que cada servicio entregue los datos que necesita otro para poder realizar operaciones.
- Las interfaces son invocables, dejando a un lado cualquier diferencia entre protocolos de comunicación, arquitectura, etc. Lo importantes es que respondan a los llamados de peticiones de servicios.

Dentro de SOA, la clave es la interfase, y es foco de la aplicación que la llama. Ella define los parámetros requeridos y la naturaleza del resultado. Esto significa que ella define la naturaleza del servicio, no la tecnología utilizada para implementarlo. El sistema debe efectuar y administrar la invocación del servicio, no lo hace la aplicación que llama. Esta función permite hacer realidad dos características críticas: que los servicios sean realmente independientes y segundo que puedan ser administrados. Esta administración incluye:

- Seguridad, para autorizar solicitudes, criptografía y validaciones de la información.
- Instalación y configuración, para permitir desplazar el servicio en la red optimizando el desempaño o eliminando redundancias para una disponibilidad óptima.
- Registro, para auditoria y mediciones.
- Enrutamiento dinámico, para recuperación de fallas y balanceo.
- Mantenimiento, para administrar versiones del servicio.

# OBJETIVOS

## OBJETIVO GENERAL

Desarrollar un documento en el que se profundice en el estado del Arte de aplicaciones SOA con el fin de crear una idea sólida de las técnicas, metodologías, tecnología y patrones.

## OBJETIVOS ESPECIFICOS

- Conocer de donde provienen las aplicaciones Orientadas a Servicios.
- Identificar los servicios como piezas fundamentales en el desarrollo de aplicaciones SOA
- Analizar las metodologías, procesos, elementos, patrones, técnicas y tecnologías existentes para el desarrollo de aplicaciones SOA
- Extraer las principales características de cada una de las metodologías SOA para establecer diferencias.
- Identificar los beneficios que ofrecen las Aplicaciones Orientadas a Servicios al ser implementadas en una empresa

# 1. MARCO TEORICO

## 1.1. ANTECEDENTES

Cuando surge un nuevo concepto, por lo regular tiene sus antecedentes, antecedentes que permiten entender porqué se originó, que necesidades conllevaron a que se diera, y quizás conocer un poco hacia donde se dirige. En nuestro caso las Arquitecturas Orientadas a Servicios tienen su origen en los siguientes conceptos:

- Sistemas Distribuidos
- Servicios Web

Para entender como estos conceptos las originaron, se describen a continuación:

### *1.1.1. Sistemas Distribuidos*

La computación desde sus inicios ha sufrido muchos cambios en especial en lo que hace referencia a los Sistemas tecnológicos y de información, esto debido a que siempre se busca ofrecer un trabajo eficiente y eficaz, garantizando la confiabilidad, disponibilidad, integridad y seguridad de la información que transportan o intercambian.

Uno de estos avances son los Sistemas Distribuidos, quienes dejan atrás la Computación centralizada para empezar en la década de los 80 con una

tecnología posibilitadora que permitía el intercambio entre el sistema operativo y la aplicación distribuida, a dicha tecnología se denomino DCE (Distributed Computing Environment), la cual evoluciono a CORBA (Common Object Request Broker Architecture).

CORBA presentó una tecnología mucho más distribuida, ya que implementó el concepto de bus lógico, el cual permite la conexión de todos los elementos que hacen parte del sistema distribuido, además incluyó aspectos tales como los de ciclo de vida (con operaciones para la creación, desplazamiento o borrado de componentes en el bus), el servicio de eventos (para registrar o eliminar el interés de determinados tipos de eventos), el servicio de transacciones (para posibilitar la coordinación de los commits en dos fases), etc.

Pero lo que debemos tener en cuenta de esta arquitectura es la oferta de distintos tipos de servicios que permiten garantizar una serie de funciones que no requieren ser contempladas en cada una de las aplicaciones implementadas.

Es una arquitectura en la que cada una de las aplicaciones se basan en marcos (horizontales y verticales) que a su vez se asientan en los servicios CORBA y éstos a su vez en un bus común de objetos el cual puede representarse de la siguiente manera.

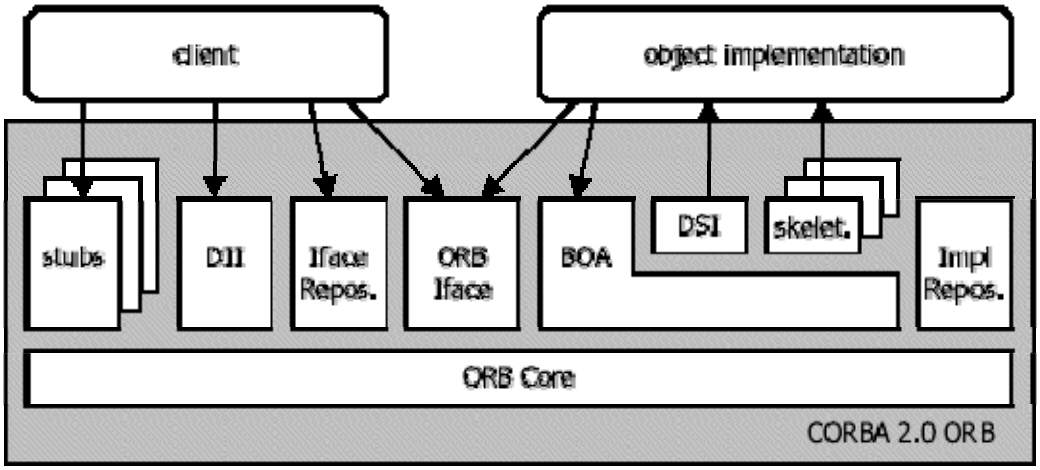


*Figura 1. La arquitectura CORBA*

Tomado de **http://www.geocities.com/hsagastegui2002/TrabajoCorba1.html**

### *1.1.2. Servicios Web*

Aun sin introducirnos por completo en el concepto de SOA, muchas personas siempre piensan que una aplicación SOA está compuesta de servicios WEB, pues esto es algo errado, es claro definir que los Servicios Web si fueron necesarios para que surgieran las aplicaciones Orientadas a Servicios por las siguientes razones:

- Por completa autonomía que presentan los servicios, es decir cada servicio es responsable de su propio dominio, lo que se traduce normalmente en limitar su alcance a una función específica de negocio (o un grupo de funciones relacionadas).

- Dejan a un lado los componentes utilizados en los sistemas distribuidos, para implementar Estándares de comunicación, de interpretación de texto, como lo son: la comunicación vía protocolos Internet (más frecuentemente HTTP2) y el envío y recepción de documentos formateados como XML.

- Por último los servicios Web permitieron que un mismo servicio fuera capaz de ser el peticionario y suministrador de servicios, lo cual proporciona que los servicios puedan solicitar información entre sí. Haciendo uso además de un agente descubridor el cual permite que dichos servicios sean localizados.

### *1.1.3. Arquitectura Orientada a Servicios (SOA)*

SOA es la combinación de los dos conceptos descritos anteriormente y mucho más, SOA es un paradigma, cuyo objetivo fundamental es la utilización de servicios para satisfacer los requerimientos del cliente a través de un protocolo común de comunicaciones, donde la aplicación de servicios Web es clave fundamental para construir una aplicación de este tipo, dejando claro que cada uno de estos servicios Web XML deben cumplir o garantizar que presentan bajo acoplamiento para brindar la posibilidad de la reusabilidad (Ver capitulo 6).

Soportando este modelo, SOA promueve el uso de mecanismos de descubrimiento de los servicios a través de un broker de servicio o agente descubridor.

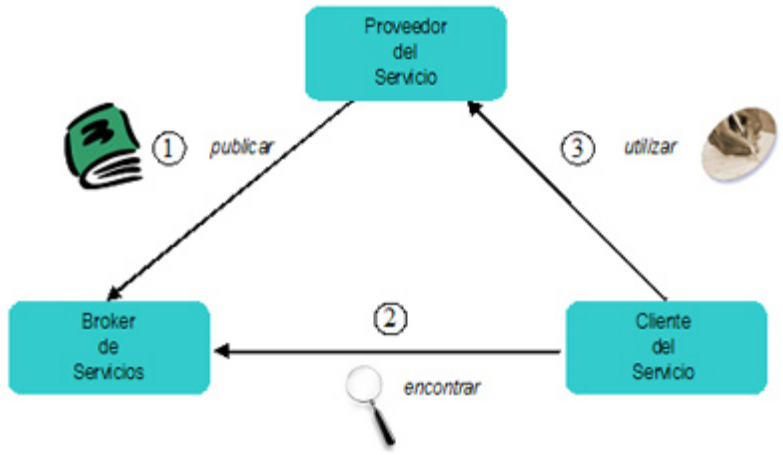


***Figura 2. Relación entre Peticionario, Proveedor y Broker de servicio***

Tomado de **http://www.grupoassa.com/Espanol/gif/gr%E1ficowebservices1.jpg**

Por supuesto, antes de que un cliente o peticionario- pueda solicitar un servicio, necesita encontrar el proveedor (el cual ha publicado previamente el servicio). Este servicio de ubicar un posible proveedor es realizado por el broker de servicio, el cual opera típicamente con un repositorio. Cuando se le solicita, el broker de servicio devuelve un documento que permite al cliente, primero, localizar y luego unirse con el proveedor.

Un cliente puede requerir múltiples servicios, cada uno con un proveedor diferente. El hecho de registrar los servicios en un registro central "repositorio" en el que los clientes pueden buscar, les brinda la flexibilidad necesaria para realizar búsquedas basadas en conjuntos de criterios que cambian dinámicamente y, de esta forma, no estar ligados estáticamente al proveedor. Sin dicho registro, el cliente debería estar obligado a codificar la ubicación del proveedor del servicio y complicar así el mantenimiento.

## 1.2. DEFINICIONES DE SOA

Son importantes los conceptos que diferentes investigadores, profesionales IT, compañías, empresas, etc. han ofrecido sobre SOA, ya que permite tener una visión más amplia de lo que esto significa. A continuación algunos conceptos de los más importantes:

- SOA son un conjunto de servicios tanto de negocio como tecnológicos que interactuando entre ellos, proporcionan la lógica necesaria para construir aplicaciones de una manera rápida y cumpliendo siempre con los principios de la Orientación a Servicios (Antonio Barco – Blog Arquitectura Orientada a Servicios)

- W3C: “Conjunto de componentes que pueden ser invocados, cuyas descripciones de interfaces se pueden publicar y descubrir”

- CBDI: “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”

- “Infraestructura de alto nivel basada en las mejores practicas y patrones para crear soluciones basadas en servicios, de alta cohesión y bajo acoplamiento” (Geniant®).

- “Estilo arquitectónico apto para implementar bajo acoplamiento entre agentes. Los agentes son proveedores y consumidores de servicios, que son la unidad de trabajo”. (Hao He).

- “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” (IBM).

- Una aplicación SOA es una colección de servicios, donde servicios es la unidad atómica de una SOA, los servicios encapsulan procesos de negocios, los proveedores de servicios se registran solos, un servicio involucra: Find, Bind, Execute, las instancias más conocidas son los web services (MITRE).

- “SOA es una arquitectura de software que comienza con una definición de interfase y construye toda la topología de la aplicación como una topología de interfaces, implementaciones y llamados a interfaces. 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” (Gartner).

Se puede continuar definiendo a SOA y no se termina, ya que cada investigador, cada empresa, han definido SOA de acuerdo a sus necesidades y a como lo han implementado en la empresa y en su vida, sin embargo queda claro, en cada una de las definiciones anteriormente descritas, que la unidad funcional de SOA son los servicios, que estos se implementan a través de interfaces, que la relación existente entre estos se caracteriza por presentar una alta cohesión y un bajo acoplamiento, donde los consumidores de servicios al interactuar en este proceso forman una unidad completa de negocio.

# 2. PRINCIPIOS

## 2.1. GARTNER Y SU IMPORTANCIA EN SOA

Gartner[1] como proveedor líder de investigación y análisis de la industria global de tecnologías de información, es quien, por primera vez describe la arquitectura orientada a servicios en 1996, el cual apoyado por el concepto de Servicios Web aumento el interés en la misma. Además estás dos tendencias se potencian mutuamente: el interés por los Servicios Web lleva hacia SOA, y las ventajas de la arquitectura SOA ayudan a que las iniciativas de Servicios Web tengan éxito.

En 2003, SOA entra al fin por completo en el mundo de las IT empresariales, a través de los Servicios Web:

- Al contrario que CORBA y DCE, los estándares de servicios Web no tienen detractores entre los fabricantes.

- La flexibilidad de los Servicios Web para soportar aplicaciones multicanal.

- La capacidad de SOAP de pasar por los firewalls, aprovechando la ubicuidad del http.

[1] http://www.gartner.com/2_events/conferences/asset_145451_2.jsp

- El soporte de Servicios Web en servidores de aplicaciones que albergan lógica empresarial.

- Los ESBs (Definido más adelante), que combinan Servicios Web con middleware orientado a mensajes (MOM), más algunas capacidades de transformación y enrutado.

## 2.2. DEFINICIÓN DE SERVICIOS

Es un componente de software que puede ser invocado remotamente y que puede ser descrito de una manera estándar a través de un archivo WSDL (Web Services Definition Lenguaje)(Para más detalle Ver capitulo 5.1). Sin embargo, la definición descrita no es la correcta, ya que podemos cometer graves errores, esto debido a que al momento de identificar los servicios, se definirán servicios que no son para nada útiles en el desarrollo de la aplicación. Por ejemplo podríamos crear un servicio que calculara la raíz cuadrada. ¿Pero realmente, es una buena idea?, la respuesta es definitivamente, no. ¿Por qué? La razón es sencilla, el tiempo que tardaríamos en invocar este método sería mucho más largo que el tiempo que tardaría en ejecutarse la función. Por lo tanto la función raíz cuadrada nunca dejará de ser una función del lenguaje de programación para convertirse en un servicio. Tomemos una función más compleja, una que dependiendo de un nivel de severidad decide imprimir un error a un log o no. ¿Es esto un servicio? En este caso, si la red es rápida y el disco no tanto, es posible que el tiempo de invocación de la función sea más bajo que el de ejecución. Sin embargo, eso no lo convierte en un servicio. ¿Por qué? Porque no es una función de negocios. El objetivo de implementar SOA es responder a las necesidades del negocio, ya sea a través de EAI (Enterprise Application Integration), (Para más detalle Ver capitulo 3) o a través de BPM (Para más detalle Ver capitulo 3). Los usuarios de negocios no están

interesados en funciones de IT como guardar mensajes de error en un log. Este tipo de funciones, propias de las áreas de sistemas se siguen resolviendo mejor a través de librerías compartidas utilizadas por las aplicaciones. Por lo tanto, vemos que la adopción de SOA no significa que de repente vamos a ver una proliferación incontrolada de servicios ya que en la mayoría de los casos, las empresas no tienen miles de funciones de negocio.

## 2.3. CLASIFICACIÓN DE LOS SERVICIOS

El foco de SOA está en la infraestructura funcional y sus servicios del negocio, no en la infraestructura técnica y sus servicios técnicos. La entidad básica es la función o proceso del negocio.

Los servicios se clasifican en distintos tipos con determinadas características que tienen distinto significado desde el punto de vista del diseño implementación y gestión del proyecto. Estas características son significativamente distintas en cuanto a reusabilidad, mantenibilidad, escalabilidad y performance de los servicios.

Los servicios en SOA se clasifican de la siguiente forma:

**SERVICIOS BÀSICOS**: Son la base de SOA, son servidores puros y no mantienen el estado conversacional de la sesión. Se dividen en servicios centrados en los datos y centrados en la lógica.

- Servicios Centrados en los datos: Su propósito es manejar los datos persistentes, de almacenamiento y recuperación del negocio.

- Servicios Centrados en la lógica: Estos servicios proveen encapsulamiento para cálculos complejos o reglas del Negocio, tradicionalmente encapsulados en bibliotecas y frameworks del negocio

**SERVICIOS INTERMEDIARIOS:** Son servicios sin estado que hacen de puente entre las inconsistencias técnicas o discrepancias conceptuales en el diseño. Son tanto clientes como servidores en SOA, mediando entre los distintos elementos que deben funcionar juntos. Se dividen a su vez en technology gateways, adapters, facades y funcionality-adding.

- **Technology gateways:** Estos servicios hacen de puente entre discrepancias tecnológicas, incorporando dos o más tecnologías para comunicación o codificación de datos. Hacen de proxys para sus servicios de Negocio y representan la funcionalidad de los servicios que están por debajo en un ambiente que es distinto tecnológicamente que el ambiente de ejecución del servicio de Negocio original.

- **Adapters:** Es un tipo especial de servicio intermediario que mapea las firmas y formatos de mensaje de un servicio a los requerimientos de un cliente. Por ejemplo, si se tiene dos servicios para el mismo concepto de distintas compañías, con un adapter para cada uno en cada sentido de los pedidos, se mapean los requerimientos de la aplicación que usa un servicio en el otro y viceversa.

- **Facades:** Su propósito es proveer una vista distinta (y posiblemente agregada) de uno o más servicios existentes, por lo que también pueden actuar como technology gateways y/o adapters. Como el patrón de Gamma, una facade puede utilizarse para proveer una vista específica de un conjunto de servicios que se encuentran por debajo, que en general son servicios básicos.

- **Funtionality - adding:** Este tipo de servicio se usa cuando se quiere agregar funcionalidad a un servicio existente sin cambiar el servicio mismo. En este caso se crea un servicio que provea la misma funcionalidad del servicio original y agregue las nuevas características requeridas.

