UN ENFOQUE TEÓRICO E INTUITIVO A LA ARQUITECTURA ORIENTADA A SERVICIOS (SOA)

Julián Astorga Campos
Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática,
San José, Costa Rica
julianastorga@gmail.com

Juan Luis Quirós Venegas
Universidad de Costa Rica
Escuela de Ciencias de la Computación e Informática,
San José, Costa Rica
jlquiros@gmail.com

Abstract

SOA is a paradigm which enables us to design and to build systems which will have more flexibility, scalability and will be reusable. In a SOA environment service producers offer their services to potential consumers which will have access to them in a very standardized way. To reach a better understanding of this interaction we will represent the communication between the key elements of SOA using service automatons. Futhermore, a short reference to Web Services and SOA´s is going to be made, also the reasons that make us to think about why SOA is the right solution under specific conditions.

Keywords: SOA, services, software architecture, service automatons.

Resumen

SOA es un paradigma que permite diseñar y construir sistemas que serán flexibles, escalables y reutilizables. En un ambiente SOA, los productores de servicios hacen disponibles sus recursos a los consumidores como servicios independientes a los que tienen acceso de un modo estandarizado. Para lograr entender esa interacción, se representa la interacción entre los distintos elementos constitutivos de SOA mediante autómatas. Además, se hace referencia a la relación entre los Web Services y SOA´s, así como a las razones que nos obligan a reflexionar si es correcto este enfoque bajo ciertas condiciones.

Palabras clave: SOA, servicios, arquitectura de software, autómatas de servicio.

1. Introducción

Indudablemente, existen cambios en el tiempo y estas variaciones siempre están ligadas con actos o acciones que nos obligan a analizarlos y a descifrarlos, con el fin de lograr un mejor entendimiento de nuestro entorno actual, así como intentar sacar provecho del mismo. Actualmente, ante la vorágine de cambios que embarga a nuestro mundo, podríamos afirmar que estamos viviendo un renacimiento del darwinismo: Ya no es el espécimen más fuerte quien

alcanza la supervivencia, ni tampoco el más inteligente, sino el espécimen que es más receptivo al cambio [1]; afirmación tácitamente aceptada por el mundo económico y tecnológico.

En otras palabras, es la flexibilidad el factor clave de nuestra sociedad. Mas, los procesos y los sistemas también van evolucionando, siendo una característica inherente de los mismos que ese crecimiento esté unido a un incremento de la complejidad de ellos. Además, hemos llegado a un punto, en el cual las visiones antiguas para resolver estos problemas de estabilidad y evolución de los sistemas dejan de ser efectivas, dejan de brindar esa armonía resultante de la centralización de los paquetes de software; consecuentemente, un nuevo enfoque debe emerger como solución a estos límites.

Indudablemente, ese nuevo paradigma reside en aceptar la heterogeneidad de las soluciones y las necesidades informáticas y es, desde esa perspectiva, donde emerge la Arquitectura Orientada a Servicios (SOA). Este enfoque no solo brinda una respuesta que ayuda a los sistemas a mantener su escalabilidad y flexibilidad mientras evolucionan, sino que también brinda la satisfacción de alcanzar un facilitador que una los cambios en tecnologías de la información y los cambios en los procesos de múltiples organizaciones.

Por lo tanto, podríamos definir SOA no como un objeto a comprar, mas sí una nueva forma de pensamiento que beneficia la arquitectura y diseño de sistemas. En consecuencia, en este artículo, se pretende mostrar que los conceptos y características asociadas a SOA no son difíciles de comprender, esto debido a que se opta por un enfoque intuitivo o informal para abordar el tópico en cuestión. Igualmente, en las próximas secciones se pretende brindar una definición de SOA, que contenga sus respectivas características básicas. Asimismo, se presenta una simplificación del enfoque basado en "service automaton"[1], para demostrar la funcionalidad de un "service registry", la asociación de los SOA´s con los Web Services y esclarecer cuando no es apto optar por un SOA.

