Arquitectura de software. Arquitectura orientada a servicios

Architecture of software. Services oriented architecture

Yanet Espinal Martín

Universidad de las Ciencias Informáticas

yanete@uci.cu

Resumen

Dentro del proceso de desarrollo de software de un sistema uno de los temas importantes a tratar es la arquitectura de software, es la parte de la ingeniería de software que se dedica al estudio, análisis y descripción para formalizar el esquema global de un sistema, poniendo énfasis en el estudio de las interacciones entre sus elementos básicos, denominados ***componentes***.

La capacidad para responder rápidamente ante los cambios y optimizar los procesos de negocio es un factor clave para la competitividad y el crecimiento de las organizaciones. La Arquitectura Orientada a Servicios (SOA, Service Oriented Architecture) es una filosofía de diseño que permite un mejor alineamiento de las Tecnologías de Información (TI) con las necesidades de negocio, permitiendo a los usuarios de forma más rápida adaptarse adecuadamente a las presiones del mercado.

El presente trabajo realiza un pequeño estudio general de la arquitectura de software haciendo énfasis en la arquitectura orientada a servicio.

Palabras clave: Arquitectura, Ingeniería de Software, Software.

Abstract

Within the process of software development of a system one of the important issues to be addressed is the software architecture, is part of software engineering that is dedicated to the study, analysis and description to formalize the outline of an overall system, putting emphasis on the study of interactions among its basic elements, called components.

The ability to respond rapidly to change and optimize business processes is a key factor for competitiveness and growth of organizations. The Service Oriented Architecture (SOA, Service Oriented Architecture) is a design philosophy that allows a better alignment of Information Technology (IT) with business needs, allowing users to more quickly adapt adequately to the pressures market.

This work makes a small study of the overall software architecture with an emphasis on the service oriented architecture.

Key words*: Architecture, Software, Software Engineering.*

Introducción

Breve Historia de la Arquitectura de Software

Cada vez que se narra la historia de la arquitectura de software o de la ingeniería de software, se reconoce que en un principio, hacia 1968, Edsger Dijkstra, propuso que se estableciera una estructuración correcta de los sistemas de software antes de lanzarse a programar, escribiendo código de cualquier manera (Dijkstra Enero de 1983). Dijkstra, quien sostenía que las ciencias de la computación eran una rama aplicada de las matemáticas y sugería seguir pasos formales para descomponer problemas mayores, fue uno de los introductores de la noción de sistemas operativos organizados en capas que se comunican sólo con las capas adyacentes y que se superponen "como capas de cebolla". Inventó o ayudó a precisar además docenas de conceptos: el algoritmo de camino más corto, los stacks, los vectores, los, los abrazos mortales. De sus ensayos arranca la tradición de hacer referencia a "niveles de abstracción" que ha sido tan común en la arquitectura subsiguiente.

Aunque Dijkstra no utiliza el término arquitectura para describir el diseño conceptual del software, sus conceptos sientan las bases para lo que luego expresarían Niklaus Wirth (Wirth Abril de 1971) como stepwise refinement y DeRemer y Kron (Kron 1976) como programming-in-the large o programación en grande, ideas que poco a poco irían decantando entre los ingenieros primero y los arquitectos después.

En la conferencia de la NATO de 1969, un año después de la sesión en que se fundara la ingeniería de software, P. I. Sharp formuló estas sorprendentes apreciaciones comentando las ideas de Dijkstra:

"Pienso que tenemos algo, aparte de la ingeniería de software, algo de lo que hemos hablado muy poco pero que deberíamos poner sobre el tapete y concentrar la atención en ello, es la cuestión de la arquitectura de software. La arquitectura es diferente de la ingeniería, ejemplo de lo que quiero decir, echemos una mirada a OS/360. Partes de OS, si vamos al detalle, han utilizado técnicas que hemos acordado constituyen buena práctica de programación. La razón de que OS sea un amontonamiento amorfo de programas es que no tuvo arquitecto. Su diseño fue delegado a series de grupos de ingenieros, cada uno de los cuales inventó su propia arquitectura. Y cuando esos pedazos se clavaron todos juntos no produjeron una tersa y bella pieza de software (Sharp 27 al 31 de Octubre de 1969)."