**SERVICIOS CENTRADOS EN PROCESOS:** Estos servicios encapsulan el conocimiento de los procesos del Negocio de la Organización, controlan y mantienen el estado del proceso en ejecución. Desde el punto de vista técnico son la clase de servicios más sofisticada. Actúan tanto de clientes como de servidores en una SOA, manteniendo el estado del proceso para sus clientes. Una de las principales ventajas que presentan estos servicios es que separan la lógica de procesos, asignando la lógica del Negocio núcleo en los servicios básicos, el control de diálogo en la aplicación de frontend y la lógica de los procesos en estos servicios centrados en procesos. Estos son mayormente específicos del proyecto definiendo los procesos del Negocio y su control, en base a orquestación de los servicios existentes.

**SERVICIOS EMPRESARIALES PÙBLICOS:** Los tipos de servicio anteriores son solamente para uso dentro de los límites de una empresa en particular. Los servicios empresariales públicos son servicios que una empresa que ofrece a socios y clientes para su consumo. Por ejemplo, proveer servicios para que los clientes hagan seguimiento de sus envíos en la empresa que los hace, proveer funcionalidad para enviar mensajes SMS de forma que las empresas puedan incorporarla a sus sistemas propios. Estos servicios tienen requerimientos específicos de interface, desacoplamiento, seguridad y facturación de uso, ya que las empresas deben acordar claramente como se realiza el uso de los mismos.

## 2.4. MODELADO DE SERVICIOS

- **Servicios controladores**: Este tipo de servicios representan los procesos de negocios que se quieren implementar en una determinada secuencia para obtener un objetivo, ya que son los encargados de recibir las peticiones de los clientes y realizar las llamadas necesarias a otros servicios (en la secuencia adecuada) para devolver una respuesta. Es decir, son los servicios encargados de coordinar al resto de servicios.

- **Servicios de negocio**: Este tipo de servicios son los que simbolizan o constituyen una tarea de negocio, la cual forma parte de un proceso de negocio, por lo regular estos servicios suelen ser poco reutilizables ya que están orientados a resolver una tarea muy puntual.

- **Servicios de utilidad**: Este tipo de servicios, como característica principal tienen la propiedad de ser altamente reutilizables ya que representan tareas de negocios que pueden ser implementadas constantemente en diferentes etapas, procedimientos y contextos del sistema. 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 y por tanto altamente reutilizables (Ej.: servicio de acceso a bases de datos relacionales).

Con lo cual, una aplicación SOA la podemos dividir en tres capas. La capa de recepción de peticiones (servicios controladores), la capa de tareas (servicios de negocio) la capa de lógica reutilizables (servicios de utilidad).


***Figura 3. Modelado de Servicios***

Tomado de http://arquitecturaorientadaaservicios.blogspot.com/2006/08/modelado-de-servicios.html

## 2.5. MODELO CONCEPTUAL DE SOA

El modelo conceptual de SOA se basa en un estilo arquitectónico donde interaccionan tres principales partes:

- El proveedor de servicios, que publica una descripción del servicio y proporciona la aplicación para el servicio.

- Un servicio al consumidor, que puede utilizar el identificador de recursos uniforme de la descripción de servicio directamente o puede encontrar la descripción de servicio de un servicio de registro y obligar a invocar y el servicio.

- El servicio intermediario proporciona y mantiene el servicio de registro.

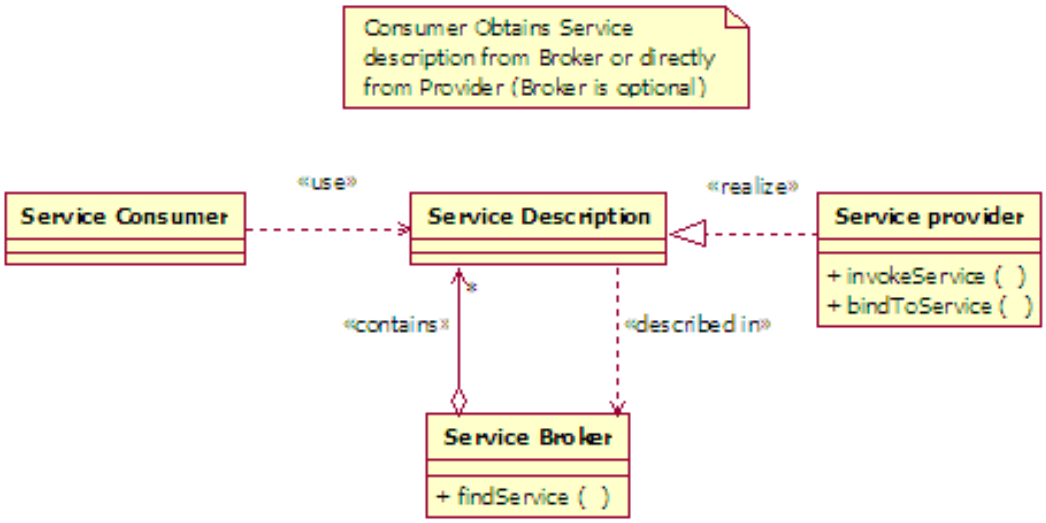


***Figura 4. Modelo Conceptual***

Tomado de http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/

## 2.6. EL ESTILO ARQUITECTONICO Y PRINCIPIOS

SOA la podemos describir como una empresa, donde los recursos de TI se vinculan para suplir la demanda. En SOA, es indispensable el rol de los participantes ya que ellos determinan la línea de negocio de la empresa, la cual puede presentarse dentro de una misma empresa o de varias empresas. Se compone de un conjunto de las empresas de servicios de TI alineados que, en conjunto, cumplen con los procesos de negocio y objetivos. Los servicios en las aplicaciones compuestas se invocan a través de protocolos estándar, como se muestra en la figura de abajo.

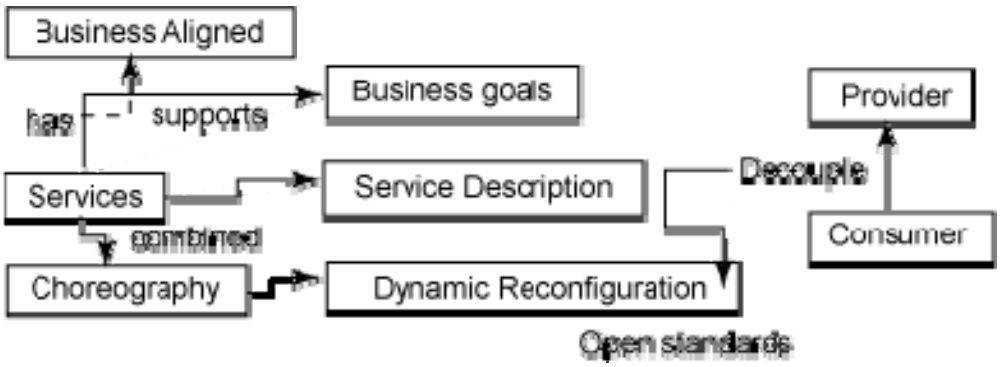


***Figura 5. Atributos de SOA***

Tomado de http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/

Un servicio es un software de recursos (detectable) con una descripción de servicios externalizados. Este servicio está disponible para la búsqueda y la invocación por un consumidor de servicios. El proveedor de servicios comprende la descripción de la aplicación y el servicio también ofrece la calidad de los requisitos de servicio a un consumidor.

Agilidad empresarial es adquirida por los sistemas de TI que sean flexibles, principalmente por la separación de la interfaz, la aplicación, y de carácter vinculante (protocolos) que ofrece SOA, que permite el aplazamiento de la elección de que el proveedor de servicios de optar por un determinado punto en el tiempo sobre la base de Nuevos requerimientos de negocio, (funcionales y no funcionales (por ejemplo, el rendimiento, la seguridad, la capacidad, y así sucesivamente) de las necesidades).

## 2.7. PLANTILLA PARA UNA ARQUITECTURA SOA[2]

Para el desarrollo de una Arquitectura Orientada a Servicios existe una plantilla, la cual permite visualizar claramente los diferentes servicios que se alinean con los procesos de negocios.

A continuación se muestran dos figuras las cuales describen la arquitectura y la interacción entre los diferentes elementos del proceso de negocio.

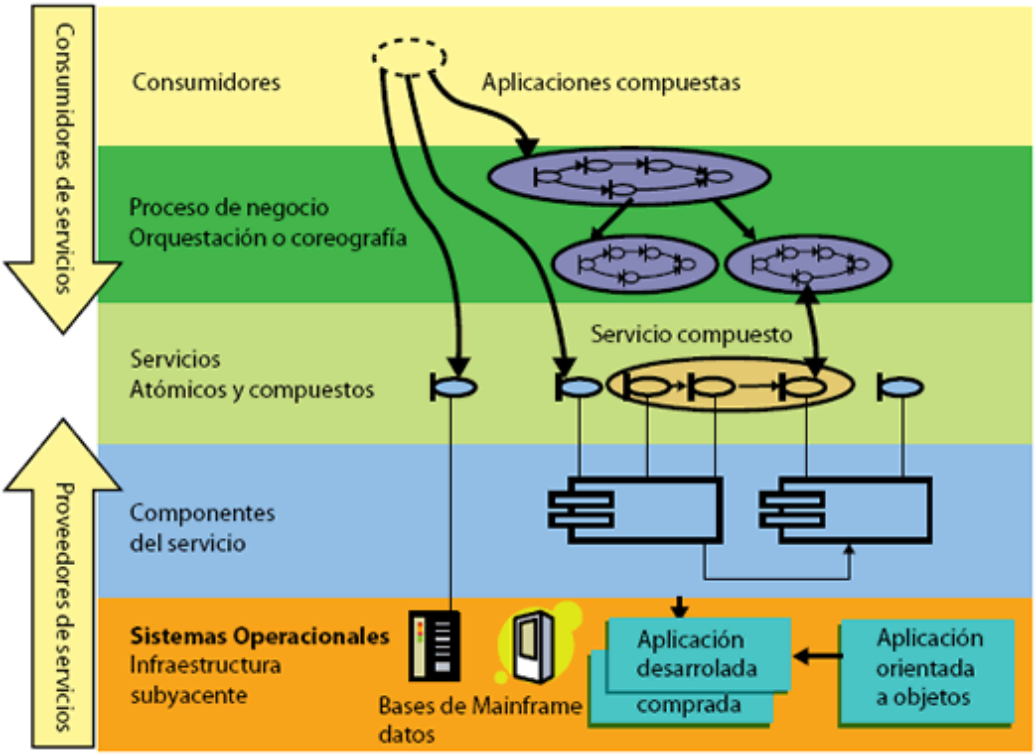


***Figura 6. Capas de una aplicación SOA***

Tomado de http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

[2] COLÁN, Mark. Service-Oriented Architecture expands the vision of Web services, Part 1.2004. http://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html

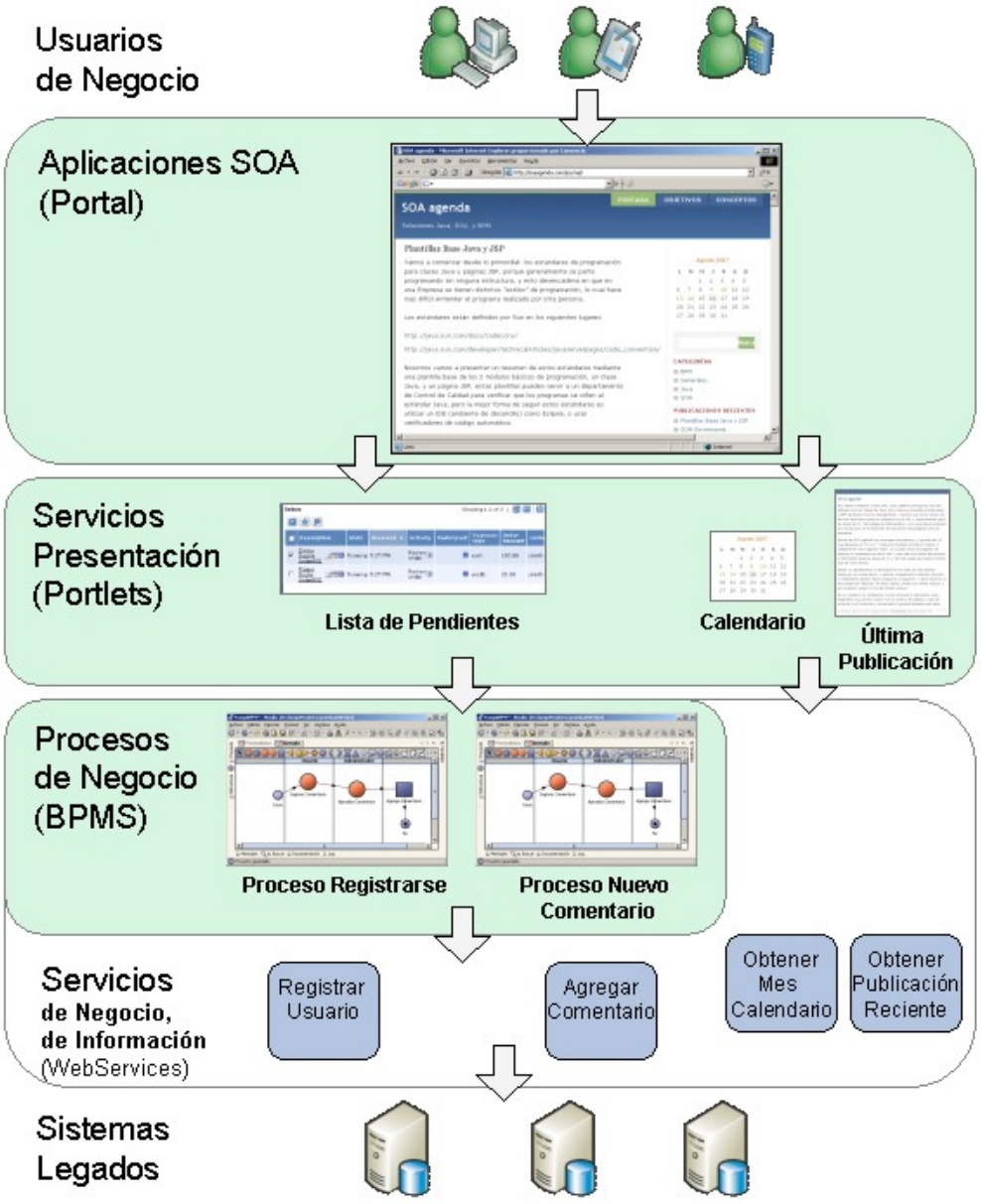


*Figura 7. Capas de una Arquitectura SOA*

Tomado de www.soagenda.com

Los componentes, que se muestran en las figuras, son los más generales y se toman como referencia. Sin embargo, cada proveedor de soluciones (IBM, Oracle, BEA, etc.) tiene su propia Arquitectura SOA de Referencia, que incorpora sus herramientas específicas, descritas a continuación:

- **Usuarios de Negocio**: Son los usuarios que participan en el proceso de negocio, utilizando distintas tecnologías para acceder a la aplicación (o proceso de negocio), tales como Desktop, Notebooks, Dispositivos móviles, etc.

- **Aplicación SOA y Portal**: Son las aplicaciones que implementan todo los descrito en este documento, destacando el concepto de reusabilidad, ya que se implementa mediante la utilización de componentes como son los Portlets y Servicios, para lo cual se utiliza la tecnología de Portales. Estos componentes garantizan el desarrollo de una aplicación orientada a Servicios y con la característica de poder ser implementada tanto en computadores, como en dispositivos móviles. Por ejemplo www.soa.com .

- **Servicios de Presentación (Portlets)**: Son los componentes realmente reutilizables, y se identifican por que de gran uso en la aplicación y cumplen la misma función. Ejemplos: un portlet de “Calendario”, un portlet para mostrar las “Publicaciones Recientes” de un blog. En el caso de los “Procesos de Negocio” (BPMS) generalmente ellos ofrecen un portlet para ejecutar los procesos, al que llamaremos portlet “Lista de Pendientes”.

- **Procesos de Negocio**: En esta etapa es donde implementamos las técnicas que veremos en el capitulo 3, una de ellas puede ser BPM, lo cual permite identificar las tareas interactivas (interacción participante), con actividades automatizadas (servicios), logrando un mejor proceso de negocio. Ejemplo: el proceso de “publicar un comentario en un Blog”, que dentro de sus tareas interactivas esta el “ingresar el comentario”, y “aprobar el comentario para su publicación”, y una actividad automatizada es el servicio de “ingresar el comentario en el sistema de Blog”.

- **Servicios de Negocio**: Son actividades del proceso de negocio que se pueden reutilizar, inclusive en una aplicación diferente, esto debido a que son servicios claramente identificados y garantizan su reusabilidad. Ejemplo “ingresarComentarioBlog”.

- **Servicios de Información**: Son los servicios que tienen como principal característica integrar los servicios islas o legados con el nuevo sistema que implementa aplicaciones orientadas a servicios. Son también llamados servicios atómicos, ya que encapsulan las funcionalidades de los sistemas existentes, presentándole una interfaz que permita integrarlos con la nueva aplicación, es decir, con el estándar de SOA.

- **Sistemas Legados**: Se puede decir que son servicios Islas, que tienen la característica que no hacen parte de la nuevo esquema Orientado a Servicios, pero que son los que soportaban la operación del negocio y pueden, en un momento dado aclarar e informar acerca de dudas, que se presenten en el proceso de negocio.

# 3. TÉCNICAS

Entre las técnicas para el desarrollo de aplicaciones SOA tenemos la combinación de Enterprise Architecture (EA), Businessman Process Modeling (BPM), Object-Oriented Analysis and Design (OOAD) definidos a continuación:

## 3.1. *Enterprise Architecture (EA)*[3]

La arquitectura de la empresa consiste en el conjunto de elementos que conforman entre si, una descripción exacta de las funciones y los objetivos de una empresa. Entre estos encontramos los objetivos estratégicos, los departamentos que la componen, las tecnologías que posee, el personal, etc. Son todos estos los que definen el bien o el servicio que brinda una organización a sus clientes.