2. Fundamentos de SOA

Primeramente, podemos decir que SOA se encuentra en todas partes, por más que esta proposición suene un poco abstracta [6]. Si tomamos una máquina dispensadora de bebidas, la cual, si se le ingresa una moneda, dispensa una bebida, ya sea té o café (representados por los símbolos café – C – y té – T –). Se podría tomar que la máquina es el *service provider*, el cliente es el *service consumer* y la empresa que instala las máquinas es el *service broker*, ya que instala la máquina (o sea sabe cual servicio se debe ofrecer al cliente) con la capacidad de leer las monedas de un país específico.

Según Gebhardt[3], la arquitectura de software describe la estructura de un sistema, cuyo núcleo está compuesto de componentes, que gozan de características externamente visibles y por capacidades especiales para comunicarse con otros componentes. Sin embargo, todo anhelo de toda filosofía de construcción de sistemas informáticos radica en el hecho de minimizar las dependencias entre los componentes.

Por lo tanto, podríamos afirmar que SOA es un modelo de componentes, cuyas funcionalidades están implementadas como servicios reutilizables, independientes y con un grado mínimo de acople. Estos servicios serán invocados mediante interfaces previamente definidas, que deben ser independientes del hardware, del sistema operativo y del lenguaje de programación. Asimismo, la información es habilitada por medio de componentes atómicos e independientes, cuya comunicación se da gracias a mecanismos estandarizados. Además, los elementos intrínsecos de todo SOA se pueden listar de la siguiente forma:

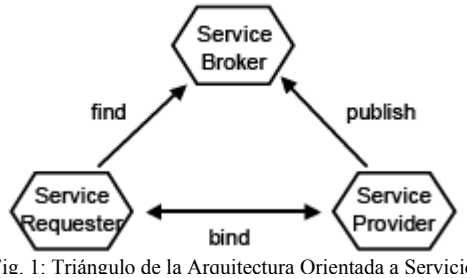

Fig. 1: Triángulo de la Arquitectura Orientada a Servicios

[1] Según los autores, una traducción del término anglosajón sería "autómata de servicio".

- *Service Requestor* – Consumidor de Servicio

  La función que consume el resultado del servicio provisto por un proveedor.

- *Service Provider* – Proveedor de Servicio

  La función que brinda un servicio en respuesta a una llamada o petición desde un consumidor.

- *Service Broker* .- Registro de Servicios

  El consumo y registro de servicios es facilitado mediante un registro de servicios, donde los servicios son descritos mediante protocolos y mensajes.

Este panorama general de los conceptos que engloban un SOA y las características básicas de este paradigma se pueden complementar con una visión más detallada de la interacción de los elementos estructurales de esta filosofía de arquitectura de software, con un especial énfasis en las funcionalidades de un *service registry*, con el fin de comprender el rol de cada elemento dentro de un sistema que promueva las características SOA.

3. Representación de la interacción de los elementos estructurales de SOA mediante un "service automaton"

Indiscutiblemente, un sistema basado en un SOA necesita interactuar idóneamente con los otros servicios que están presentes en la solución, con el fin de poder garantizar un rendimiento óptimo. Por lo tanto, a la hora de querer utilizar un servicio, el *service requester* envía una solicitud al *service broker*, para poder conocer al service provider que mejor pueda satisfacer el servicio requerido. Además, vale mencionar que un *service provider* tuvo que publicar su servicio al *service broker*, con el fin de que pueda ofrecer sus servicios.

Sin embargo, ya sea por razones internas de la empresa que haya implementado el servicio a proveer o por cuestiones de propiedad intelectual, un proveedor de servicios no hace público información acerca de la estructura interna de su servicio, mas sí todas las posibilidades de interacción con el mismo. Consecuentemente, si un *service broker* posee múltiples proveedores de servicios P_1, …, P_n, entonces es él quien decide cual de las interacciones solicitadas R es cumplida satisfactoriamente por el servicio P. Para profundizar el estudio de este proceso, nos valdremos de los *service automatons* propuestos por Massuthe y Wolf [2], para poder resolver ese problema de concordancia entre el servicio solicitado y la búsqueda del servicio correcto ofrecido por P para resolver R.