Una novedad importante en ***la década de 1970*** fue el advenimiento del diseño estructurado y de los primeros modelos explícitos de desarrollo de software. Estos modelos comenzaron a basarse en una estrategia más orgánica, evolutiva, cíclica, dejando atrás las metáforas del desarrollo en cascada que se inspiraban más bien en la línea de montaje de la ingeniería del hardware y la manufactura. Poco a poco el diseño se fue independizando de la implementación, y se forjaron herramientas, técnicas y lenguajes de modelado específicos.

En la misma época, otro precursor importante, David Parnas, demostró que los criterios seleccionados en la descomposición de un sistema impactan en la estructura de los programas y propuso diversos principios de diseño que debían seguirse a fin de obtener una estructura adecuada. Parnas desarrolló temas tales como módulos con ocultamiento de información, estructuras de software y familias de programas (Parnas Diciembre de 1972), enfatizando siempre la búsqueda de calidad del software.

En 1972, **Parnas** publicó un ensayo en el que discutía la forma en que la modularidad en el diseño de sistemas podía mejorar la flexibilidad y el control conceptual del sistema, acortando los tiempos de desarrollo. Introdujo entonces el concepto de ocultamiento de información (information hiding), uno de los principios de diseño fundamentales en diseño de software aún en la actualidad. En la segunda de las descomposiciones que propone Parnas comienza a utilizarse el ocultamiento de información como criterio. Cada módulo deviene entonces una caja negra para los demás módulos del sistema, los cuales podrán acceder a aquél a través de interfaces bien definidas, en gran medida invariables.

En 1975, Brooks, diseñador del sistema operativo OS/360 y Premio Turing 2000, utilizaba el concepto de arquitectura del sistema para designar "la especificación completa y detallada de la interfaz de usuario" y consideraba que el arquitecto es un agente del usuario (Jr. 1975). También distinguía entre arquitectura e implementación; mientras aquella decía qué hacer, la implementación se ocupa de cómo.

Como escriben **Clements y Northrop** en esta época (Northrop Febrero de 1996) en todo el desenvolvimiento ulterior de la disciplina permanecería en primer plano esta misma idea: la estructura es primordial (structure matters), y la elección de la estructura correcta ha de ser crítica para el éxito del desarrollo de una solución. ***"La elección de la estructura correcta sintetiza, como ninguna otra expresión, el programa y la razón de ser de la AS"***.**En la década de 1980,** fue surgiendo un nuevo paradigma, el de la programación orientada a objetos. Paralelamente, hacia fines de la década de 1980 y comienzos de la siguiente, la expresión *arquitectura de software* comienza a aparecer nuevamente en la literatura para hacer referencia a la configuración morfológica de una aplicación.

El primer estudio en que aparece la expresión ***"arquitectura de software"*** en el sentido en que hoy lo conocemos es sin duda el de Perry y Wolf; **ocurrió en 1992**, aunque el trabajo se fue gestando desde 1989. En él, los autores proponen concebir la AS por analogía con la arquitectura de edificios, una analogía de la que luego algunos abusaron (WWISA 1999), otros encontraron útil y para unos pocos ha devenido inaceptable (Reed 2001).

No es hasta el lanzamiento del libro ***"An Introduction to Software Architecture",*** de Mary Shaw y David Garlan, en el año 1994, donde plantean, que debido al aumento del tamaño y complejidad de los productos de software, el problema principal no radica ya, en los algoritmos y las estructuras de datos, sino que se dirige a la organización de los componentes que conforman el sistema, introduciendo así, la necesidad de la creación de la AS, como disciplina científica, cuyo objeto de estudio no es más que la determinación de un conjunto de paradigmas que establezcan una organización del sistema a alto nivel, la interrelación entre los distintos componentes que lo conforman y los principios que orientan su diseño y evolución. (David Garlam, 1994).

*"**La década de 1990**,* fue la década de la ***"arquitectura de software"***, dando cumplimiento a las profecías de Perry y Wolf, fue sin duda la década de consolidación y diseminación de la AS en una escala sin precedentes. Las contribuciones más importantes surgieron en torno del instituto de ingeniería de la información de la Universidad Carnegie Mellon (CMU SEI). En la misma década, demasiado pródiga en acontecimientos, surge también la programación basada en componentes, que en su momento de mayor impacto impulsó a algunos arquitectos mayores, como Paul Clements (Clements Enero de 1996.), a afirmar que la AS promovía un modelo que debía ser más de integración de componentes pre-programados que de programación.