En esta se vinculan también la misión, las estrategias, los modelos de negocio, los procesos de negocios y las redes y TI, tal como se muestra en la figura 8.

[3] ZIMMERMANN, Olaf. Elements of Service-Oriented Analysis and Design. 2004 http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

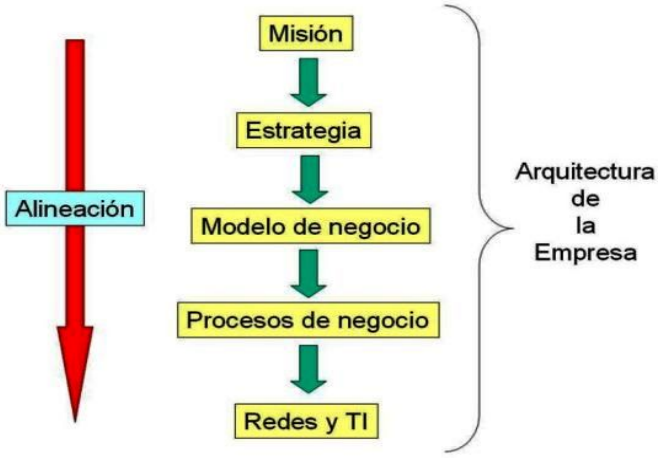


***Figura 8. Arquitectura de la empresa***

Tomado de http://es.wikipedia.org/wiki/Imagen:ArquitecturaDeLaEmpresa.jpg

**Misión**. Este es el nivel más alto y explica la razón de existencia de una organización.

**Estrategia**. Este nivel compone todas las prácticas realizadas por la empresa para cumplir con la Misión, y define los caminos y las maneras en las que se realizan las operaciones dentro de una organización.

**Modelo de negocio**. Este es con el que se logra la relación entre los procesos de negocio y las estrategias de la organización. Se define la forma en la que la compañía va a generar sus utilidades y como están relacionadas las diferentes áreas para brindar los bienes o servicios a sus clientes.

**Procesos de negocio**. En este nivel definimos las principales actividades de la organización y se presenta la manera en la que logra transformar los insumos en el producto que va a entregar a sus clientes. Estos procesos se pueden automatizar mediante sistemas.

**Redes y Tecnologías de la Información**. Esta define las tecnologías con la que es posible llevar a cabo los procesos de negocio y la manera en la que se distribuyen entre los clientes, proveedores y la misma empresa.

Con base en la arquitectura de la empresa, podemos definir cuales son las tecnologías necesarias para cumplir con los objetivos de la empresa.

### 3.2. *Businessman Process Management (BPM) 4*

Se conoce como **Business Process Management** a la metodología empresarial que mejora la eficiencia por medio de la sistematización de procesos que se deben modelar, automatizar, integrar, monitorizar y optimizar de forma continúa y lo que busca es una administración de los procesos de negocio.

Realizando un modelado de todas las actividades y procesos que se llevan acabo en una empresa, es mas fácil identificarlos y entenderlos, además de esto, la sistematización permite que la obtención de datos sea más rápida y eficaz.

Las herramientas que dan el soporte necesario para cumplir con el ciclo de vida de una empresa son conocidas como Business Process Management System y con ellas se construyen aplicaciones BPM.

Los principales motivos que llevan a utilizar BPM son:

- Extensión del programa institucional de calidad
- Cumplimiento de legislaciones
- Crear nuevos y mejores procesos

[4] ZIMMERMANN, Olaf. Elements of Service-Oriented Analysis and Design. 2004 http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

- Entender qué se está haciendo bien o mal a través de la comprensión de los procesos
- Documentar procesos para outsourcing y definición de SLA (Service Level Agreement)
- Automatización de procesos
- Crear y mantener las cadenas de valor

### 3.3. *Object-Oriented Analysis And Design (OOAD)* [5]

Para poder implementar SOA es necesario hacer uso de las técnicas de La Orientación a Objetos (OO) en la mayor medida posible. En el nivel de diseño el objetivo de OO es permitir que este sea rápido y eficaz y que el desarrollo y la ejecución de las aplicaciones sean flexibles y extensibles.

Quizás podríamos pensar que los principios de la programación OO, debido a conceptos que maneja tales como la herencia, el polimorfismo, el acoplamiento entre otros, van en contra de la programación orientada servicios, sin embargo son el pilar ya que a través de los componentes identificados en este tipo de programación podemos llegar a un nivel más alto como son los servicios, tal como lo ilustra las imágenes.

[5] ZIMMERMANN, Olaf. Elements of Service-Oriented Analysis and Design. 2004 http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

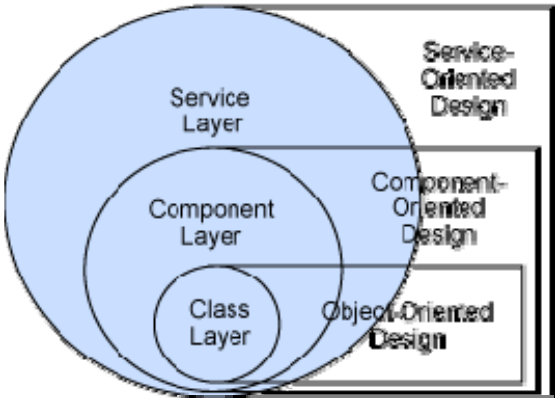


***Figura 9. Las capas de Diseño***

Tomado de http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/

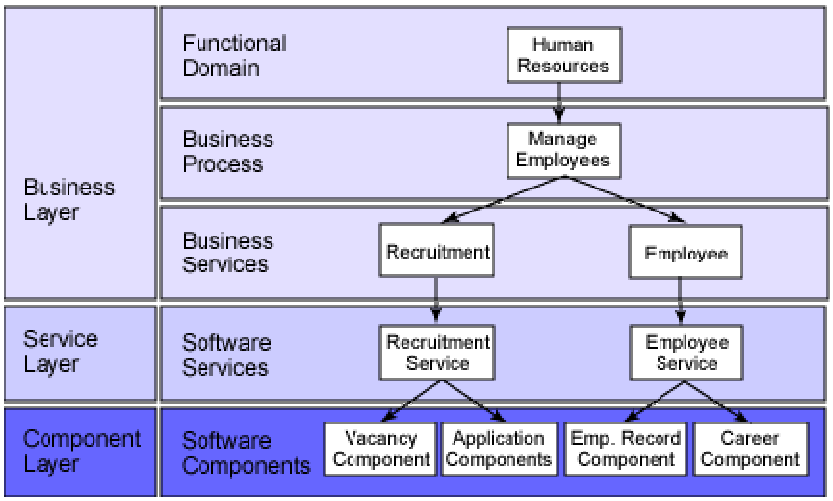


***Figura 10. Definición de jerarquía de un servicio SOA***

Tomado de http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/

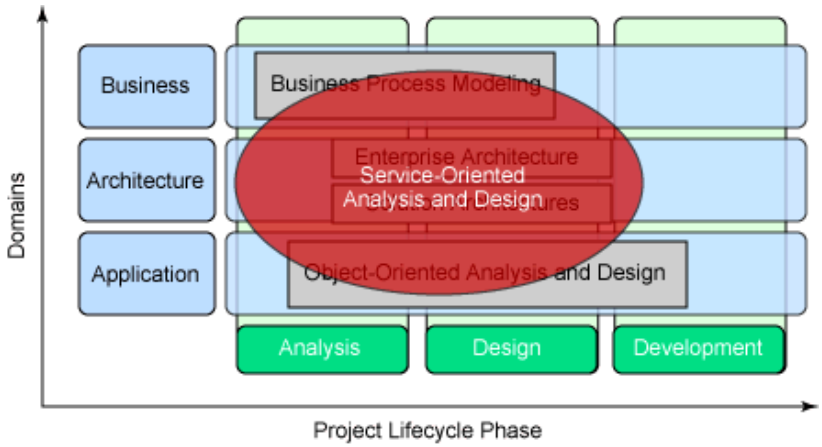


***Figura 11. SOA – Combinación de Elementos***

Tomado de http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/

### 3.4. *Extract, Transform and Load (ETL)*

El propósito principal de esta es traspasar información entre repositorios de datos y esta basada en procesos que utilizan archivos y bases de datos.

ETL es una de las tecnologías más antiguas de integración a nivel de datos y ha logrado mantenerse al día en cuanto a nuevas tocologías que incluso en la actualidad es utilizada dentro de SOA. La base de ETL es extraer, transformar y cargar información entre diferentes tipos de repositorios de datos.

- **Extraer**: Permite sacar información de diferentes fuentes de datos, principalmente bases de datos, por medio de tablas o de procedimientos almacenados. También permite extraer datos de archivos diferentes a bases de datos como lo son XML, Excel, CSV, Texto, etc. E inclusive se ha logrado obtener información de colas de mensajes, objetos Java y servicios Web.

- **Transformar**: Esta función recopila toda la información sacada de un repositorio de datos y hace las conversiones necesarias para que esta pueda ser entregada por medio de cargar entre servicios.

- **Cargar**: Permite guardar la información que fue extraída y transformada en un nuevo repositorio que puede ser de cualquier tipo como se explico previamente.

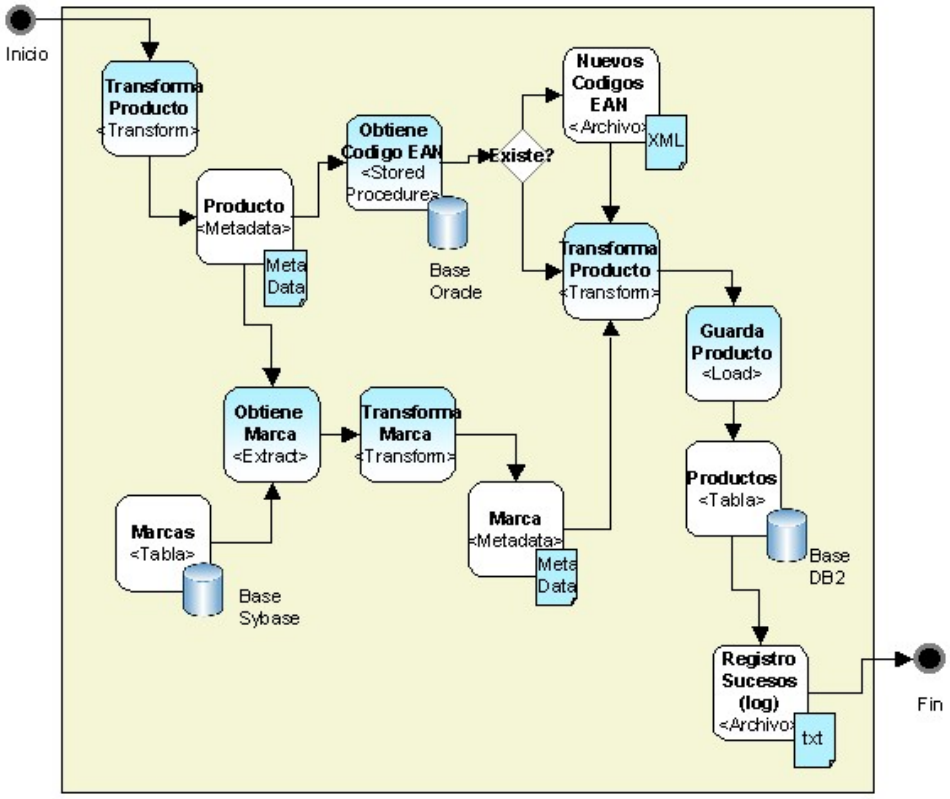


*Figura 12. Ejemplo: Proceso de Información Agrega Producto*

Tomado de www.soaagenda.com

### 3.5. *WorkFlow Tradicional*

Esta es la tecnología que permite la coordinación de actividades realizadas por personas o Human Task. En este se definen roles, actividades, reglas de negocio, que definen el flujo de trabajo (WorkFlow), pero que a diferencia de BPM no contempla las actividades de sistemas (System Task) bajo el estándar SOA.

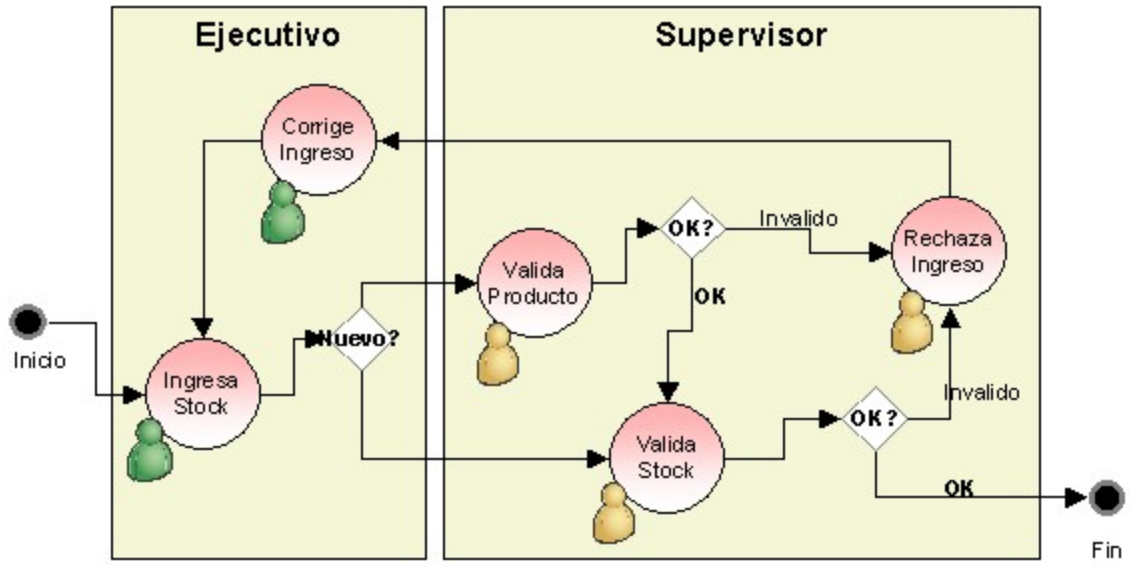


*Figura 13. WorkFlow. Actualizar Stock de Productos*
Tomado de www.soaagenda.com

Dentro de esta tecnología, es el usuario el encargado de definir en que momento se finaliza una actividad y define así la secuencia del Work Flow.

Con Work Flow se le da seguimiento a las actividades que se realizan conociendo en que estado y en que parte se encuentra la información. La información que se trabaja en estos flujos de trabajo es conocida como metadata u objeto de negocio.

### 3.6. *Enterprise Service Bus (ESB)*

Esta es la tecnología que administra todo lo correspondiente a actividades realizadas por el sistema o actividades automatizadas.

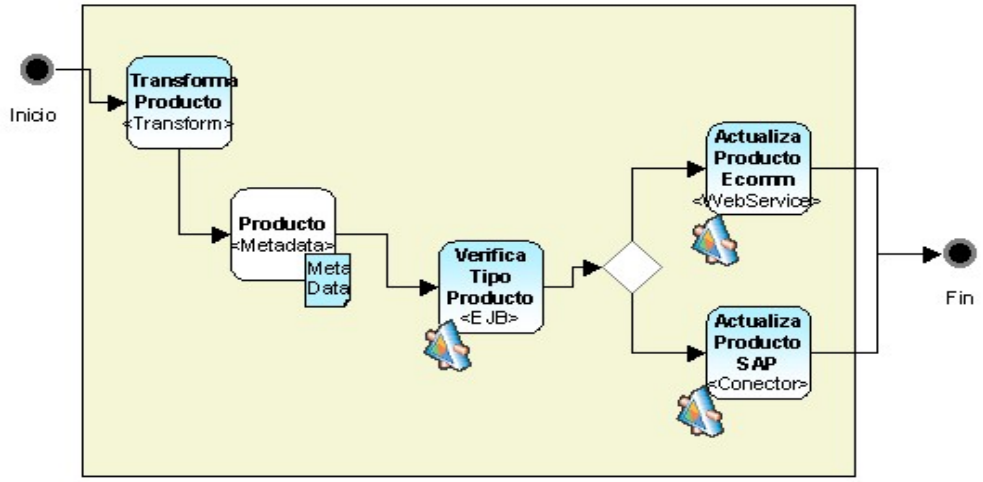


*Figura 14. Proceso ESB. Actualiza Producto en Sistemas*
Tomado de www.soaagenda.com

La diferencia con BPM es que el ESB solo maneja actividades automatizadas (System Task).

La diferencia con ETL (tradicional) es que ETL esta orientado a componentes de datos, y ESB esta orientado a componentes de Servicios.

Los objetos que principalmente maneja ESB son:

- WebServices
- Conectores a Colas de Mensajes, y JMS
- Conectores a Aplicaciones de clase Mundial (PeopleSoft, SAP, JDEdwards)
- Conectores a J2EE (EJB) y .Net
- Transformaciones de Datos

### 3.7. *BPM Suite.*

BPM es la tecnología que junta los conceptos de Work Flow con los de ESB, es decir que trabaja tanto actividades humanas como del sistema. Para esto utiliza una sola interfaz en la que combina las dos tecnologías.