Por lo tanto, un *service automaton* refleja el flujo interno de un servicio, así como el comportamiento de la comunicación entre R y P mediante sus interfaces, cuya representación se da con las etiquetas de las transiciones del *service automaton*. Una etiqueta *!x* representa enviar un mensaje por el canal x, mientras que *?x* significa recibir un mensaje por el canal x, además r sería equivalente a la representación de ϵ. Cabe resaltar, que el contenido del mensaje no es importante a la hora de utilizar este método. Igualmente, un estado de transición en cada estado de cada *service automaton* (tanto para R como para P) concuerde, por lo que una comunicación exitosa de los servicios de R y P es expresada como una propiedad de la tabla de transiciones (si ambos servicios alcanzan un estado final al procesar todas las transacciones). Tomando como ejemplo las definiciones de autómatas según Aho [4], un *service automaton* se puede definir de la siguiente manera:

Definición de un s*ervice automaton*: Un service automaton es *un autómata no determinista* $A = [I ; Q ; T ; q_0 ; \Omega]$ *que consiste de*

- *una* interface *I*
- *un conjunto finito de estados Q*
- *un conjunto finito de estados* $T \subseteq Q \times L \times Q$ *de transiciones donde* $L = \{ ?x \ I \ x \in Iin \} \cup \{ !x \ I \ x \in Iout \} \cup \{ r \}$
- *un estado inicial* $q_0 \in Q$

- *un conjunto Ω ⊆ Q de estados finales*

Como resultado de la aplicación del enfoque de *service automatons* a los servicios del caso del dispensador de bebidas presentado en la sección anterior, se producirán los siguientes *service automatons* :

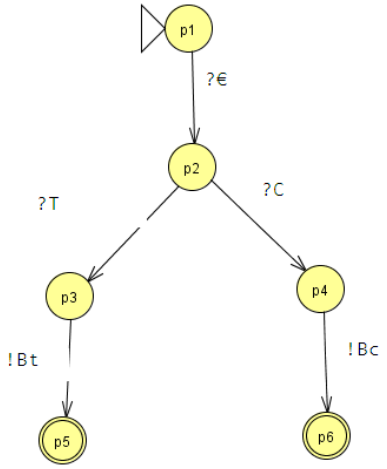

Fig2. *Service Automaton* para P_V [2]

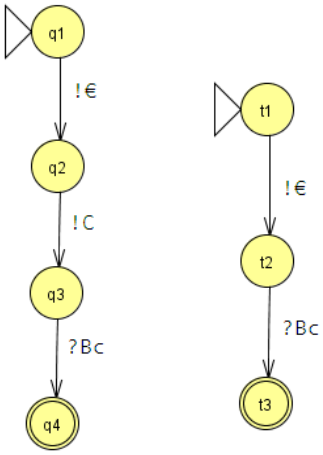

Fig3. Service Automaton para R_C y R_E [2]

Cabe destacar que dos autómatas en comunicación R y P deben poseer interfaces I tal que un autómata envía únicamente mensajes que puedan ser recibidos por el otro autómata y viceversa, por lo que se puede generalizar que $I_{inR} = I_{outP}$ y $I_{outR} = I_{inP}$. Por último, para poder garantizar que el servicio solicitado R sea resuelto satisfactoriamente por P, nos valemos de la siguiente definición:

Definición de la interacción de un service automata: Sean P y R 2 service automatas. Sea $QR \cap QP = \varnothing$*, el sistema de transición* $R + P = [Q; T; q_0 ; \Omega]$ *consiste de*

- *conjunto de estados* $Q \subseteq QR \times QP \times bufferMensajes\ (MC)$,
- *conjunto de transiciones* $T \subseteq Q \times (LP \cup LR) \times Q$,
- *estado inicial* q_0
- *conjunto* $\Omega \subseteq Q$ *de estados finales*