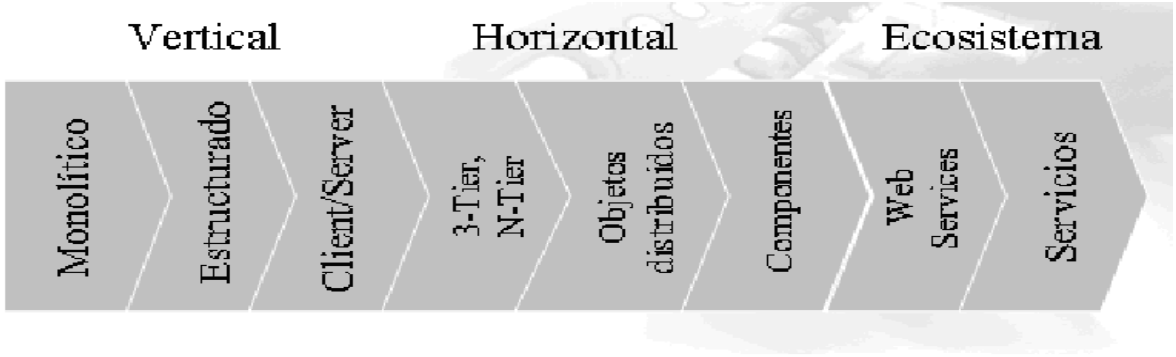

Fig. 1. Evolución de la Arquitectura de Software.

En el transcurso de los años, la complejidad y tamaño de los sistemas de software se fue incrementando de manera espectacular. La capacidad para responder rápidamente ante los cambios y optimizar los procesos de negocio es un factor clave para la competitividad y el crecimiento de las organizaciones.

Cada vez más las organizaciones dependen de su infraestructura de TI para alcanzar sus objetivos, pero en un entorno competitivo como el actual, aprovechar las oportunidades de negocio exige moverse con rapidez. Sin embargo, con frecuencia las Tecnologías de Información no permiten estas respuestas rápidas ni disponen de la flexibilidad necesaria para competir de forma efectiva. Un alto porcentaje de las ineficiencias organizativas tienen un mismo origen: el predominio de procesos manuales con un nivel de error elevado, sistemas ineficaces para compartir la información en el seno de la organización; las ineficiencias propias del servicio a clientes, etc.

En la raíz de todas estas deficiencias está la información. No como un problema de escasez de información sino de la imposibilidad de presentar la información de forma sencilla y útil a los usuarios y directivos de una manera coherente y sistemática. En última instancia, esto se debe a que las aplicaciones no compartían informaciones entre ellas y, por consiguiente, no pueden aportar una visión general de los procesos de negocio cuando éstos abarcan varias áreas funcionales.

Lo que se necesita es una herramienta basada en estándares para integrar sistemas y aplicaciones heterogéneos sobre una serie de plataformas y protocolos de comunicación heterogéneos, así como una metodología bien establecida para lograr el nivel óptimo de integración, de manera que la infraestructura subyacente facilite los cambios posteriores que puedan surgir como respuesta a

la evolución en las necesidades de la empresa.

Así surgió La Arquitectura Orientada a Servicios (SOA, Service Oriented Architecture) fue descrita por primera vez por Gartner en 1996.

Como suceso importante, fue el lanzamiento de la tesis de Roy Fielding, en el año 2000, la cual presenta el modelo REST, quien establece definitivamente los modelos orientados a servicios y recursos en el centro de las preocupaciones de la disciplina. En el mismo año se publica la versión definitiva de la recomendación IEEE Std 1471, que procura homogeneizar y ordenar la nomenclatura de descripción arquitectónica y homologa los estilos como un modelo fundamental de representación conceptual. (Fielding, 2000)