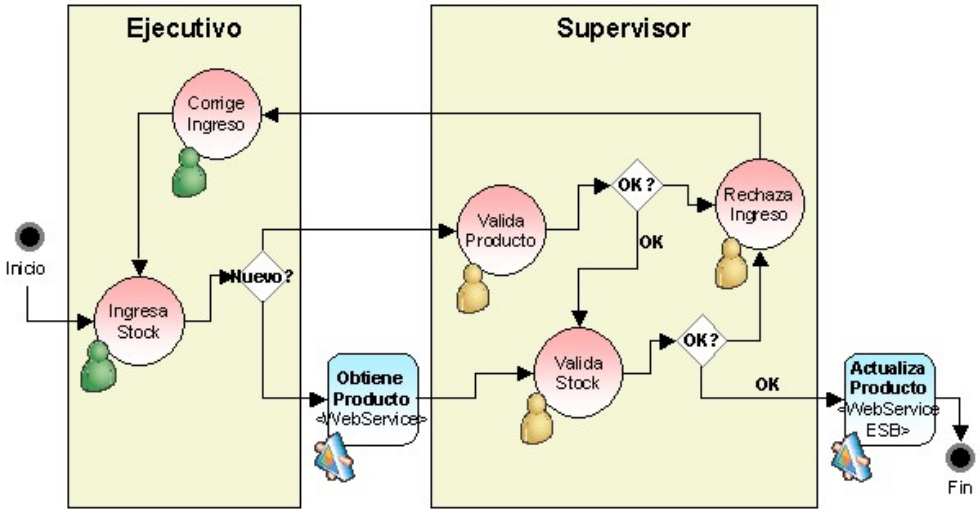


*Figura 15. Ejemplo. Proceso de Negocio Actualizar Stock de Productos*
Tomado de www.soaagenda.com

BPM cuenta con tres herramientas que permiten el control del ciclo de vida de un proceso con el fin de obtener un mejor manejo de los procesos de negocio, estas son el moderador, el IDE y el monitor.

# 4. PATRONES SOA

## 4.1 PATRONES DE ARQUITECTURA

### *4.1.1 Patrón Layer*

Se encarga de estructurar el sistema en varias capas, de tal forma que cada capa es responsable de una serie de funciones. Cada capa solicitará la ejecución de determinadas operaciones a la capa inmediatamente inferior, procesará la respuesta y devolverá el resultado al nivel superior.

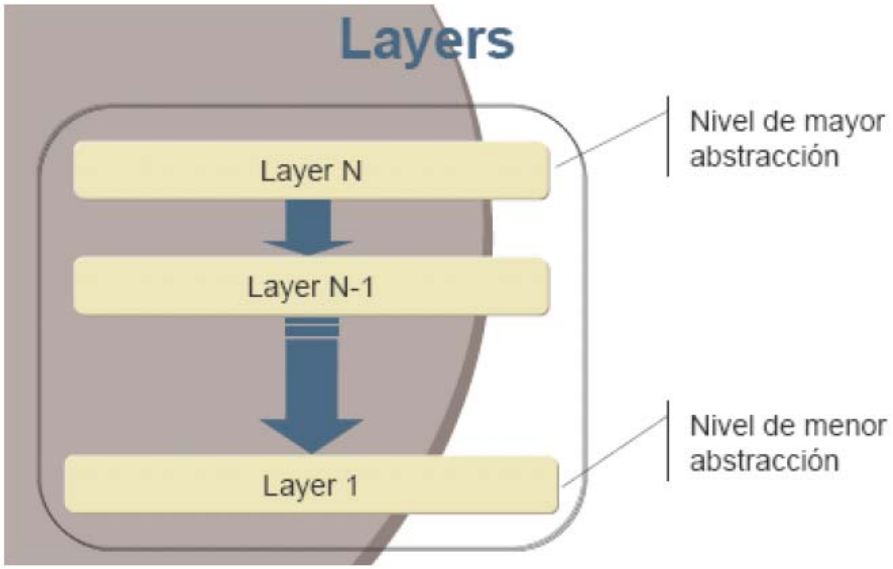


***Figura 16. Ejemplo Estructura Patron Layer***
RIVERA, Ignacio. Patrones Arquitectónicos Layers. Octubre, 2006, 2p

### *4.1.2 Patrón Broker*

Este patrón se encarga de conectar diferentes componentes de un sistema distribuido de forma débilmente acoplada mediante una red en estrella. Hay un objeto Broker en el centro de la red al que todos los componentes están conectados. En un momento dado que un componente necesite ponerse en contacto con otro envía un mensaje al Broker y este a su vez lo reenvía al componente destino.

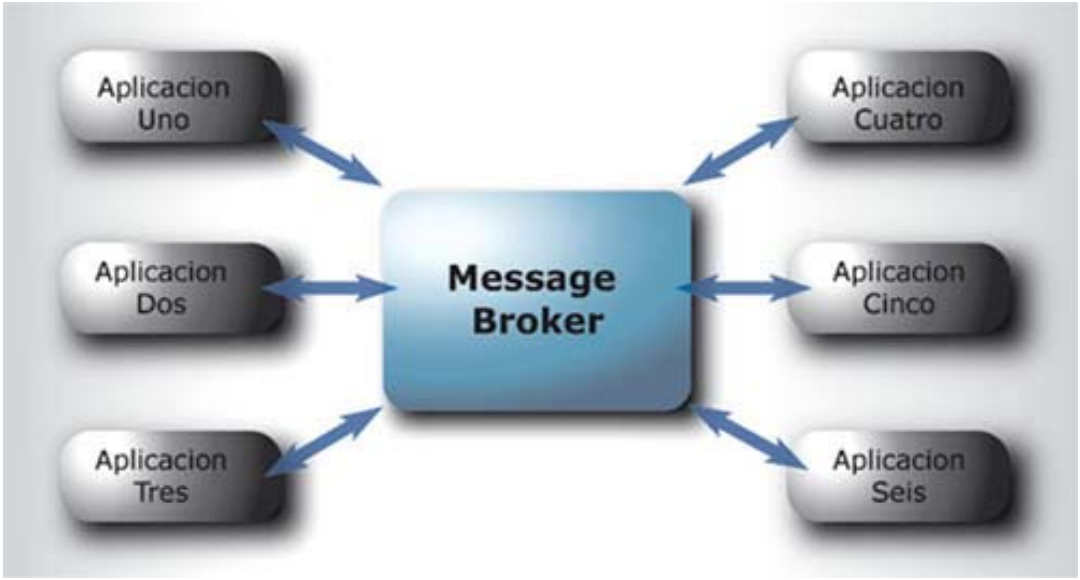


***Figura 17. Patrón Broker***
Tomado de **www.microsoft.com/.../MTJ_3625/default.aspx**

## 4.2 PATRONES DE DISEÑO

### *4.2.1 Modelo Vista Controlador*

Consiste en realizar un modelo o diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad, conseguir una

interfaz escalable y fácilmente adaptable a las necesidades del usuario final. De esta forma las modificaciones en las vistas impactan en mejor medida en la lógica del negocio o de datos. Los componentes son tres: Vista, que es donde reside la lógica de presentación; Controlador, que es el que gestiona el flujo de la presentación de las vistas y se encarga de tareas como la autenticación, autorización., etc.; por último Modelo, son las clases que albergan la lógica del negocio y de entidad de la aplicación.

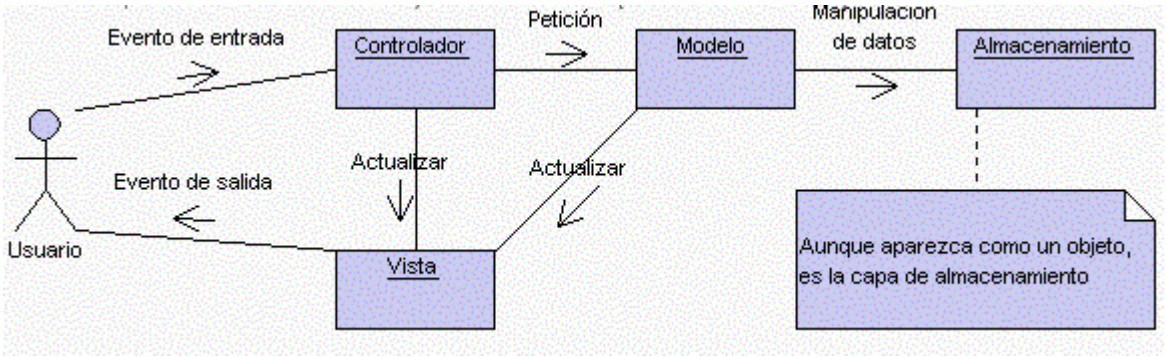


***Figura 18. Patrón MVC***
Tomado de http://www.proactiva-calidad.com/java/patrones/mvc.html

### *4.2.2 Patrón Proxy*

Se utiliza como un intermediario para acceder a un recurso, con varios propósitos, tales como diferir la carga, controlar el acceso, hacer caché, el funcionamiento consiste en utilizar una clase que implementa la misma interfaz de otro objeto, generalmente remoto, es decir se invoca al objeto real, posiblemente externo y de más difícil acceso, dentro de una clase más cercana y accesibles por los posibles clientes del Proxy.

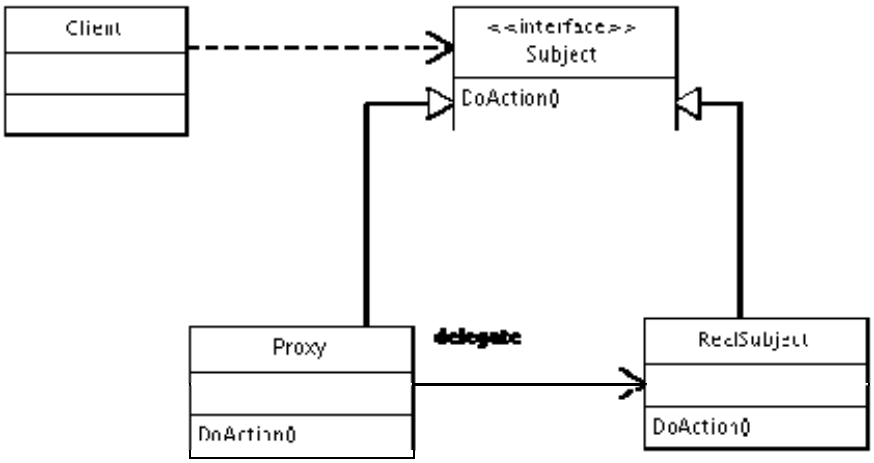


***Figura 19. Patrón Proxy***
Tomado de http://www.proactiva-calidad.com/java/patrones/mvc.html

### *4.2.3 Patrón Facade*

Sirve para proveer una interfaz unificada y sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas.

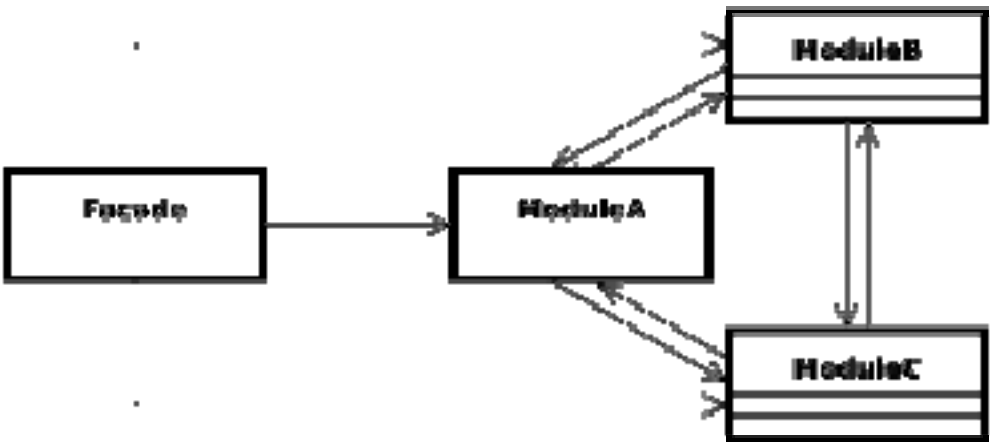


***Figura 20. Patrón Facade***
Tomado de http://es.wikipedia.org/wiki/Facade_(patr%C3%B3n_de_dise%C3%B1o)

### 4.2.4 Patrón Unit of Work

Mantiene una lista de objetos que son afectados por una transacción de negocio, coordina las acciones tras el éxito de una transacción compleja y controla posibles problemas de concurrencia.

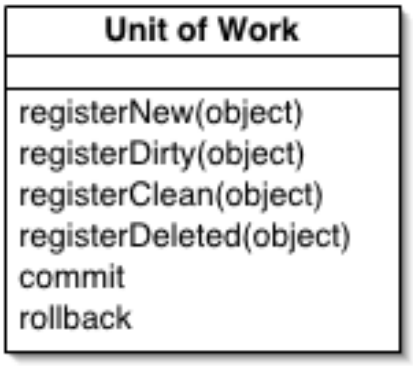


***Figura 21. Patrón Unit of Work***
Tomado de **lucasontivero.spaces.live.com/blog/cns!375AE0...**

### 4.2.5 Patrón Observer

Define una dependencia de 1:n de forma que cuando el objeto 1 cambie su estado, los n objetos sean avisados y se actualicen automáticamente, dicho en otras palabras proporciona a los componentes una forma flexible de enviar mensajes de difusión a los receptores interesados.

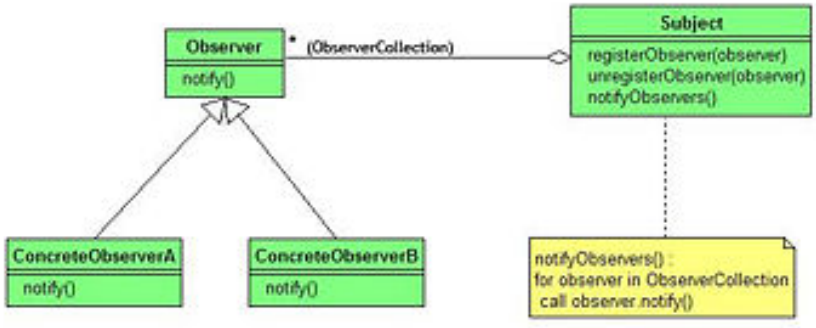


***Figura 22. Patrón Observer***
Tomado de http://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o)

### *4.2.6 Requester side cachê*

Este patrón permite intermediar entre uno o más clientes y uno o más proveedores de información, esto trae como ventaja que el acceso a la información sea más rápido y menos costoso. Con este patrón se puede diseñar una política para cachear el origen de datos.

## 4.3 PATRONES DEL SISTEMA DE MENSAJE

### *4.3.1 Patrón Broker de Mensaje*

Este patrón se encarga de recibir los mensajes desde múltiples destinos y se encarga de determinar el destino correcto de cada mensaje.

### *4.3.2 Process Manager*

Este patrón permite tomar decisiones de encaminado en tiempo de ejecución, ya que permite encaminar mensajes en distintos pasos decidiéndose por un destinatario u otro en función de la respuesta del paso anterior y el estado que almacena el componente.

### *4.3.3 Router basado en Contenido*

Este patrón como su nombre lo indica, enruta los mensajes hasta su destinatario, leyendo el valor de algún campo de contenido de este y luego lo clasifica utilizando un criterio determinado.

### *4.3.4 Lista de Destinatarios*

Este patrón recibe los mensajes por un canal de entrada y utilizando una estructura con la cuentan los mensajes, seleccionara el destinatario y enviara los mensajes por el canal de salida que los conduzca a estos.

### *4.3.5 Traductor de mensajes*

Este patrón toma los mensajes por su canal de entrada y luego de examinarlos transforma la información en un nuevo mensaje y lo envía a su destinatario.

### *4.3.6 Modelo Canonico*

Este patrón crea un modelo de información con el que se representan entidades, implementa transformadores para así convertir los mensajes que ingresan al modelo propio y de ese momento en adelante, trabaja con su propio modelo.

### *4.3.7 Messaging Gateway*

Es una variación del patrón gateway que consiste en envolver las diferentes funcionalidades de una aplicación y exponerlas a modo de API hacia los sistemas de mensaje.

### *4.3.8 Service Activator*

Es una variación del patrón Message Gateway, que consiste en un End-point que trabaja como intermediario entre unos sistemas de mensajes y una aplicación que no conoce como comunicarse directamente con el sistema de mensajes. Este recibe mensajes de ejecución de algún servicio de la aplicación e invoca a la aplicación que se esta llamando haciendo las veces de cliente, reestructurando el mensaje y reenviándolo a través de su canal de respuesta.

### *4.3.9 Polling Consumer*

Este tipo de End-points trabajan activamente como solicitantes de mensajes al canal de entrada.

### *4.3.10 Competición de Consumidores*

Se trata de un conjunto de hilos que compiten entre si para obtener primero un mensaje de un determinado canal.

### *4.3.11 Consumidor dirigido por eventos*

Este a diferencia del Polling Consumer, esta inactivo hasta que recibe una señal de nuevo mensaje.

# 5. TECNOLOGÍA

Es necesario para la implementación de un sistema sobre la arquitectura planteada por SOA, tener herramientas para la descripción y la ejecución de los servicios establecidos. Para esto, existen las tecnologías WS-BPEL, WS-CDL, UDDI y SOAP para desarrollar dichas tareas y permitir la elaboración de procesos bien estructurados con la propiedad de ser una arquitectura de servicios.

## 5.1. WS-CDL (Web Services-Choreography Escription Language)

Protocolo que tiene como principal objetivo la descripción del comportamiento de cada uno de los servicios establecidos para lograr una finalidad, es un protocolo utilizado para la definición de servicios dentro de la plataforma SOA, basado en XML. Fue un protocolo que evoluciono logrando ir perfeccionándose hasta poder identificar algunos comportamientos que inicialmente no eran observables, los cuales permiten identificar las razones por las cuales un proceso obtuvo un determinado resultado. Es decir, la descripción de la integración realizada entre los servicios para lograr el objetivo definido.

WS-CDL es importante dentro de SOA porque es una tecnología escalable, garantiza la interoperabilidad efectiva y segura de servicios, permite tener servicios más robustos reduciendo el tiempo de implementación de los mismos, esto debido a que este lenguaje o protocolo permite la descripción sin ambigüedades de las colaboraciones establecidas entre servicios, determinando

un protocolo de negociación, de tal forma que cada organización desarrolle de manera independiente su propio rol en el desarrollo de la aplicación, respetando el contrato global para que se garantice la interoperabilidad.