Esta definición establece que R y P se mueven libremente en la transición R + P y que una acción de recibido se da únicamente cuando el mensaje está presente en el buffer de mensajes. Si al final de la aplicación de las transiciones se llega a un estado final de R como de P y el buffer de mensajes está vacío, implica que R sí puede ser resuelto por P. Para ejemplificar este concepto, se muestran los diagramas de transición para R+P y R+P, asimismo una representación no tan estricta de como los autómatas de servicio interactúan entre ellos para demostrar que un servicio solicitado puede ser realizado por un proveedor de servicios (tanto un caso de aceptación como otro de rechazo):

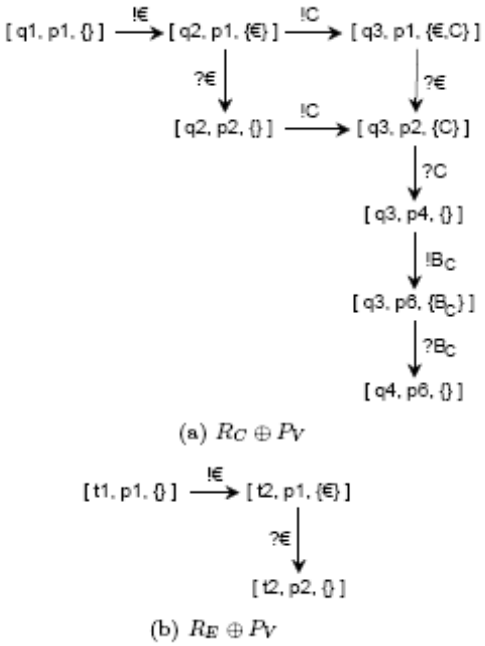

Fig4. Transiciones de R_C+P y R_E+P [2]

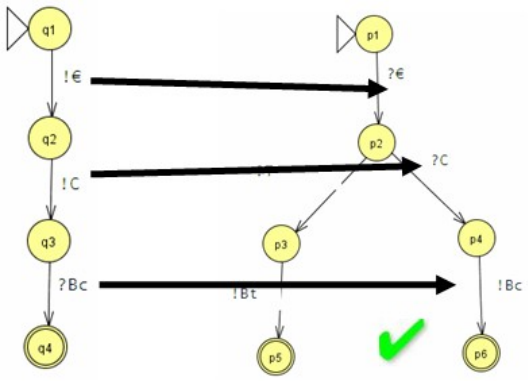

Fig5. Servicio solicitado R_C sí es satisfecho por P_V [2]

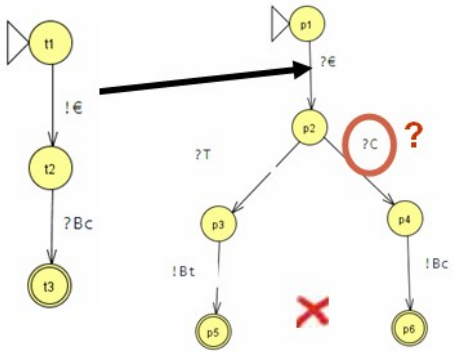

Fig6. Servicio solicitado R_E no es satisfecho por P_V [2]

Consecuentemente, al aplicar un enfoque simplificado del método *service automaton* propuesto por Massuthe y Wolf [2], logramos comprender no solo los elementos estructurales intrínsecos de este paradigma, sino que también logramos entender como estos elementos interactúan entre sí, para satisfacer la solicitud de un servicio. Este conocimiento de los fundamentos de SOA nos permite establecer una base sólida para ver como es que los SOA´s hacen un intensivo uso de la web para facilitar las interacciones de los mismos.

4. SOA´s y Web Services

Web Services representan una forma de poder realizar ciertos aspectos técnicos de SOA, mas ellos mismos te pueden introducir ciertos problemas. Primero, los estándares no son lo suficientemente maduros para garantizar interoperabilidad. Segundo, Web Services por sí solos no garantizan un nivel de bajo acoplamiento apropiado.