Desarrollo

Definiciones de SOA

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

- **CBDI** rechaza esa definición:
  - Los componentes pueden no ser conjuntos.
  - La definición sólo considera los componentes y no la práctica o el arte de construir la arquitectura
- **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 interface".
- **IBM:** "Modelo de componente que interrelaciona las diferentes unidades funcionales de las aplicaciones, denominadas **servicios**, a través de **interfaces** y **contratos** bien definidos entre esos servicios. La interfaz se define de forma **neutral,** y debería ser independiente de la plataforma hardware, del sistema operativo y del lenguaje de programación utilizado. Esto permite a los servicios, construidos sobre sistemas heterogéneos, interactuar entre ellos de una manera uniforme u universal".
- **BEA:** Una forma de modulizar los sistemas y aplicaciones en componentes de negocio que pueden combinarse, con interfaces bien definidas para responder a las necesidades de la empresa.
- **Wikipedia:** Es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario, proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

Tomando como partida todas estas definiciones y sus puntos en comunes, SOA para mi no es mas que: " *una filosofía de diseño que permite un mejor alineamiento de las Tecnologías de Información (TI) con las necesidades de negocio, dándole la posibilidad a los clientes de responder de forma más rápida y adaptarse adecuadamente a las presiones del mercado, establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicio* ".

SOA en la Industria

- "La recompensa potencial de SOA es enorme para las empresas que entiendan esta evolución y se muevan hacia estas arquitecturas. La tecnología de computación distribuida promete ser lo suficientemente flexible y elegante para responder a las necesidades de negocios y proporcionar la agilidad de negocios que las compañías han anhelado tanto tiempo, pero siempre ha estado fuera de alcance".(Bloomberg 2004).
- "La mejor solución a la integración de negocios...".
- "SOA ha surgido como la mejor manera de afrontar el desafío de hacer más con menos recursos. Promete hacer la re-utilización y la integración mucho más fáciles, ayudando a reducir el tiempo de desarrollo y aumentando la agilidad

organizacional. No sorprendentemente, el 80% de las organizaciones de TI están implementando aplicaciones usando SOA con web services subyacentes. SOA proporciona mayor flexibilidad para afrontar los cambios tanto en el ambiente de negocios como en la infraestructura tecnológica". (Luis Felipe Cabrera Setiembre 2004)

- "SOA es la próxima ola de desarrollo de aplicaciones. Es más rápida, mejor y más barata".

"Comprender el rol y el significado de SOA, más allá del *hype* simplista, es imperativo para cualquier arquitecto de software empresarial. ... Hacia 2008, SOA y Web Services serán implementados juntos en más del 75% de los proyectos que utilicen SOA y Web Services (probabilidad 0.7)".

- "Hacia 2008, más del 75% de los paquetes de aplicación de ese entonces serán nativamente SOA o expondrán interfaces SOA a través de una capa de envoltura de interfaces (probabilidad 0.8)".
- "Hacia 2008, SOA será la práctica prevalente de ingeniería de software, acabando con los 40 años de dominación de las arquitecturas monolíticas (probabilidad 0.7)".
- "Giga recomienda a los arquitectos considerar SOA como la prioridad número uno en sus esfuerzos de planeamiento arquitectónico". (TI 2003)

	Programación Estructurada	**Objetos**	**Componentes**	**Servicios**
Granularidad	Muy Fina	Fina	Intermedia	Gruesa
Reusabilidad	Baja	Baja	Intermedia	Alta
Acoplamiento	Fuerte	Fuerte	Débil	Muy Débil
Dependencias	Tiempo de Compilación	Tiempo de Compilación	Tiempo de Compilación	Run-Time
Ámbito de Comunicación	Intra -Aplicación	Intra-Aplicación	Inter-Aplicación	Inter-Empresas

Fig. 2. Propiedades de las diferentes arquitecturas existentes.

Beneficios 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 TI.

Desde el punto de vista de la empresa, SOA permite el desarrollo de una nueva generación de aplicaciones dinámicas que resuelven una gran cantidad de problemas de alto nivel, fundamentales para el crecimiento y la competitividad. Las soluciones SOA permiten entre otras cosas:

- **Mejorar la toma de decisiones.** Al integrar el acceso a los servicios e información de negocio dentro de un conjunto de aplicaciones dinámicas compuestas, los directivos disponen de más información y de mejor calidad (más exacta y actualizada), lo que posibilita a las organizaciones poder reaccionar de manera más ágil y rápida cuando surgen problemas o cambios.
- **Mejorar la productividad de los empleados**. Un acceso óptimo a los sistemas, la información y la posibilidad de mejorar los procesos permiten a las empresas aumentar la productividad individual de los empleados.
- **Potenciar las relaciones con clientes y proveedores**. Las ventajas de SOA trascienden las fronteras de la organización, los procesos de fusión y compra de empresas se hacen más rentables al ser más sencilla la integración de sistemas y aplicaciones diferentes. Con SOA se puede conseguir mejorar la capacidad de respuesta a los clientes, habilitando por ejemplo portales unificados de servicios.