**Estructura de WS-CDL**

WS-CDL es un lenguaje organizado por capas, que permiten diferentes niveles de expresión de las coreografías de un servicio. En el nivel más alto, existe un paquete que contiene todas las definiciones realizadas por WS-CDL, estas coreografías, deben incluir como mínimo un conjunto de roles definidos por ciertos comportamientos, una serie de relaciones entre dichos roles, canales utilizados por los roles para interactuar y un bloque de coreografías utilizado por los canales para definir la interacción. En este nivel, se describe un conjunto básico de conexiones de servicios que permiten la colaboración entre roles para lograr un objetivo; sin embargo es posible adicionar una composición estructurada, permitiendo la combinación en secuencias o actividades paralelas de las interacciones y otras coreografías.

## 5.2. WS-BPEL (Web Services-Business Process Execution Language)

Es un lenguaje basado en la orquestación, es decir, que permite la definición de servicios, a partir de servicios ya existentes, nace de la necesidad de integrar o engranar las diversas tecnologías que funcionan bajo el ambiente de SOA.

WS-BPEL nace tomando propiedades de dos lenguajes propietarios con características diferentes pero complementarias, estos lenguajes son: WSFL (Web Services xte Language), desarrollado por IBM, el cual se basa en gráficos de actividades y conectores de control para definir un modelo global describiendo las interacciones entre los servicios Web existentes y permitiendo la definición de

nuevos servicios a partir de los existentes. El otro lenguaje es XLANG desarrollado por Microsoft, el cual basa su funcionamiento en una notación orientada al comportamiento de intercambio de mensajes a través de los Servicios Web participantes, de esta manera automatiza el proceso de negocios. WS-BPEL combina el estilo orientado en gráficos de WSFL y el estilo algebraico propuesto por XLANG.

WS-BPEL es visto como un lenguaje basado en un flujo de trabajo extensible, el cual permite la agregación de servicios de manera recursiva y en un entorno altamente dinámico; WS-BPEL al permitir un entorno altamente dinámico favorece el cambio frecuente de servicios, desarrollando procesos desacoplados para que se adapten de manera adecuada a instancias particulares de servicios en una coreografía específica.

**Arquitectura WS-BPEL**

WS-BPEL se basa en un modelo de composición el cual establece varios requerimientos que se cumplen para su correcto comportamiento; el modelo de composición requiere una integración flexible para que se puedan expresar de manera adecuada los escenarios de negocios y se adapten fácilmente, requiere una composición recursiva permitiendo la integración de servicios Web e incrementando la escalabilidad y la reusabilidad, requiere separación y habilidad de composición definiendo cómo el servicio Web hace parte de un ”Framework” pero desacoplándolo de mecanismos pertenecientes al ”Framework” como son la calidad del servicio, requiere conversaciones estables y manejo de ciclo de vida donde el flujo de trabajo tiene definido un modelo de ciclo de vida y los servicios Web pueden mantener varias conversaciones con los servicios que interactúan con él. El bloque principal de un proceso de negocios en BPEL contiene primordialmente las relaciones con servicios asociados externos, declaraciones

para procesos de datos y las actividades a ser ejecutadas. Los datos en BPEL pueden ser leídos o escritos por las actividades que intercambian mensajes entre procesos, además permite la manipulación de expresiones para facilitar el proceso del negocio.

Las actividades en un proceso WS-BPEL pueden ser de carácter básico o estructurado, las actividades estructuradas contienen otras actividades, combinan múltiples actividades y definen la lógica de negocios entre ellas, mientras las actividades básicas se encargan de la manipulación de datos o de las interacciones entrantes o salientes en un Servicio Web. En las actividades estructuradas se puede hacer uso de actividades o expresiones para combinar actividades, tales como: Secuencia (sequence), selección (switch), repetición (while), flujo (flow); Además, las actividades tratadas en un servicio Web pueden ser recibidas (receive), invocadas (invoke) o replicadas (reply). Las anteriores funcionan a similitud como un lenguaje de programación estructurado.
WS-BPEL maneja condiciones de transición (transition) o de unión (join), donde las condiciones de transición están asociadas con cada vínculo de control y son evaluadas en la completitud de la actividad fuente del vínculo; Las condiciones de unión están asociadas con una actividad que es un punto de referencia para el vinculo (Figura23).

WS-BPEL soporta dos tipos de patrones de uso, los cuales son: procesos abstractos (abstract processes) y ejecutables (executable processes); los procesos abstractos se utilizan para describir los protocolos del negocio o protocolos de intercambio de mensajes, los cuales describen las interacciones visibles externamente sin necesidad de definir la lógica interna del negocio, por el contrario, los procesos ejecutables se encargan de la lógica del negocio con el servicio asociado que se encuentra detrás de los protocolos externos. Los dos tipos de procesos utilizan el mismo lenguaje de construcción, utilizando construcciones del lenguaje específicas según el tipo de proceso.

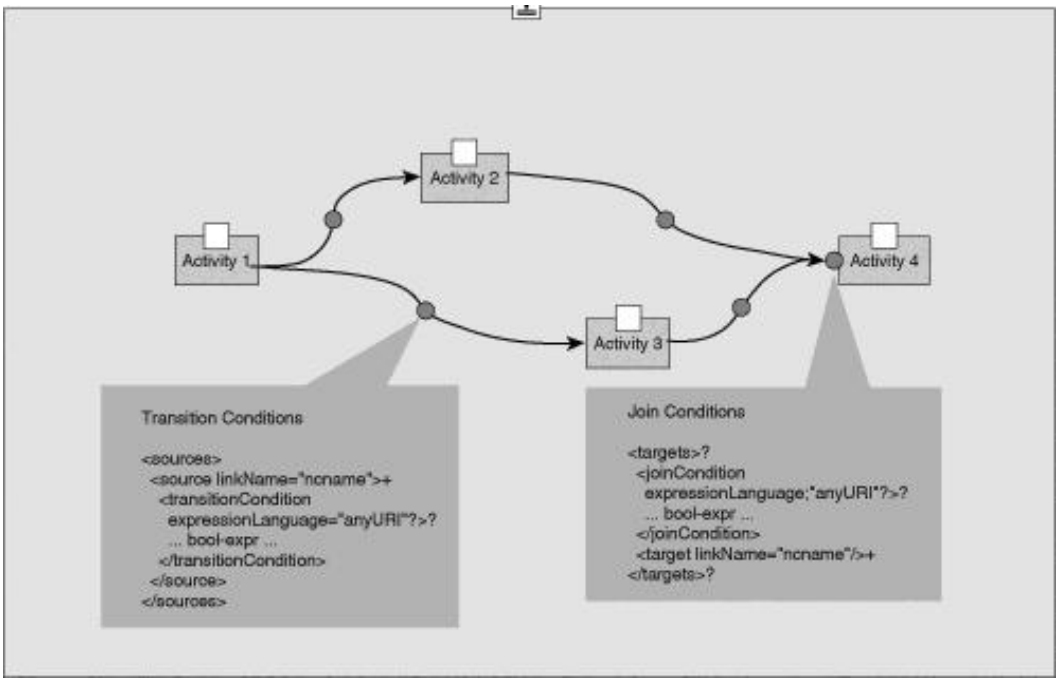


***Figura 23. Proceso con WS-BPEL***

Tomado de Documento de estado de Arte en SOA y Cálculo PI, p7.

## 5.3. UDDI (Universal Description, Discovery & Integration)

UDDI juega un papel clave dentro de SOA, ya que a través de esta herramienta, podemos conocer todos los servicios que ofrecen las diferentes compañías en la Web, que clase de servicios son, como es su utilización, etc. Lo que hace UDDI es construir un catalogo universal de servicios Web con una infraestructura muy similar a la del DNS.

UDDI se divide en tres secciones, la sección blanca en la que están la descripción de los servicios y sus proveedores, la sección amarilla en la que están los servicios Web que proporcionan y la sección verde en la que esta toda la información técnica para el acceso a los servicios. Es importante resaltar que UDDI trabaja como un registro de servicio mas no como repositorio de estos,

como se intento hacer en un principio. UDDI se basa en varios estándares ampliamente difundidos como HTTP, XML, XML Schema (XSD), SOAP y WSDL.

Un registro UDDI esta formado por nodos, que trabajan como servidores que soportan la interacción de datos UDDI mediante al menos una API, como se muestra en la siguiente figura.

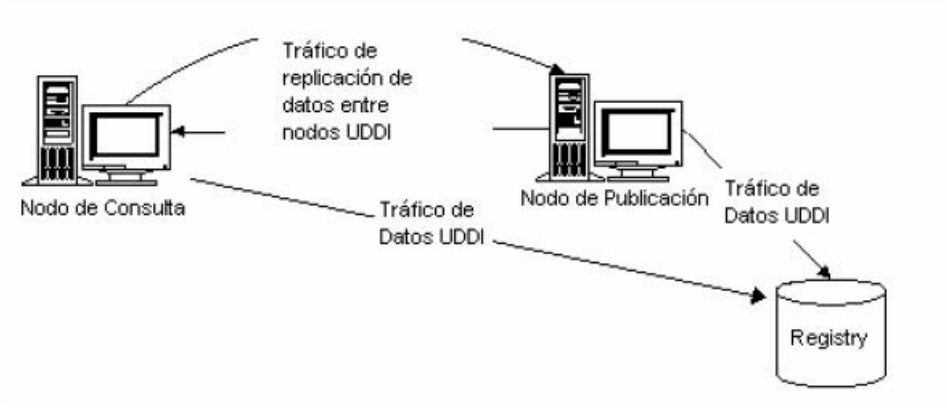


***Figura24. Esquema de Nodos de un registro UDDI***
Tomado de Documento Extensibilidad de UDDI, p17.

**Estructura de datos UDDI**

Este tipo de estructuras de datos conforman un modelo de información persistente que almacena en su registro toda la información que necesita para su correcto funcionamiento. Estas estructuras se expresan en varios esquemas XML. El identificador de este tipo de estructuras de datos es conocido como UDDI Key.

Como se planteo anteriormente, UDDI se divide en secciones, y sus estructuras de datos están enfocadas hacia estas mismas así es como para la sección blanca esta la estructura que define el proveedor de servicios Web que es bussinessEntity. Para la sección amarilla la estructura encargada de almacenar la información de los servicio ofrecidos por un proveedor es la bussinessService. La sección verde contiene la estructura bindingTemplate, la cual almacena la

información técnica para el acceso a los servicio prestados por un proveedor y la estructura tModel que almacena modelos técnicos o metadatos reutilizables que pueden representar cualquier concepto como el tipo de servicio Web, algún protocolo usado por los servicios o un sistema de categorización.

Además de estas entidades existen otras de primer nivel como pubisherAssertion y subscirption, en las que la primera representa la relación existente entre las bussinesEntity y la segunda describe un mecanismo de notificación y transferencia de cambios sobre entidades.

Existe una relación de jerarquía padre-hijo entre las entidades bussinnessEntity, bussinessService y bindingTemplate. Un bussinnessEntity contiene información del proveedor y de los servicios que ofrece, a través de los bussinessService y de los bindingTemplate que dependen de él. El bussinnessEntity contiene el nombre del proveedor, una descripción, una URL de referencia, información de contactos y de clasificación de su actividad. Toda la información admite multiplicidad y esto se da ya que se puede concretar en varios idiomas o porque la propia información es múltiple, como los contactos o las actividades del proveedor de servicios.

### 5.4. SOAP (Simple Object Access Protocol)

A pesar de que en un principio fue concebida como la solución a la brecha entre plataformas basadas en RPC, SOAP se ha convertido en el protocolo de intercambio de mensajes entre servicios Web mas utilizado, por esto se utiliza mas este concepto relacionándolo como el protocolo de la arquitectura orientada a servicios que al protocolo de acceso simple a objetos.

Las especificaciones de SOAP establecen un estándar de formato de mensaje que consta de un documento XML, capaz de acoger un RPC y un documento centrado

de datos. Esto facilita tanto la sincronía (Petición y respuesta) como la asincronía (Procesos impulsados por un evento) de intercambio de datos entre modelos

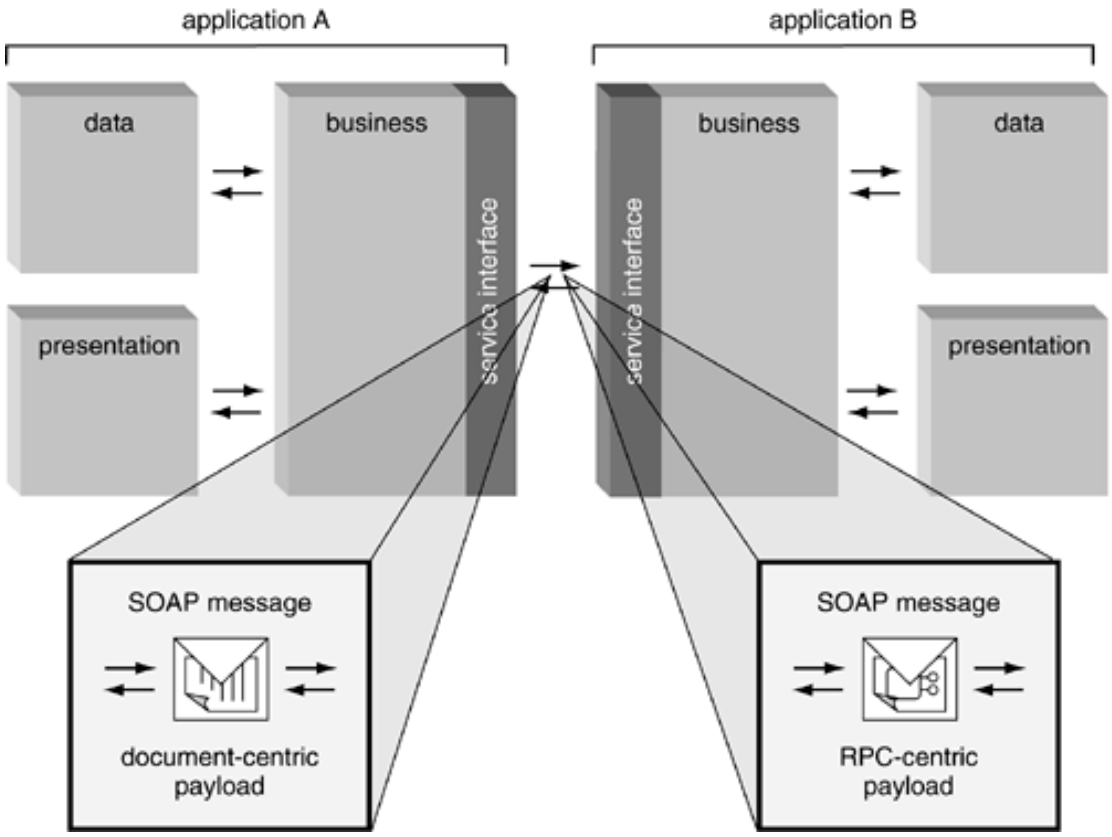


***Figura 25 Principales formatos de mensajes estándar***

Tomado de http://www.informit.com/articles/article.aspx?p=336265&seqNum=3

### *SOAP* Messaging framework

SOAP tiene un método de comunicación basado en un modelo de transformación que está relacionado con el marco general de los servicios Web y se diferencian sólo en la medida en que se introducen términos y conceptos que se relacionan específicamente con la manera en que *los mensajes SOAP* deben manejarse.

## Nodos *SOAP*

Un nodo SOAP representa la lógica de procesamiento de los responsables de transmitir, recibir y realizar una variedad de tareas de procesamiento de los mensajes SOAP. Una aplicación de un nodo SOAP es típicamente una plataforma específica, y es comúnmente esta etiquetada como SOAP server o SOAP listener. También existen variaciones, como los SOAP routers.

SOAP establece el nodo como el mecanismo de transporte subyacente para un servicio Web.


***Figura 26 Nodo SOAP***

Tomado de http://www.informit.com/articles/article.aspx?p=336265&seqNum=3

## Tipos de Nodos *SOAP*

Al igual que los servicios Web, los nodos *SOAP* pueden existir como remitentes iniciales, intermediarios y receptores. Considerando que los servicios Web también están clasificados como solicitantes y proveedores, la ejecución de nodos *SOAP* equivalen a tareas (enviar, recibir) y se denominan *SOAP remitentes* y *SOAP receptores.*

# 6. METODOLOGÍAS DE DESARROLLO PARA EL DISEÑO DE APLICACIONES ORIENTADAS A SERVICIOS (SOA)

## 6.1. PRINCIPIOS DE ORIENTACIÓN A LOS SERVICIOS

Estos son principios que nos indican si un desarrollo realizado con base en SOA es un desarrollo compatible con estos conceptos en verdad, es decir, es como una especie de verificación para comprobar que en verdad nuestro desarrollo es SOA. A continuación se presentan cuales son los principios que debe cumplir un desarrollo de una aplicación SOA:

- **Los Servicios deben ser reusables:** Todos los servicios deben tener como base el concepto de reutilización, esto es que puedan ser usados nuevamente dentro de la misma aplicación, dentro de la empresa que desarrolla la aplicación o para todo el público que desarrolle aplicaciones que busquen satisfacer el mismo servicio.

- **Los Servicios deben proporcionar un contrato formal:** Todo servicio desarrollado, debe proporcionar un contrato en el cual figuren: el nombre del servicio, su forma de acceso, las funcionales que ofrece, los datos de entrada de cada una de las funcionalidades y los datos de salida. Esto con el fin de lograr la independencia entre el usuario y la implementación de la aplicación, por medio del acceso utilizando el contrato o definiciones preestablecidas.