Sin embargo, a pesar que SOA es una tendencia de la arquitectura de software que enfatiza en la construcción de aplicaciones intercomunicables y con bajo sentido de acople, es ampliamente aceptado decir que un web service es un SOA, los cuales deben cumplir estos dos requerimientos:

- Las interfaces de comunicación deben estar basadas en un protocolo de transporte como http, ftp, smtp, CORBA.

- Los mensajes deben ser en formato xml.

Por lo tanto, es necesario hacer la advertencia en no caer en aspectos muy específicos de web services, ya que los mismos no serán el estándar final para la integración de sistemas, por lo que web services se deben de utilizar solamente cuando aspectos de infraestructura específicos son de gran importancia. Además, se debe reconocer cuando no es prudente optar por un SOA, lo que resultaría en un enfoque erróneo para solucionar un problema.

5. ¿ Cuándo no es apto optar por un SOA ?

Existen situaciones en las cuales no es recomendable optar por un SOA [5], las cuales son:

1. *Cuando se tiene un ambiente de TI homogéneos*

   Si se utiliza las tecnologías de un solo proveedor, entonces es posible que la sobrecarga adicional de una SOA no sería eficaz en función del costo para usted, un SOA es muchas veces un poco práctico. Además, entornos heterogéneos de hardware no podrán beneficiarse de una SOA menos que también tienen una infraestructura heterogénea de software -- es decir, diferentes sistemas operativos o de middleware.

2. *Cuando ocurre en tiempo real el rendimiento es crítico*

   Confiar en la comunicación asíncrona para proporcionar acoplamiento entre los consumidores y los productores de servicios, SOA´s no están bien adaptadas a las situaciones que requieren aplican estrictamente los tiempos de respuesta. Mas, SOA es un excelente método para las empresas que buscan formas de acelerar el procesamiento de sus archivos, sin tener que deshacerse de sus aplicaciones.

3. *Cuando las cosas no cambian*

   Si hay pocas razones para cambiar la lógica de negocio, presentación, flujo de datos, proceso, o cualquier otro aspecto de la aplicación, conversión de estas aplicaciones a un SOA podría no regresar valor suficiente para hacer que el esfuerzo valga la pena.

Claramente, no se puede garantizar que este paradigma es apto para todas las soluciones de software, ya que pueden existir condiciones que impidan que una solución basada en este paradigma tenga realmente el impacto deseado. Sin embargo, la arquitectura orientada a servicios está emergiendo como una solución muy interesante cuando las necesidades de los clientes radican en la flexibilidad, reutilización y bajo acoplamiento.

6. Conclusiones

En síntesis, en este artículo se logró establecer una explicación intuitiva de los conceptos relacionados con SOA, en la cual se optó por una gran simplicidad en la definición de los conceptos intrínsecos del misma, sin dejar de lado sus cualidades más importantes, ni aspectos de importancia como los Web Services. Finalmente, se establecieron ciertos parámetros mínimos que ayudan al programador a discernir acerca de la conveniencia del uso de un SOA en una situación particular.

De igual forma, para futuros escritos, resultaría favorable ahondar en el estudio de modelos que permitan a las empresas adoptar, de forma poco traumática, este paradigma. Igualmente, sería beneficioso estudiar como se comportan los SOA´s en ambientes de desarrollo ágil y como herramientas de cuarta generación soportan la creación de SOA´s.

Bibliografía

[1] Josuttis, Nicolai M. **SOA in Practice**. O´Reilly Media, Inc. Agosto, 2007

[2] Massuthe, Meter y Wolf, Karsten. **An Algorithm for Matching Nondeterministic Services** with Operating Guidelines. 2006

[3] Gebhardt, Mike. **Serviceorientierte vs. Event-basierte Architekturen**. Johann Wolfgang Goethe – Universität. 2005

[4] Aho, Alfred V & Sethi, Ravi & Ullman, Jeffrey D.: ***Compilers: Principles, Techniques and Tools***, Addison Wesley. 1979.

[5] Blommberg, Jason. **When not to use an SOA.** http://www.zapthink.com/report.html?id=ZAPFLASH-02162004 . (29.10.2007)

[6] He, Hao. **What is Service Oriented Architecture?** http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html . (29.10.2007)