Desde el punto de vista de los departamentos de TI, la orientación a servicios supone un marco conceptual mediante el cual se puede simplificar la creación y mantenimiento de sistemas, aplicaciones integradas, y una fórmula para alinear los recursos de TI

con el modelo de negocio y las necesidades de cambio que le afectan.

- **Aplicaciones más productivas y flexibles.** La estrategia de orientación a servicios permite a TI conseguir una mayor productividad de los recursos de TI existentes, como pueden ser las aplicaciones y sistemas ya instalados e incluso los más antiguos. La orientación a servicios permite además el desarrollo de una nueva generación de aplicaciones compuestas que ofrecen capacidades avanzadas y multifuncionales para las organizaciones con independencia de las plataformas y lenguajes de programación que soportan los procesos.
- **Desarrollo de aplicaciones más rápido y económico.** El diseño de servicios basado en estándares facilita la creación de un repositorio de servicios reutilizables que se pueden combinar en servicios de mayor nivel y aplicaciones compuestas en respuesta a nuevas necesidades de la empresa. Con ello se reduce el coste del desarrollo de soluciones y de los ciclos de prueba, se eliminan redundancias y se consigue su puesta en valor en menos tiempo.
- **Aplicaciones más seguras y manejables.** Las soluciones orientadas a servicios proporcionan una infraestructura común (y una documentación común también) para desarrollar servicios seguros, predecibles y gestionables. SOA facilita la posibilidad de añadir nuevos servicios y funcionalidades para gestionar los procesos de negocio críticos. Se accede a los servicios y no a las aplicaciones, además puesto que se utilizan mecanismos de autenticación y autorización robustos en todos los servicios la estrategia de SOA permite dotarse de un nivel de seguridad superior.

Los elementos básicos que conforman SOA son:

- Consumidores.
- Bus de Servicios.
- Repositorio de Servicios.
- Servidores.

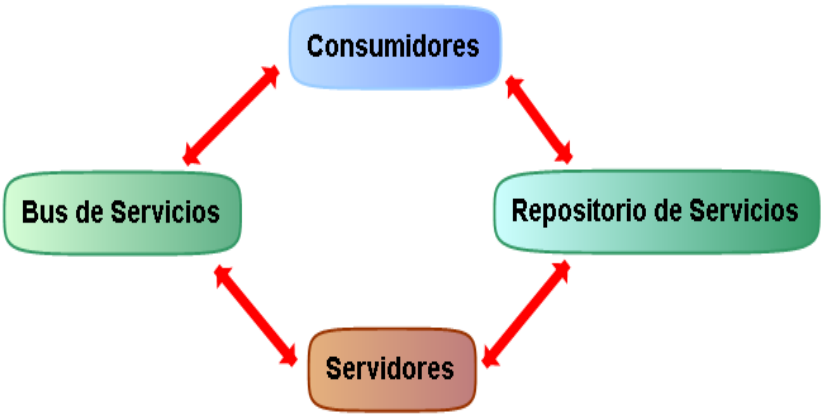

Fig. 3. Relación entre los elementos de una SOA.

¿Qué es un servicio exactamente?

Un servicio es una funcionalidad concreta que puede ser descubierta en la red y que describe tanto lo que puede hacer como el modo de interactuar con ella.

La estrategia de orientación a servicios permite la creación de servicios y aplicaciones compuestas que pueden existir con independencia de las tecnologías subyacentes. En lugar de exigir que todos los datos y lógica de negocio residan en un mismo ordenador, el modelo de servicios facilita el acceso y consumo de los recursos de TI a través de la red. Puesto que los servicios están diseñados para ser independientes, autónomos y para interconectarse adecuadamente, pueden combinarse y recombinarse con suma facilidad en aplicaciones complejas que respondan a las necesidades de cada momento en el seno de una organización.