- **Los Servicios deben tener bajo acoplamiento:** Es decir, que los servicios tienen que ser independientes entre ellos. Nuevamente aquí se utiliza el contrato, pero esta vez buscando la independencia entre servicios. Llevando a cabo este principio, podemos lograr otro que es que los servicios sean reusables.

- **Los Servicios deben permitir la composición:** Esto quiere decir, que los servicios deben ser desarrollados buscando que un conjunto de estos sean capaces de formar un servicio de más alto nivel. En el caso de los Servicios Web, esto se logrará mediante el uso de los protocolos para orquestación (WS-BPEL) y coreografía (WS-CDL).

- **Los Servicios deben de ser autónomos:** Este es otro de los principios que busca la reutilización de servicios y lo hace creando ambientes de ejecución propios de cada servicio, con el fin de que sea autónomo dentro de la aplicación.

- **Los Servicios no deben tener estado:** Esto quiere decir que ningún servicio debe guardar ningún tipo de información, con el fin de prevenir errores generados por información errada o mal entregada entre servicios.

- **Los Servicios deben poder ser descubiertos:** Con esto se busca evitar la creación de servicios que cumplan las mismas funciones o duplicidad de servicios. En el caso de los Servicios Web, el descubrimiento se logrará publicando los interfaces de los servicios en registros UDDI.

Cuando nos tomamos la tarea de realizar un desarrollo basándonos en SOA, es necesario seguir estas pautas con el fin de cumplir lo que se busca con esta metodología de desarrollo y recibir así todas sus ventajas.

Queda evidenciado en esta parte que todos los **Principios de la Orientación a Servicios** están inter-relacionados.

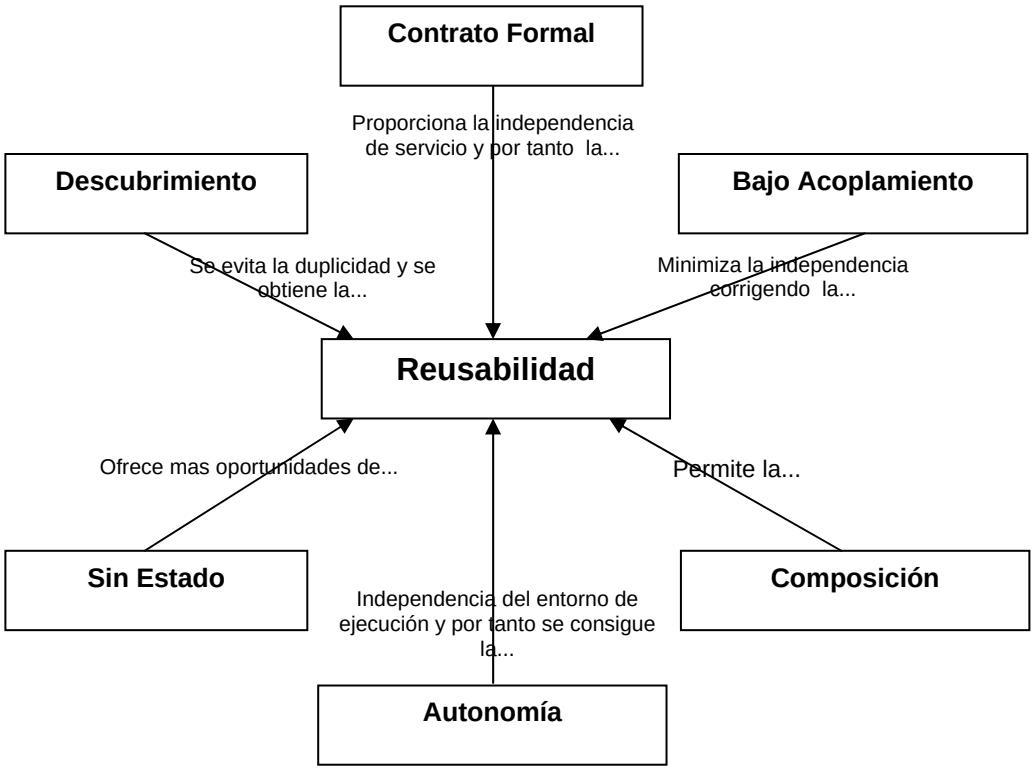


***Figura 24. Representación de los principios de la orientación a servicios***

Tomado de http://arquitecturaorientadaaservicios.blogspot.com/2006/06/principios-de-la-orientacin-servicios.html

## 6.2. METODOLOGÍAS DE DESARROLLO ORIENTADO A SERVICIOS.

EL conjunto de normas y practicas que de deben implementar dentro de el desarrollo basado en la metodología SOA que nos permiten garantizar la eficiencia y respuesta del modelo, esta definido por el Desarrollo Ágil de Software descrito a continuación.

### 6.2.1. *Metodologías Ágiles* [6]

Este concepto hace relación al conjunto de procesos utilizados en el desarrollo de software para que estos sean creados de una manera más rápida, eficiente, ligera y enfocada a los usuarios.

Este concepto nace en febrero de 2001, tras una reunión celebrada en EEUU. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto, para dejar a un lado los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó The Agile Alliance3, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía ágil.

**El Manifiesto Ágil.**

Según el Manifiesto se valora:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La clave esta en elegir primero el grupo de desarrollo antes que el entorno, esto buscando que sean los miembros del grupo

[6] CANÓS, José H. LETELIER, Patricio. PENADÉS, María Carmen. Metodologías ágiles en el desarrollo de software. 2005, 1p. Documento de Investigación. (Ingeniería de Sistemas). Universidad Politécnica de Valencia.

quienes definan el entorno en que para su sentir, serán mas productivos y eficientes.

- Desarrollar software que funciona más que conseguir una buena documentación. No producir documentos que conviertan el desarrollo en algo engorroso, producir solo los documentos que necesitemos para tomar una decisión importante.

- La colaboración con el cliente más que la negociación de un contrato. El cliente debe estar compenetrado con el grupo de desarrollo, esto con el fin de lograr que la aplicación desarrollada sea eficaz y brinde los servicios que son necesarios.

- Responder a los cambios más que seguir estrictamente un plan. Ser capaces de asimilar cualquier cambio que se presente en la estructura del grupo o en el mismo plan de desarrollo.

Los valores anteriores inspiran los doce principios del manifiesto. Estos son características que diferencian un proceso ágil de uno tradicional. Los principios son:

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
IV. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

V. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
VII. El software que funciona es la medida principal de progreso.
VIII. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
X. La simplicidad es esencial.
XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

#### 6.2.1.1. Programación Extrema ( Extreme Programming)[7]

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

[7] KROGDAHL, Pal. LUEF, Gottrried. STEINDL, Christoph. Service-oriented agility: Methods for successful Service-Oriented Architecture (SOA) development, Part 1: Basics of SOA and agile methods. 2005.
http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea. A continuación presentaremos las características esenciales de XP organizadas en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas.

**Las Historias de Usuario**

Son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

**Roles XP**

Los roles de XP son:

- Programador. El programador escribe las pruebas unitarias y produce el código del sistema.
- Cliente. Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- Encargado de pruebas (Tester). Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- Encargado de seguimiento (Tracker). Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo

real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.

- Entrenador (Coach). Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- Consultor. Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- Gestor (Big boss). Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

**Proceso XP**

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

**Prácticas XP**

La principal suposición que se realiza en XP es la posibilidad de disminuir costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione.

- El juego de la planificación. Existe una relación constante entre el cliente y los desarrolladores. Los clientes definen las historias de usuarios y los tiempo de entrega y los desarrolladores tratan al máximo de cumplir con estas pautas.

- Entregas pequeñas. Se producen versiones de la aplicación que sean completamente operativa, esto produce la sensación de valor a la inversión de una empresa. Una entrega no debería tardar más 3 meses.

- Metáfora. El sistema es definido por una o varias metáforas, que consisten en un dialogo entre el cliente y los desarrolladles que tiene como resultado una idea de cómo debe funcionar la aplicación.

- Diseño simple. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.
- Pruebas. El desarrollo de código esta relacionado a pruebas unitarias. Estas pruebas deben dar como resultado, la satisfacción de las necesidades planteadas por el cliente, antes de desarrollar la aplicación.

- Refactorización (Refactoring). Lo que se busca con esto es lograr que el código no tenga partes repetidas que cumplan las mismas funciones, lo que lo hace más eficiente.

- Programación en parejas. Todo el desarrollo se realiza en parejas, esto tiene como ventaja un mayor conocimiento del código y mayor satisfacción de los desarrolladores.

- Propiedad colectiva del código. Cualquier programador puede cambiar cualquier parte del código en cualquier momento.

- Integración continúa. Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

- 40 horas por semana. Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. El trabajo extra desmotiva al equipo.

- Cliente in-situ. Como se puede apreciar, es una de las principales características de XP, y lo que logra esto es que se encamine el desarrollo de un aplicación a lo que de mayor valor para el negocio que representa el cliente.

- Estándares de programación. Esto con el fin de que el código sea más legible y el conocimiento sobre el desarrollo sea mayor entre los desarrolladores, así como la comunicación entre estos.

El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. Esto se ilustra en la Figura 25,

donde una línea entre dos prácticas significa que las dos prácticas se refuerzan entre sí. A pesar de que la mayoría de estas practicas no son novedosas, el merito que tiene XP es que logra unirlas hasta el punto de que se refuerzan las unas con las otras.

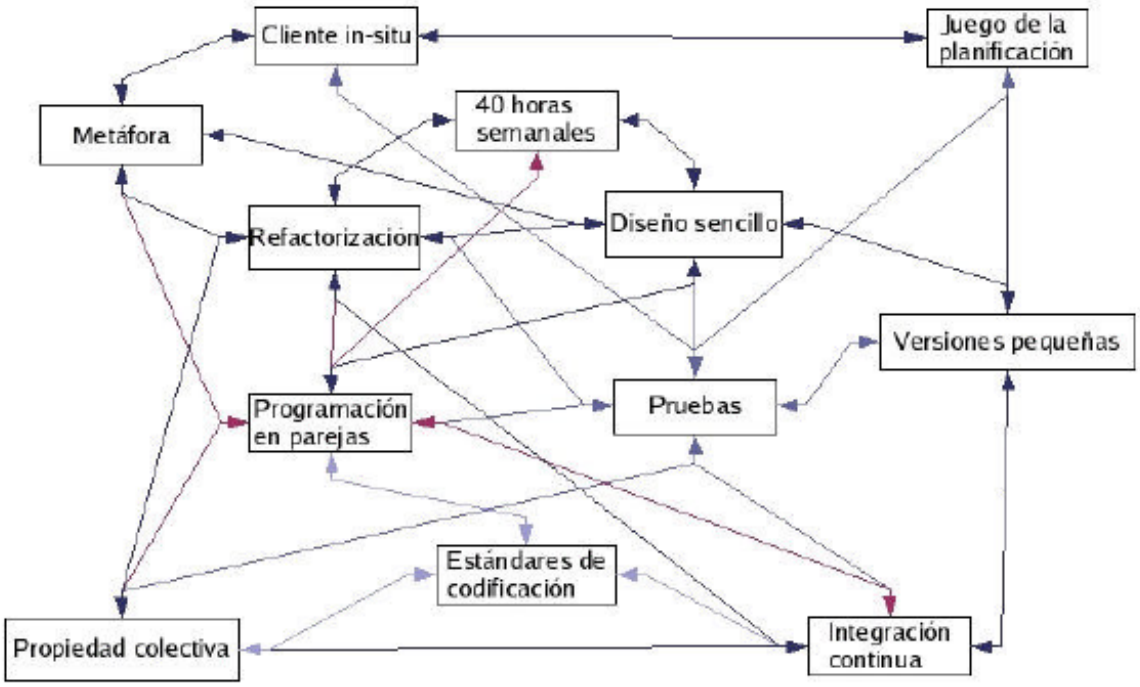


***Figura 25. Las prácticas se refuerzan entre sí***

Tomado de Metodologías ágiles en el desarrollo de software, p6

### *6.2.2.* Otras Metodologías Ágiles

Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios enunciados anteriormente, cada metodología tiene características propias y hace hincapié en algunos

aspectos más específicos. A continuación se resumen otras metodologías ágiles. La mayoría de ellas ya estaban siendo utilizadas con éxito en proyectos reales pero les faltaba una mayor difusión y reconocimiento.

#### *6.2.2.1. SCRUM[8]*

Es una metodología de desarrollo ágil basado en el desarrollo de software mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. Está especialmente indicada para proyectos con un rápido cambio de requisitos.

[8] KROGDAHL, Pal. LUEF, Gottrried. STEINDL, Christoph. Service-oriented agility: Methods for successful Service-Oriented Architecture (SOA) development, Part 1: Basics of SOA and agile methods. 2005.
http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

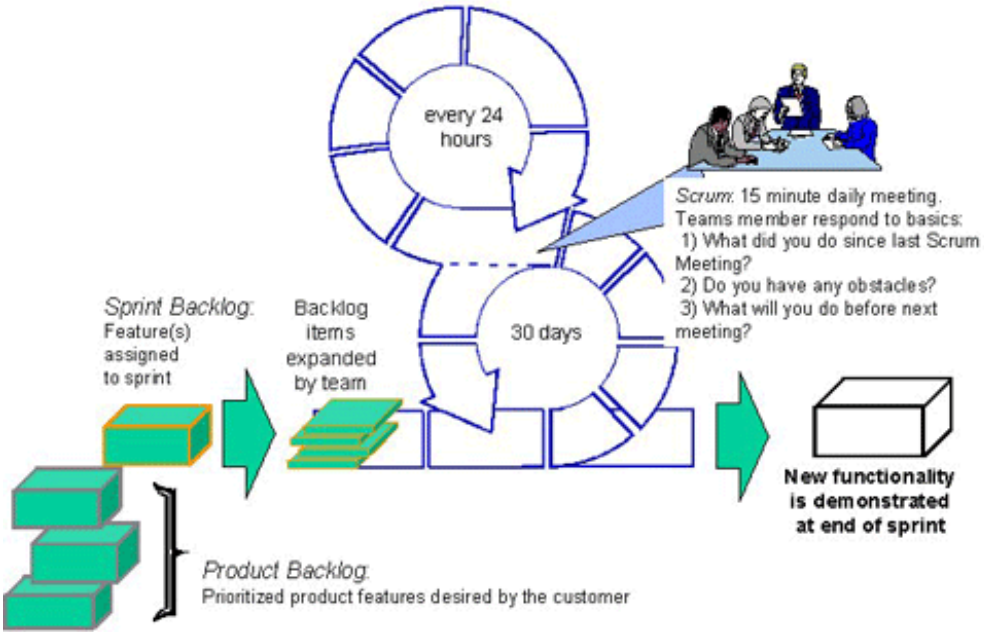


***Figura 26. Metodología SCRUM***

Tomado de http://www.gravitar.biz/index.php/category/metodologias-agiles/

### 6.2.2.2. *CRYSTAL* [9]

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una

[9] KROGDAHL, Pal. LUEF, Gottrried. STEINDL, Christoph. Service-oriented agility: Methods for successful Service-Oriented Architecture (SOA) development, Part 1: Basics of SOA and agile methods. 2005.
http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

clasificación por colores, por ejemplo Extensi Clear (3 a 8 miembros) y Extensi Orange (25 a 50 miembros).

#### *6.2.2.3. Dynamic Systems Development Method (DSDM)*[10]

Define el marco para desarrollar un proceso de producción de software. Las principales características de este modelo son que es un proceso basado en iteraciones y es incremental, y durante todo el desarrollo hay una relación constante entre el cliente y el equipo de desarrollo. Este modelo propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación.

Las tres últimas son iterativas, además de existir realimentación a todas las fases.

#### *6.2.2.4. Adaptive Software Development (ASD)*

Metodología de desarrollo que se basa en procesos iterativos, orientado a los componentes software más que a las tareas y tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda desarrollan las características y finalmente en la tercera se revisa su calidad, y se entrega al cliente. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

---

[10] KROGDAHL, Pal. LUEF, Gottrried. STEINDL, Christoph. Service-oriented agility: Methods for successful Service-Oriented Architecture (SOA) development, Part 1: Basics of SOA and agile methods. 2005.
http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

#### *6.2.2.5. Desarrollo Manejado Por Rasgos (FDD)*

Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software.

Los 5 procesos de FDD se describen a continuación. Los primeros tres se hacen al principio del proyecto.

- Desarrollar un Modelo Global
- Construir una Lista de los Requerimientos
- Planear por requerimientos
- Diseñar por requerimientos
- Construir por requerimientos

Los últimos dos se hacen en cada iteración. Cada proceso se divide en tareas y se da un criterio de comprobación.

Los desarrolladores entran en dos tipos, dueños de clases y programadores jefe.

Los programadores jefe son los desarrolladores más experimentados. A ellos se les asignan requerimientos a construir. Sin embargo ellos no los construyen solos. Sólo identifican qué clases se involucran en la implantación de un requerimiento y juntan a los dueños de dichas clases para que formen un equipo para desarrollar ese requerimiento. El programador jefe actúa como el coordinador y diseñador líder, mientras los dueños de clases hacen gran parte de la codificación del requerimiento.

### 6.2.2.6. Lean Development (LD) [11]

Este modelo tiene como base la eliminación de los desechos que aparecen en la codificación, para lograr una mayor eficiencia en el desarrollo y en la aplicación.

Otros aspectos esenciales de Lean Manufacturing son la relación participativa con el empleado y el trato que le brinda la compañía, así como una especificación de principios, disciplinas y métodos iterativos, adaptativos, auto-organizativos e interdependientes en un patrón de ciclos de corta duración que tiene algo más que un aire de familia con el patrón de procesos de los Métodos Ágiles.

LD se inspira en doce valores centrados en estrategias de gestión:

1. Satisfacer al cliente es la máxima prioridad.
2. Proporcionar siempre el mejor valor por la inversión.
3. El éxito depende de la activa participación del cliente.
4. Cada proyecto LD es un esfuerzo de equipo.
5. Todo se puede cambiar.
6. Soluciones de dominio, no puntos.
7. Completar, no construir.
8. Una solución al 80% hoy, en vez de una al 100% mañana.
9. El minimalismo es esencial.
10. La necesidad determina la tecnología.
11. El crecimiento del producto es el incremento de sus prestaciones, no de su tamaño.
12. Nunca empujes LD más allá de sus límites.

[11] KROGDAHL, Pal. LUEF, Gottrried. STEINDL, Christoph. Service-oriented agility: Methods for successful Service-Oriented Architecture (SOA) development, Part 1: Basics of SOA and agile methods. 2005.
http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

### 6.2.2.7. Lean Software Development (LSD) [12]

Metodología basa en identificar y eliminar los residuos a través de la mejora continua, y de reducir los defectos y tiempo de ciclo.

Un punto clave a observar es que el LSD no es un método de desarrollo en sí, como XP, DSDM, SCRUM, y otros, sino que más bien ofrece una serie de principios fundamentales que deben aplicarse para mejorar el desarrollo de software. A continuación los 6 principios básicos que rigen la metodología LSD:

**Principio 1: Eliminar los residuos**

La eliminación de los residuos es el principio fundamental. El primer paso para la eliminación de los residuos es aprender a identificarlos (todo lo que no brinde mas funcionalidades, mejores tiempos de respuesta y efectividad del código). El segundo paso consiste en descubrir las mayores fuentes de los desechos y eliminarlos. Estos son casi siempre características definidas a principios de especificaciones que podría nunca ser utilizado después que la solución está terminada y entregada.

**Principio 2. Ampliar el aprendizaje**

Con esto lo que se busca es lograr la sincronización entre los miembros del grupo, para lograr aumentar el conocimiento y las fuentes de aprendizaje.

**Principio 3. Decida lo más tarde posible**

La manera más fácil de hacer grandes errores es la de ver los detalles detalle demasiado rápido.

---

[12] KROGDAHL, Pal. LUEF, Gottrried. STEINDL, Christoph. Service-oriented agility: Methods for successful Service-Oriented Architecture (SOA) development, Part 1: Basics of SOA and agile methods. 2005.
http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

La idea es que analicemos todas las situaciones y las consecuencias que puede tener una decisión a la hora de desarrollar una aplicación, con el fin de toma r la decisión que más favorezca al cliente y al grupo de desarrollo.

**Principio 4. Cumplir con la mayor rapidez posible**

Entregar los avances además de completamente funcionales en el menor tiempo posible, esto no ayudara también a ver si se están cumpliendo con los requerimientos del usuario y de no ser así a corregir más rápido los errores que se presenten.

**Principio 5. Dar poder al equipo**

La mejor manera de lograr que el grupo de desarrollo este a gusto, es reconociendo que como principales conocedores del proyecto de desarrollo sean ello mismos los que tomen las decisiones relacionadas con este.

**Principio 6. Construir en la integridad**

Trabajar buscando siempre un producto integral y que tenga un equilibrio efectivo entre la flexibilidad, la mentalidad, la eficiciencia y la receptividad.

### *6.2.3. Agile Modeling*

AM es una estrategia de modelado (de clases, de datos, de procesos) pensada para contrarrestar la sospecha de que los métodos ágiles no modelan y no documentan. Lo que busca es lograr un modelado de una manera más efectiva y ágil.

Los principales objetivos de AM son:

1. Definir y mostrar de qué manera se debe poner en práctica una colección de valores, principios y prácticas que conducen al modelado de peso ligero.

2. Enfrentar el problema de la aplicación de técnicas de modelado en procesos de desarrollo ágiles.
3. Enfrentar el problema de la aplicación de las técnicas de modelado independientemente del proceso de software que se utilice.

Los valores de AM incluyen a los de XP: comunicación, simplicidad, feedback y coraje, añadiendo humildad. Una de las mejores caracterizaciones de los principios subyacentes a AM está en la definición de sus alcances:

1. AM es una actitud, no un proceso extensible. Comprende una colección de valores a los que los modeladores ágiles adhieren, principios en los que creen y prácticas que aplican. Describe un estilo de modelado; no es un recetario de cocina.
2. AM es suplemento de otros métodos. El primer foco es el modelado y el segundo la documentación.
3. AM es una tarea de conjunto de los participantes. No hay “yo” en AM.
4. La prioridad es la efectividad. AM ayuda a crear un modelo o proceso cuando se tiene un propósito claro y se comprenden las necesidades de la audiencia; contribuye a aplicar los artefactos correctos para afrontar la situación inmediata y a crear los modelos más simples que sea posible.
5. AM es algo que funciona en la práctica, no una teoría académica. Las prácticas han sido discutidas desde 2001 en comunidad (http://www.agilemodeling.com/feedback.htm).
6. AM no es una bala de plata.
7. AM es para el programador promedio, pero no reemplaza a la gente competente.
8. AM no es un ataque a la documentación. La documentación debe ser mínima y relevante.
9. AM no es un ataque a las herramientas CASE.
10. AM no es para cualquiera.

Los principios de AM incluyen:

1. **Presuponer simplicidad**. La solución más simple es la mejor.
2. **El contenido es más importante que la representación**. Pueden ser notas, pizarras o documentos formales. Lo que importa no es el soporte físico o la técnica de representación, sino el contenido.
3. **Abrazar el cambio**. Aceptar que los requerimientos cambian.
4. **Habilitar el esfuerzo siguiente.** Dar garantía de que el sistemas desarrollado acepta por completo cambios que pueda tener.
5. **Todo el mundo puede aprender de algún otro**. Tener en cuenta que el conocimiento es mejor compartirlo para lograr mejores resultados.
6. **Cambio incremental**. No esperar hacerlo bien la primera vez.
7. **Conocer tus modelos**. Saber cuáles son las fortalezas y las debilidades.
8. **Adaptación local**. Producir solo lo que se esta pidiendo.
9. **Maximizar la inversión del cliente**. Lograr que el cliente se involucre por completo con el desarrollo.
10. **Modelar con un propósito.** Identificar que se busca con lo que se esta desarrollando.
11. **Modelos múltiples.** Mas de un paradigma relacionados entre si.
12. **Comunicación abierta y honesta.** Comunicación abierta entre los participantes del desarrollo.
13. **Trabajo de calidad.** Buscar lograr las cosas de la mejor manera.
14. **Realimentación rápida.** No esperar que sea demasiado tarde.
15. **El software es el objetivo primario.** Debe ser de alta calidad y coincidir con lo que el usuario espera.
16. **Viajar ligero de equipaje.** No crear más modelos de los necesarios.
17. **Trabajar con los instintos de la gente**. Aceptar supuestos.

Lo mejor de AM son su conjunto de prácticas, que a pesar de ser guía no son de riguroso cumplimiento, estos son:

1. Colaboración activa de los participantes.

2. Aplicación de estándares de modelado.
3. Aplicación adecuada de patrones de modelado.
4. Aplicación de los artefactos correctos.
5. Propiedad colectiva de todos los elementos.
6. Considerar la verificabilidad.
7. Crear diversos modelos en paralelo.
8. Crear contenido simple.
9. Diseñar modelos de manera simple.
10. Descartar los modelos temporarios.
11. Exhibir públicamente los modelos.
12. Formalizar modelos de contrato.
13. Iterar sobre otro artefacto.
14. Modelo en incrementos pequeños.
15. Modelar para comunicar.
16. Modelar para comprender.
17. Modelar con otros.
18. Poner a prueba con código.
19. Reutilizar los recursos existentes.
20. Actualizar sólo cuando duele.
21. Utilizar las herramientas más simples (CASE, o mejor pizarras, tarjetas, post-its).

Como AM debe ser utilizado como complemento de otras metodologías, depende de la metodología utilizada el tamaño del equipo, los roles, la duración de iteraciones, la distribución de trabajo y la criticidad.

### *6.2.4.* Programación Pragmática (PP) [13]

La Programación Pragmática no tiene proceso, fases, roles distintos ó productos de trabajo. Este MA cubre, sin embargo, la mayoría de las mejores prácticas de programación. Esta contiene un grupo de filosofías tan globales que pueden ser utilizadas en cualquier proceso de desarrollo.

La filosofía está definida en 6 puntos:

1. Tome las responsabilidades para usted, se debe tener mas en cuenta las soluciones que buscar excusas.
2. No diseñe o codifique mal, corrija cualquier error de código o planee corregirla rápido.
3. Tome un rol activo para introducir cambios donde se vea que son necesarios.
4. Haga software que satisfaga a su cliente, pero sepa cuándo parar.
5. Constantemente amplíe su conocimiento.
6. Mejore sus habilidades de comunicación.

El enfoque de la Programación Pragmática respecto del testing consiste en probar el código que está siendo implementado sobre el código real, y de forma automática. La idea es no repetir pruebas en códigos que las han pasado.

Recomienda conservar las especificaciones a un nivel razonable de detalle. La PP demuestra prácticas de software simples, responsables y disciplinadas. Las prácticas para la PP son definidas desde el punto de vista de un programador, de

[13] KROGDAHL, Pal. LUEF, Gottrried. STEINDL, Christoph. Service-oriented agility: Methods for successful Service-Oriented Architecture (SOA) development, Part 1: Basics of SOA and agile methods. 2005.
http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630

manera independiente a como desarrolle, para evitar errores típicos en la codificación y en el diseño y errores de comunicación en el grupo de trabajo.

Estas metodologías intentan resolver las siguientes preguntas para un Proyecto:

- ¿Quien? (roles dentro de un proyecto).
- ¿Que? (procesos, actividades y entregables dentro de un proyecto).
- ¿Cuando? (plan de un proyecto, oportunidad, reglas de decisión).
- ¿Como? (como se asignan roles, como se realizan actividades, herramientas).

Al momento de seleccionar una metodología hay que tener las siguientes consideraciones:

*"no hay una única tecnología mejor que todas, ya que algunas se adaptan mejor que otras a una Empresa u organización, y esto depende de la tolerancia al riesgo de la empresa, su nivel de madurez, su tamaño, etc. Todas las metodología manejan las grandes etapas del ciclo de vida de un proyecto, lo que cambia entre ellas es el nivel de disciplina"*.

Otro aspecto a tomar en cuenta es que toda metodología se debe adaptar a la Organización, e incluso se pueden combinar las metodologías porque generalmente no son excluyentes, logrando una metodología mas ajustada a la realidad de la Organización.

### 6.3. Tabla comparativa de Metodologías ágiles

| METODOLOGÍA | CARACTERÍSTICAS | LIMITACIONES | FORTALEZAS | APLICACIONES |
|---|---|---|---|---|
| XP | Parejas de desarrollo, retroalimentación | Diseños simples | Simplicidad en las soluciones, Refactorización, | Proyectos cambiantes |

| | | | | |
|---|---|---|---|---|
| | continua, | | Cliente in - situ | |
| SCRUM | Desarrollo iterativo, Reuniones diarias | Tiempo | Aplicaciones funcionales en cada iteración | Proyectos cambiantes |
| | Capacitación constante, grupos de desarrollo grandes | Costo | Políticas de desarrollo establecidas, Unicidad en el código | Grandes proyectos |
| DSDM | Desarrollo iterativo, retroalimentación continua | Costo | Solo se aplica si es viable | Proyectos cambiantes |
| ASD | Desarrollo iterativo | Pruebas al final | Se corrigen los errores desde su origen | Proyectos cambiantes |
| FDD | Desarrollo iterativo, procesos divididos en tareas | Separación por nivel de programación | Tiempo de entrega de iteraciones | Proyectos cambiantes |
| LD | Se elimina lo inútil, Mejoras constantes | Minimalismo, limitado en el desarrollo de aplicaciones | Satisfacción del cliente, Reutilización de código | Proyectos cambiantes |
| LSD | Se elimina lo inútil | Minimalismo | Supervisión de procesos de desarrollo/ rapidez | Proyectos cambiantes |

## 6.4. Contexto de SOA

En esta parte presentaremos una serie de actividades para la construcción de un modelo SOA mediante el análisis y diseño de las principales actividades y técnicas que se necesitan para crearla

Como ejemplo, vamos a asumir que usted está trabajando en un proyecto y el objetivo es migrar una parte de la línea de negocio de banca, que tiene un autoservicio, sistema de contabilidad, a una SOA.

Con el fin de migrar a una SOA, necesita algunos elementos adicionales que van más allá de los servicios de modelado. Estos incluyen:

- Adopción y modelos de madurez. ¿Dónde se encuentra su empresa en la escala relativa de la madurez en la adopción de SOA y de Servicios Web? Cada nivel diferente de adopción tiene sus propias necesidades.

- Evaluaciones. ¿Tiene algunos pilotos? ¿Cuán buena es su arquitectura resultante? En caso de que usted continúe en la misma dirección ¿Has pensado en todo lo que necesita para considerar?

- Estrategia y planificación de actividades. ¿Cómo planea migrar a una SOA? ¿Cuáles son los pasos, herramientas, métodos, tecnologías, estándares, y la formación que necesita para tener en cuenta? ¿Cuál es su plan de trabajo y la visión, y ¿cómo llegar? ¿Cuál es el plan?

- Gestión de los asuntos públicos. Cada servicio debe ser creado con la intención de aportar valor al negocio de alguna manera. ¿Cómo gestionar este proceso?

- La aplicación de "buenas prácticas". ¿Cuáles son algunas probado la forma de aplicar la seguridad, la garantía de ejecución, el cumplimiento de normas de interoperabilidad, el diseño para el cambio?

### *6.4.1. Caso Programación Extrema aplicado en SOA*

Además de la identificación, descripción, y las realizaciones descritas en la presente monografía, el modelado de orientación a servicios incluye las técnicas necesarias para el despliegue, la supervisión, la gestión, la gestión de los asuntos públicos y la obligación de apoyar la plena ciclo de vida SOA.

Es útil también la implementación de las metodologías al momento de aplicar SOA en una compañía, a continuación un ejemplo de cómo emplear la metodología ágil de desarrollo programación extrema (XP) en el desarrollo de una aplicación SOA, teniendo en cuenta inicialmente los siguientes aspectos:

- Identificar las piezas de software de granularidad gruesa llamadas servicios.
- Identificar los procesos de negocio.
- Identificar la secuencia de ejecución de los distintos servicios identificados en una "orquestación de servicios" o "coreografías de servicios" según si el control del proceso de Negocio se encuentra en la propia Organización o es compartido por varias Organizaciones que acuerdan sobre la realización de dicho proceso.

En base a los aspectos definidos, SOA implementa las metodologías ágiles en especial XP de la siguiente forma:

#### *6.4.1.1. Exploración del negocio*

El propósito de la disciplina Modelado del Negocio es:

- Asegurar que clientes, usuarios finales, desarrolladores y otros involucrados tengan un conocimiento común de la Organización.

- Derivar los requerimientos del sistema de software que son necesarios para apoyar a la Organización objetivo.

- Comprender como el sistema de software a implantar se inscribe en la Organización

El esfuerzo de modelado del Negocio puede tener distintos alcances dependiendo del contexto y las necesidades.

En este caso se está en el escenario de obtener un mapa de la Organización y sus procesos para ganar un mejor entendimiento de los requerimientos de la aplicación en desarrollo. El modelado del Negocio es parte entonces del proyecto de Ingeniería de Software y es realizado principalmente durante la Fase Inicial.

Este tipo de esfuerzos en general comienzan como un relevamiento sin intenciones de cambiar la organización, sin embargo, en realidad el desarrollo e implantación de una nueva aplicación siempre puede incluir un nivel de mejora del negocio.

**Evaluar la Organización Objetivo** : Se debe describir el estado actual de la Organización en la cual la aplicación será implantada, en términos de sus procesos actuales, herramientas, competencias de las personas, actitudes de las personas, clientes, competencia, desafíos tecnológicos, problemas y áreas de mejora, identificando claramente los involucrados (stakeholders) en el esfuerzo de modelado del negocio.

**Identificar Procesos de Negocio**: Esta actividad está compuesta de varias sub-actividades que en conjunto permiten identificar los Procesos del Negocio, describiendo los actores participantes y el flujo de ejecución de los mismos en la Organización. Para esto se deben encontrar los actores y Casos de Uso del

Negocio, las reglas del Negocio y los términos que se manejan en el Negocio. Es importante definir los límites del Negocio a ser modelado y quién y qué interactúa con el Negocio, describir los procesos del Negocio y plasmarlos en diagramas de actividad en el Modelo de Casos de Uso del Negocio.

| ACTIVIDAD | ENTRADA | SALIDA | ROL RESPON-SABLE | ROL INVOLUCRADO | FRECU-ENCIA (SEMA-NAS) |
|---|---|---|---|---|---|
| Evaluar la Organización Objetivo | Información sobre la Organización y sus involucrados. | Evaluación de la Organización Objetivo | Analista, Cliente, Consultor, Entrena-dor, Gestor | Arquitecto | 1,2 |
| Identificar procesos del Negocio | Acta de reunión de requerimientos<br><br>Evaluación de la Organización Objetivo | Modelo de Casos de Uso del Negocio<br>o<br>Historias del Usuario<br><br>Glosario | Analista, Cliente, Consultor, Entrena-dor, Gestor | Arquitecto | 1,2,3,4 |

***Tabla 1. Modelado de negocio***

Tomado de[http://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html](http://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html)

### *6.4.1.2. Planificación*

El propósito de la disciplina de Diseño en una SOA agrega los siguientes objetivos a los definidos en el modelo base:

- Identificar y catalogar los servicios necesarios para cumplir con los procesos identificados de la Organización en los Casos de Uso del Negocio o historias de usuario definidas.

- Especificar estos servicios definiendo las interfaces y sus operaciones, así como los componentes que los implementarán y la reutilización de otros servicios y componentes dentro de la Organización.