Tipos de Servicios:

- **Servicios Básicos:** Están centrados en datos o en lógica y encapsulan funcionalidades como cálculos complejos, accesos a datos.
- **Servicios de Negocios:** Representan una tarea de negocio, y que forman parte de un proceso de negocio. Este tipo de servicio suelen ser pocos reutilizables porque están orientados a resolver una tarea muy puntual.
- **Servicios de Procesos:** Servicios de negocio que encapsulan la lógica de procesos. Suelen conservar estado y pueden residir en herramientas BPM.
- **Servicios Públicos:** Servicios accesibles fuera de la organización.

Servicios Web

La adopción de una solución de diseño basada en SOA no exige implantar servicios Web. No obstante, los servicios Web son la forma más habitual de implementar SOA.

Los servicios Web son aplicaciones que utilizan estándares para el transporte, codificación y protocolo de intercambio de información. Los servicios Web permiten la intercomunicación entre sistemas de cualquier plataforma y se utilizan en una gran variedad de escenarios de integración, además posibilitan la abstracción del lenguaje, de la tecnología y del sistema operativo.

Los servicios Web se basan en un conjunto de estándares de comunicación, como son XML para la representación de datos, SOAP (Simple Object Access Protocol) para el intercambio de datos y el lenguaje WSDL (Web Services Description Language) para describir las funcionalidades de un servicio Web.

Repositorio de Servicios: Un Repositorio de Servicios proporciona facilidades para descubrir servicios y adquirir la información necesaria para su uso.

- Información de Contrato.
- Localización.
- Persona de contrato.
- Acuerdos a nivel de servicios.
- Descubrimientos dinámicos.

Bus de Servicios. (ESB): Elementos de la arquitectura SOA que conecta los servicios con sus consumidores y que proporciona:

- Conectividad: Interconectar a los participantes de una arquitectura SOA.
- Soporte a la heterogeneidad de tecnologías: Capaz de conectar clientes basados en distintos lenguajes de programación, sistemas operativos, entornos de ejecución y protocolos de comunicación.
- Soporte a la heterogeneidad de paradigma de comunicación: Capaz de mantener distintos modos de comunicación (sincronías y asíncronas).

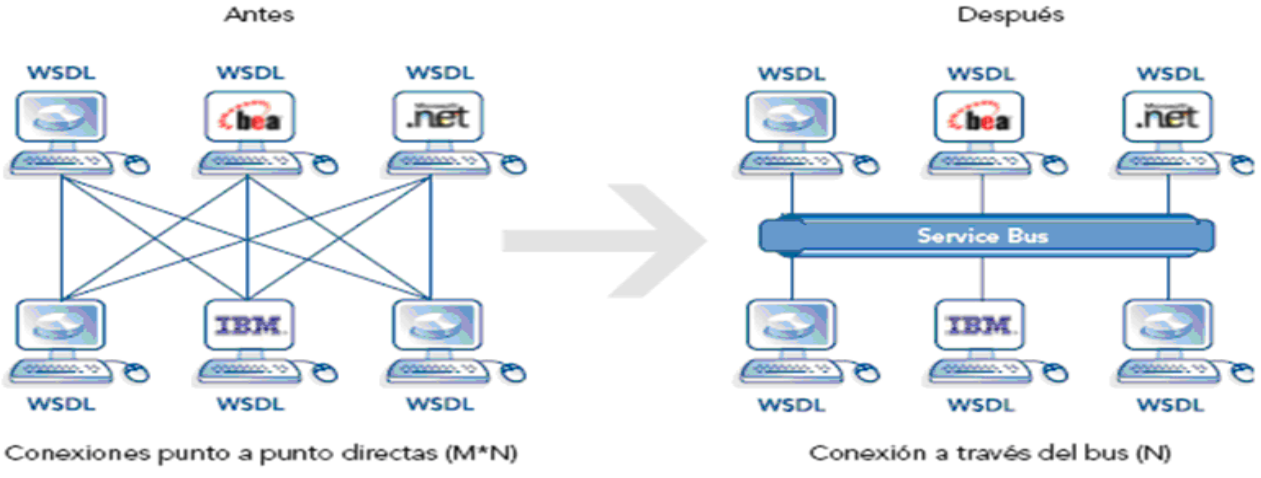

Fig. 4. Representación antes y después de los ESB.

Consumidores de Servicios:

- Pueden Descubrir servicios a través de un repositorio.
- Realizar llamadas a los mismos de acuerdo al contrato y a través del interfaz definido, utilizado o no ESB.

Procesos*:*

Business Process Management (BPM): El concepto de BPM está también muy ligado a SOA. La gestión de procesos de negocio tiene sus orígenes en los Sistemas de Gestión de Calidad Total y la reingeniería de procesos. BPM es más que una combinación de estas disciplinas: BPM es una disciplina de gestión de procesos dirigida mediante Tecnologías de Información, capaz de mejorar la agilidad organizativa, posibilita encadenar los servicios para ganar en eficiencia y asegurar la mejora continua de los mismos, permite la modificación rápida en función de la demanda cambiante y reduce los costos de mantenimiento.

Beneficios de los BPMs

- Traduce la lógica de negocio de una organización definiendo sus flujos de interacciones manuales y automáticas de forma completa.
- Dinamismo, respondiendo a la demanda de los clientes.
- Soporta la larga duración, una instancia de un proceso puede permanecer activa durante mucho tiempo.

Una suite BPM debe dar soporte al modelado, ejecución y monitorización de procesos.

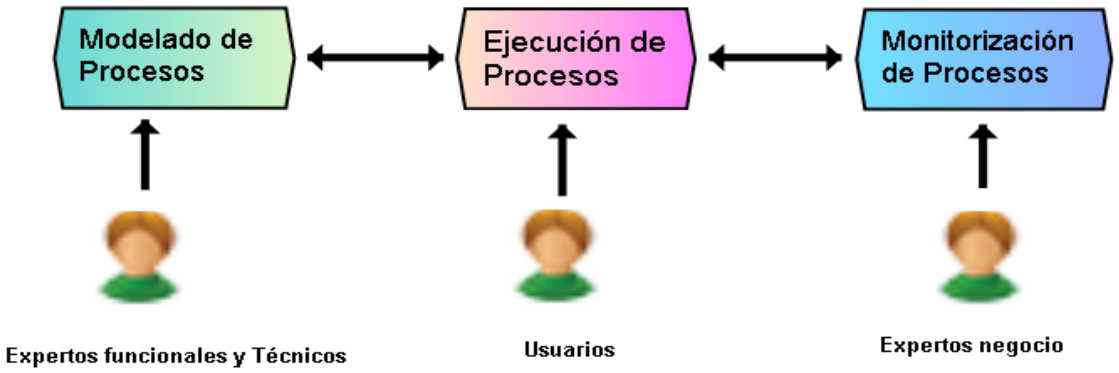

Fig. 5. Representación (Suites BPM).

Modelado de Procesos:

El modelado de procesos a los expertos funcionales deben proveerles:

- Capacidad para definir nuevos procesos, realizar modificaciones sobre los ya existentes.
- Capacidad para capturar procesos.
- Simulación de parámetros de procesos.
- Plantillas de procesos predefinidos.
- Automatización de la documentación en un formato fácilmente exportable.

Desde el punto de vista de los expertos técnicos debe proveerles:

- Instanciaciones de subprocesos, toma de decisiones basándose en las reglas predefinidas.
- Soporte a eventos.
- Gestión de excepciones.
- Facilidades para agilizar la importación y exportación de modelos analíticos creados por expertos funcionales a modelos ejecutables.
- Adecuación a lenguajes estándares de definición de procesos (como BPEL).

Ejecución de Procesos:

- Ofrece diversos mecanismos de invocación de procesos de manera síncrona y asíncrona.
- Permite la invocación de procesos desde otros procesos.
- Permite el versionado de procesos.
- Tienen en cuenta la escalabilidad, el rendimiento y la fiabilidad.

Monitorización de Procesos:

- Al ejecutar procesos, es posible monitorearlos para analizarlos y poder encontrar posibilidades de mejora en los mismos.
- Con este modelo la dirección puede saber en tiempo real la situación del negocio.