- Definir la secuencia de invocaciones a servicios necesaria para la ejecución de los procesos del Negocio identificados como Casos de Uso del Negocio en la orquestación de servicios que define la coreografía de servicios.

- Propender a la utilización de un BPMS (Business Process Management System) que permita la definición del Workflow del proceso, y BPML (Business Process Modeling Language) para la orquestación de servicios.

| ACTIVIDAD | ENTRADA | SALIDA | ROL RESPON-SABLE | ROL INVOLUCRADO | FRECUENCIA REALIZA-CIÓN (SEMA-NAS) |
|---|---|---|---|---|---|
| Identificar y categorizar servicios | Descrip-ción de la Arquitec-tura<br><br>Historias del Usuario | Modelo de Servicios | Arquitecto,<br><br>Cliente<br><br>Entrena-dor | Analista<br><br>Especialista Técnico | 3,4,5,6,7,8 |
| Especificar servicios | Modelo de Servicios | Modelo de Servicios | Arquitecto, Cliente, Entrenador | Analista | 3,4,5,6,7,8 |
| Identificar servicios existentes | Evaluación de la Organizaci | Modelo de Servicios | Arquitecto, Cliente, Entrenador | Analista | 3,4,5,6 |

| | | | | | |
|---|---|---|---|---|---|
| | ón Objetivo | | | | |
| Asignar servicios a componentes | Modelo de Servicios<br><br>Modelo de Diseño | Modelo de Servicios<br><br>Modelo de Implementación | Arquitecto, Cliente, Entrenador | Especialista Técnico | 3,4,5,6,7,8,9,10 |
| Definir orquestación de servicios | Modelo de Casos de Uso del Negocio<br><br>Modelo de Servicios | Modelo de Servicios<br><br>Modelo de Implementación | Arquitecto, Cliente, Entrenador | Analista<br><br>Especialista Técnico | 3,4,5,6,7,8,9,10 |

***Tabla 2. Diseño de un Modelo SOA***

Tomado de http://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html

#### *6.4.1.3. Implementación e Iteraciones*

El propósito de la disciplina de Implementación en SOA agrega los siguientes objetivos a los definidos en el modelo base:

- Implementar los componentes proveedores de servicios según lo establecido por la asignación de servicios a componentes.

- Implementar el ligamiento de servicios en las application frontend y en los servicios que llaman a otros servicios, utilizando la estrategia definida en la Organización para hacer este ligamiento.

- Este proceso se realiza no debería tardar más de 3 meses, debido a que es una programación ágil y lo que se busca es producir versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema.

| ACTIVI-DAD | EN-TRADA | SALIDA | ROL RESPO-SABLE | ROL INVOLU-CRADO | FRECUEN-CIA (SEMANAS) |
|---|---|---|---|---|---|
| Implemen-tar servicios | Modelo de Servicios<br><br>Modelo de Implemen-tación | Servicio implement ado (todas las partes requerí-das) | Encargado de Pruebas, Cliente, Gestor, Consultor, Entrenador | ninguno | 5,6,7,8,9,10,1 1,12 |

***Tabla 3. Implementación de un modelo SOA***

Tomado dehttp://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html

#### *6.4.1.4. Verificación*

Esta actividad tiene como objetivo realizar la verificación del producto desarrollado, para así poder detectar fallas en el mismo.

| ACTIVIDAD | ENTRADAS | SALIDAS | ROL RESPONSABLE | ROL INVOLUCRADO |
|---|---|---|---|---|
| Ejecutar las Pruebas | Entorno Prueba<br><br>Prueba Plan de Verificación y Validación<br><br>Plan de Pruebas Verif. de la Iteración | Informe de Verificación de Integración<br><br>Reportes de Pruebas | Encargado de Pruebas,<br><br>Cliente ,<br><br>Entrenador | Implementador |

*Tabla 4. Verificación de Modelo SOA*

Tomado dehttp://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html

# 7. ESTADO DEL ARTE SOA

SOA construye en base a sus principios, sistemas capaces de integrar los más amplios requerimientos, dando como resultado Aplicaciones transcendentales capaces de perdurar y evolucionar a la vanguardia de la tecnología y de los nuevos avances en desarrollo y modelado de software; esto se logra mediante unos cimientos bien definidos como lo son los antecedentes de este paradigma: Sistemas Distribuidos y Servicios Web, el primero aportando la distribución de actividades y el segundo permitiendo que la integración de dichas actividades se realizará mediante un mismo protocolo como lo son HTTP y SOAP(descrito en el capitulo IV).

Es importante destacar el rol que los servicios juegan al momento de diseñar las Arquitecturas Orientada a Servicios, y lo más importante es saber identificarlo, olvidándonos del concepto obsoleto de “Es un componente de software que puede ser invocado remotamente y que puede ser descrito de una manera estándar a través de un archivo WSDL”, ya que nos quedamos cortos puesto que esto significaría que cualquier función que resolviera una suma, resta u otra sería un servicio, y servicio es realmente aquella función que satisface una necesidad de negocio.

Con el fin de controlar el manejo de la información en los diferentes procesos de las empresas, SOA implementa las técnicas en las que encontramos Enterprise Architecture, BPM, OOAD, ETL, WorkFlow Tradicional, ESB y BPM Suite. Enterprise Archirecture se enfoca en todo lo correspondiente a los objetivos de la empresa y sus lineamientos, con el fin de ligar las tecnologías con estos; BPM

busca la eficiencia en las empresas, manejando la sistematización de sus procesos; OOAD utiliza como base la orientación a objetos con el fin de llegar a la identificación de los servicios que se van a prestar; ETL se utiliza para la integración de sistemas a través de datos o repositorios de datos. Para lograr esto ETL utiliza tres etapas que son extraer, que es la etapa en la que se obtiene los datos que están en un repositorio, transformar, que pasa la información a una forma mas entendible por el receptor y cargar, que se encarga de colocar en el destino los datos transformados; Workflow Tradicional da seguimiento a todos los procesos de negocio, ya que por medio de este se conoce en que estado y en que etapa del flujo se en cuenta una actividad que siempre es humana; ESB tiene la misma base de workflow, siendo que esta se enfoca en actividades del sistema; BPM Suit es una mezcla entre workflow tradicional y ESB, ya que esta maneja en su flujo tanto actividades del sistema como actividades humanas.

Dentro de SOA se han definido estándares para procesos repetitivos con el fin de agilizar el desarrollo de aplicaciones conocidos como patrones. De la mano de la identificación de patrones, SOA implementa una metodología ágil de desarrollo orientada a cumplir con objetivos (Satisfacer las necesidades del negocio, definidas como requerimientos), y no con fases; teniendo en cuenta que la utilización del modelado UML y de la definición de casos de uso aún forman un papel fundamental en la etapa de modelado. La utilización de una de las metodologías no es una camisa de fuerza, pero la correcta implementación disminuye los riesgos al momento de desarrollar una aplicación, es decir puede garantizar el éxito seguro. Una vez definidos todos estos lineamientos, existen tecnologías que permiten que todo este proceso de negocio este al alcance y el momento indicado para cada uno de los usuarios que hacen parte y dichas tecnologías se basan en protocolos descritos anteriormente como SOAP, UDDI, WS-CDL y WS-BPEL, que en conjunto permiten que la Arquitectura Orientada a Servicio funcione eficazmente.

Al momento de realizar este documento, los conceptos utilizados fueron los más actuales, pero apreciando el rápido avance y evolución de las técnicas, metodologías, tecnologías y patrones se deja claro que puede perderse vigencia en el tiempo por lo este documento es una herramienta útil para futuras investigaciones y desarrollo sobre el tema.

# 8. BENEFICIOS Y RETOS DE SOA

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

Desde el punto de viste de la empresa, SOA brinda soluciones que pueden brindarse a mas de un servicio, lo que hace mas eficaz los procesos del negocio. Las soluciones SOA permiten entre otras cosas:

Mejorar la toma de decisiones. Esto como resultado de la rapidez y la eficiencia en la que se entrega al personal de la organización, la información necesaria para la toma de decisiones.

Mejorar la productividad de los empleados. Un optimo acceso a los sistemas y la posibilidad de corregir lo procedimientos para mejorar el desarrollo y la aplicación de estos, dan valor agregado a las actividades de una organización. Al mismo tiempo el salir de sistemas rígidos y entrar a un tipo de sistema completamente dinámico, mejora la relación entre los usuarios y los sistemas que son los que en definitiva le ayudan a cumplir con servicios específicos.

Potenciar las relaciones con clientes y proveedores. Las ventajas de SOA trascienden las fronteras de la organización. La integración con partners comerciales y la optimización de los procesos de la cadena de suministro son, bajo esta perspectiva, objetivos perfectamente asequibles. Con SOA se puede

conseguir mejorar la capacidad de respuesta a los clientes, habilitando por ejemplo portales unificados de servicios.
SOA contribuye también a documentar el modelo de negocio de la empresa y a utilizar el modelo de negocio documentado para integrar en él y dar respuesta a las dinámicas de cambio que se produzcan y optimizarlo de acuerdo con ellas.

Desde el punto de vista de los departamentos de IT, la orientación a servicios significa una mayor facilidad en la creación y mantenimiento de sistemas, así como una más fácil relación entre los sistemas y los objetivos de una organización.

Aplicaciones más productivas y flexibles. La estrategia de orientación a servicios permite a IT conseguir una mayor productividad sus recursos, ya que les facilita la relación entre una aplicación y futuras aplicaciones, o inclusive aplicaciones anteriores.

Desarrollo de aplicaciones más rápido y económico. Como se manejan estándares, la elaboración o desarrollo de aplicaciones se convierte en algo fácil y económico.

Aplicaciones más seguras y manejables. SOA ofrece una infraestructura y una documentación común, por lo que los desarrollos de servicios se convierten en algo más seguro y gestionables. A medida que van evolucionando las necesidades de la organización, SOA brinda facilidades para añadir servicios y funcionalidades.

**COMO SE RESUELVEN LOS RETOS EN SOA**

Al iniciarse un desarrollo utilizando como base SOA, se debe tener claro que todo lo que esto conlleva no es la solución a todos los males que tiene una organización.
Para que las iniciativas de adopción de SOA tengan un fin satisfactorio, hay que asegurarse de que se cumplen una serie de condiciones indispensables:

Definir claramente los objetivos de negocio. Lo primero que se debe saber es hacia que esta encaminada la organización, esto con el fin de facilitar el establecimiento de los alcances de un desarrollo SOA.

Definir claramente el alcance del proyecto SOA. El objetivo de un desarrollo SOA no debe ser cambiar de manera drástica toda la infraestructura de sistemas con la que se cuneta, sino de establecer puntos claros y específicos a los que se quiere brindar una solución.

Evitar introducir SOA sin motivos reales que lo justifiquen. La implementación de SOA en una organización no debe hacerse por fines tecnológicos, sino buscando solucionar necesidades especificas, para que se tenga éxito en la implementación.

Gestionar el proceso. A la hora de implementar SOA debe nombrarse un encargado de este proyecto, buscando así que exista una persona que revise los avances y gestione el proceso de desarrollo.

# 9. CONCLUSIONES

Cuando pensamos en lo cambiante que es el mundo, después de haber finalizado esta investigación, se nos viene a la mente de inmediato el término SOA cuya visión permite dar un rumbo de cambio constante a las empresas, este cambio se ve representado en la eficacia y eficiencia de los servicios que ofrece.

Importante destacar como los Sistemas Distribuidos y los Servicios Web fue la base fundamental para la aparición de las aplicaciones Orientadas a Servicios, aclarando siempre las diferencias existentes entre Servicios Web y SOA.

Durante la investigación, se identifico que la base fundamental de SOA son los servicios, ya que permiten identificar cuales son las necesidades que se tienen que satisfacer, teniendo en cuenta cuales son los mecanismos que permiten identificar cuando estoy ofreciendo un servicio o no.

Como ente fundamental de esta investigación, se analizo cada una de las metodologías de desarrollo, patrones de diseño, tecnología, técnicas, elementos que permiten desarrollar una aplicación SOA, lo cual deja de manifiesto las diferentes metodologías que existen, las cuales no son camisas de fuerza para “x” o “y” proyecto, pero si garantizan que su correcta aplicación nos pueden llevar al éxito seguro.

Para finalizar, SOA es una buena estrategia para una empresa que busca posicionarse en el mercado, dejamos este documento que le permitirá en base a

las metodologías expuestas, adquirir una idea sólida de cual metodología aplicar, como aplicarla y cuales son todos elementos que se deben tener en cuenta.

# 10. GLOSARIO

**Coreografía**: puede entenderse como un proceso público y no ejecutable: es público porque define el comportamiento común y globalmente visible entre los diferentes participantes en una interacción; por otro lado es no ejecutable porque no está pensado para ser llevado a cabo, sino para actuar como un protocolo de negocio que dicta reglas de interacción que deben ser cumplidas por las entidades participantes.

**Granularidad:** Es el alcance del dominio que el servicio completo implementa, también aplica al alcance del dominio que cada método implementa dentro de la interfaz.

**HTTP (HyperText Transfer Protocol):** define la sintaxis y la semántica que utilizan los elementos software de la arquitectura Web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. Es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones Web necesita frecuentemente mantener estado. Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente.

**Orquestación**: se refiere a un proceso de negocio ejecutable que puede interactuar tanto con servicios Web internos como con externos. Las interacciones ocurren al nivel de mensajes, e incluyen la lógica de negocio y el orden de ejecución de las tareas. La orquestación siempre representa control de una de las partes.

**SOAP** (Simple Object Access Protocol): es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

**Servicio**: Componente que representa una unidad lógica de un proceso de negocio completo, al cual es posible acceder programáticamente desde contextos independientes a través de una interfaz abierta y documentada.

**Ubicuidad**: Don o poder de estar en todas partes en un momento dado, aunque no en todas partes en todos los momentos.

**XML**: es un conjunto de reglas (también se las podría pensar como líneas de guía o convenciones) para diseñar formatos de texto que permitan estructurar los datos. XML no es un lenguaje de programación, y no hace falta ser un programador para usarlo o aprenderlo. XML facilita a la computadora la tarea de generar datos, leerlos, y asegurar que su estructura no es ambigua. XML evita las fallas comunes en diseño de lenguajes: es extensible, independiente de la plataforma, y soporta internacionalización y localización. XML cumple totalmente con el standard Unicode.

**W3C:** es un consorcio internacional donde la organización miembro, personal a tiempo completo y el público en general, trabajan conjuntamente para desarrollar estándares Web. La misión del W3C es guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web.

# BIBLIOGRAFIA


- AALBERS, Huibert. Plan para la implementación exitosa de SOA, 2006, 1p.

- ARSANJANI, Ali. “Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA”. 28 de Enero de 2008. **http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/**

- BETANCOURT, Javier. Fundamentos de Arquitectura Orientada a Servicios. EN CONGRESO TECNOLOGY WORLD MEXICO (1ª, 2006, Ciudad de México). Ponencias del primer congreso Tecnology World México. Ciudad de México: Progress Software.

- CANÓS, José H. LETELIER, Patricio. PENADÉS, María Carmen. Metodologías ágiles en el desarrollo de software. 2005, 1p. Documento de Investigación. (Ingeniería de Sistemas). Universidad Politécnica de Valencia.

- COLÁN, Mark. “Service-Oriented Architecture expands the vision of Web services, Part 1”. 28 de Diciembre de 2007. **http://www-128.ibm.com/developerworks/webservices/library/ws-soaintro.html**

- ERL, Thomas. SOA: Principles of Service Design. Prentice Hall. 2008, 445p. ISBN 0-13-234482-3.

- FIELDING, Roy Thomas. “Architectural styles and the design of network-based software architectures”. Tesis doctoral, University of California, Irvine, 2000.
- Garcia, Juan Pablo. “BizTalk Messaging, Implementación del Patrón MESSAGE BROKER”. 14 de Marzo de 2008. **www.microsoft.com**

- KROGDAHL, Pal. LUEF, Gottrried. STEINDL, “Christoph. Service-oriented agility: Methods for successful Service-Oriented Architecture (SOA) development, Part 1: Basics of SOA and agile methods”. 11 de Enero de 2008. **http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630**

- MICROSOFT CORPORATION. “La Arquitectura Orientada a Servicios de Microsoft Orientada al mundo real”. 15 de Enero de 2008. **www.microsoft.com**

- NICOLAI M. JOSUTTIS. Soa in Practice. The Arto of Distributed System Design. O’REILLY. Agosto, 2007. ISBN-10: 0-596-52955-4.

- REYNOSO, Billy. “Arquitectura Orientada a Servicios (SOA)”. 19 de Enero de 2008. **http://www.microsoft.com/**

- RICO GARCIA, Jorge Alejandro. GOMEZ OTERO, Jhon Jairo. Documento de estado de Arte en SOA y Cálculo PI. 2007, 1p.

- RIVERA, Ignacio. Patrones Arquitectónicos Layers. Octubre, 2006, 5p. Universidad Nacional de Río Cuarto. Departamento de Computación, Facultad de Ciencias Exactas Físico- Químicas y Naturales.

- Soluciones Java, SOA, y BPM. “Orquestadores y SOA”. 01 de Abril de 2008. **http://soaagenda.com/**

- VASQUEZ ROMERO, William. Mecanismo de Control de Acceso en Web Services. Santafé de Bogotá, 2004, 14p. Trabajo de grado (Ingeniero de Sistemas). Pontificia Universidad Javeriana. Facultad de Ingenierías. Carrera de Ingeniería de Sistemas.

- ZIMMERMANN, Olaf. “Elements of Service-Oriented Analysis and Design”. 13 de Diciembre de 2007. **http://www-128.ibm.com/developerworks/library/ws-agile1/?ca=dnt-630**