Reflexiones:

Por las características de nuestra universidad sería muy factible seguir trabajado en aras de alcanzar un gran desarrollo en la aplicación de esta arquitectura para el desarrollo de todas las aplicaciones que se brindan, digamos para ser más precisos y comenzar por la base, de todos los servicios internos, como la amplia gama de los sistemas de gestión de RRHH, pases, comedores, trasportaciones de profesores y estudiantes, etc., que con la implantación de UDDI, donde se comparten, se describen y se publican estos servicios se logran que todos nuestros usuarios puedan acceder y usar estos para diversos fines, hasta para crear nuevos servicios, aplicando el modelado de procesos de negocios (BPM), a través de las diferentes interfaces de dichos servicios.

La implementación de la arquitectura SOA, podemos decir que va mucho mas allá de la publicación de servicios, una gran parte de esta es montar y garantizar servicios de una institución, mediantes los mencionados Bus de servicios, repositorios, servidores, esto podría ser el núcleo que gestionará los diferentes servicios descritos en él.

Estoy consiente que se ha comenzado a desarrollar y se está aplicando dicha estrategia en nuestra universidad, el éxito radica en seguir profundizando y desarrollarnos aún más en esta esfera.

Conclusiones

- SOA es un estilo arquitectónico.
- Su éxito se debe a la alineación de procesos de negocios con los sistemas de infraestructura.
- Se basa en estándares abiertos.
- Permite interoperabilidad entre tecnologías distintas.
- Separa la lógica de los procesos, de los sistemas base. Permitiendo fácil cambio de los mismos.
- Es gradual, su implementación debe hacerse planeada y paulatinamente.

Opinión de la Autora:

En definitiva, en la actualidad hablar de arquitectura de software es hablar de SOA, es el paradigma actual en cuanto a arquitectura de software se refiere. El ciclo de vida de desarrollo de software, desde el diseño hasta la operación, está encima de la mesa para ser re-estudiado en base a las aportaciones y posibilidades de SOA. A medida que el mercado vaya adoptando las estrategias y tecnologías implicadas, veremos madurar el desarrollo de software hasta niveles no alcanzados por el momento. El mundo del software no deja de reinventarse continuamente, tenemos una nueva generación de ideas, vamos a ver hasta dónde nos llevan.

Referencias Bibliográficas

1. **Bloomberg, J. (2004)**. "The Role of the Service-Oriented Architect".
2. **D. Garlan, M. Shaw.** An Introduction to Software Architecture. January 1994.

http://www.sei.cmu.edu/publications/documents/94.reports/94.tr.021.html. CMU/SEI-94-TR-021.

3. **Fielding, Roy Thomas.** Architectural Styles and the design of network-based software architectures. University of California. Irvine : s.n., 2000. Tesis Doctoral.

4. **Clements, P. (Enero de 1996.).** "Coming Attractions in Software Architecture" Technical Report.

5. **Dijkstra, E. (Enero de 1983).** "The Structure of the The Multiprogramming System." Communications of the ACM.

6. **IT, G. (2003).** "Application Architecture and Design."

7. **Jr., F. B. (1975).** The Mythical Man-Month.

8. **Kron, F. D. y. H. (1976).** "Programming-in-the-Large Versus Programming-in-the-Small", IEEE Transaction in Software Engineering.

9. L**uis Felipe Cabrera, C. K., Don Box (Septiembre 2004)**."An Introduction to the Web Service Architecture and its Specifications".

10. **Northrop, P. C. y. L. (Febrero de 1996)**. "Software Architecture: An Executive Overview".

11. **Parnas, D. (Diciembre de 1972).** "On the Criteria for Decomposing Systems into Modules." Communications of the ACM.

12. **Reed. J. B. y. K. (2001).** "Why We Need a Different View of Software Architecture". The Working IEEE/IFIP Conference on Software Architecture (WICSA), Amsterdam.

13. **Sharp, I. P. (27 al 31 de Octubre de 1969).** Comentario en Discusión sobre teoría y práctica de la ingeniería de software. Software Engineering Concepts and Techniques: Proceedings of the NATO conferences, Roma.

Wirth, N. (Abril de 1971). "Program Development by Stepwise Refinement", Communications of the ACM.

15. **WWISA (1999)** "Worldwide Institute of Software Architects." "Philosophy" **Volume**, DOI:

16. http://www.microsoft.com/spanish/msdn/arquitectura.

17. http://www.sei.cmu.edu/publications/publications.html.