Documento de estado del arte en SOA y Cálculo Pi

Jorge Alejandro Rico García Jhon Jairo Gómez Otero

10/Marzo/2007

Introducción

Este documento, pretende dar a conocer las tecnologías relacionadas con la arquitectura orientada a servicios (SOA) y los conceptos fundamentales necesarios para comprender adecuadamente su estructura. Dentro de dichos conceptos, se definirán las diversas implementaciones de lenguajes que pretenden construir este tipo de arquitecturas haciendo uso de los servicios Web, tales como WS-BPEL, WS-CDL y otras.

De igual forma, se abarcarán los conceptos manejados por el cálculo Pi y sus aplicaciones dentro del campo de los servicios Web para garantizar correctitud en las definiciones de servicios realizadas.

1 Arquitectura Orientada a Servicios (SOA: Services Oriented Architecture)

SOA es una arquitectura que emerge como una consecuencia directa de los procesos de negocio y la evolución de la tecnología. Por el lado de los negocios se basa principalmente en procesos como el "outsourcing" de operaciones no relacionadas con la actividad principal y la reingeniería. Esto le permite a SOA, ser una aproximación a la tecnología de información manejada en los negocios y por el lado de la tecnología, surge con los estándares de arquitecturas de capa media, aprendiendo de las fallas de los sistemas de objetos distribuidos y arquitecturas de capa media orientadas a la mensajería; es por esto, que SOA es un concepto arquitectural basado en componentes de poco o ningún nivel de acoplamiento.

Esta arquitectura está basada en el modelo de procesos de negocio, describiendo el orden de ejecución de actividades y las condiciones bajo las cuales se deben desarrollar. Adicionalmente, facilita la optimización de propiedades del proceso (Costos, Tiempo), al permitir cambios en el flujo de trabajo y el "outsourcing" de actividades de una forma segura y confiable. Este modelo se basa en el funcionamiento de las empresas, pues estas deben rediseñar su flujo de trabajo, cuando no sean competitivas en ciertas actividades, para lo cual es

necesaria la delegación de servicios en otras empresas o realizando adquisición de servicios o fusión. Lo anterior indica que la arquitectura fue diseñada para su adaptación a los cambios del negocio u operaciones, de una manera ágil, haciendo de los procesos, unidades funcionales de fácil transformación que apoyen la consecución de los objetivos de un sistema.

Este esquema de utilización de servicios, se describe de mejor forma en la siguiente figura:

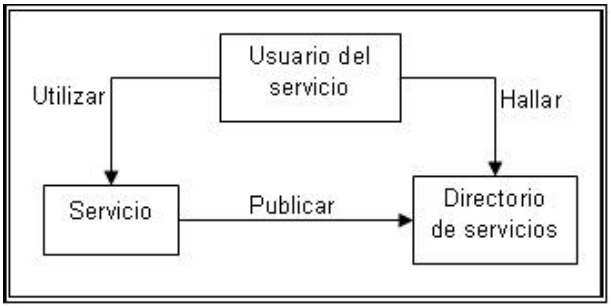

Figura 1: Ciclo de funcionamiento de los servicios

Existe un usuario que para poder utilizar un servicio, tendrá que buscarlo en un lugar donde se encuentren especificaciones de los servicios existentes. De igual forma el proveedor del servicio, tendrá que proporcionar los detalles del servicio que ofrece, para que sus clientes conozcan adecuadamente las funcionalidades y su forma de utilización, publicándolos en un lugar de acceso a los clientes y en una forma que ellos lo entiendan[1, Sección 1.4.1. Bind/Publish/Find].

Para lograr este proceso de publicación, búsqueda y utilización, se debe adicionalmente tener en cuenta ciertos protocolos y lenguajes que permitirán la comunicación necesaria entre cada uno de los involucrados en la arquitectura.

Con la llegada de las tecnologías de capa media, el proceso de búsqueda y selección de servicios se simplificó, pues la capa media incorporada realiza dichas funciones y el usuario del servicio solo tiene que comunicarse con ella (Figura 2). Esta capa es comúnmente conocida como bus de servicio y permite una separación entre los servicios y su usuario, haciendo para él imposible conocer el servicio utilizado, pero brindando la seguridad que la tarea solicitada será realizada completa y adecuadamente.

Ahora, teniendo en cuenta todas estas características de la arquitectura, es posible mencionar algunas de las funcionalidades que brinda la aplicación de SOA en una solución. Estas se encuentran plasmadas en los siguientes aspectos:

Integración de aplicaciones
Dado que uno de los objetivos de la arquitectura es la disminución de acoplamiento, al contemplar las aplicaciones como un conjunto de servicios que

se ejecutan para lograr un objetivo común, es posible que existan dentro de ese mismo ámbito otras aplicaciones que puedan complementar o mejorar su calidad. Sin embargo, es muy posible que estas sean islas de información, separadas por plataformas tecnológicas. SOA permite que éstas puedan integrarse, haciendo uso de la arquitectura y de las tecnologías desarrolladas para ella, básicamente servicios Web. Estos permitirán la integración de funcionalidades y contenidos, dicho propiamente, SOA permite la integración de aplicaciones de software y contenido de portales. El primer modo de integración, es soportado primordialmente por SOAP (Simple Object Access Protocol) y la integración de portales está soportada mediante WSRP (Web Services for Remote Portlets).

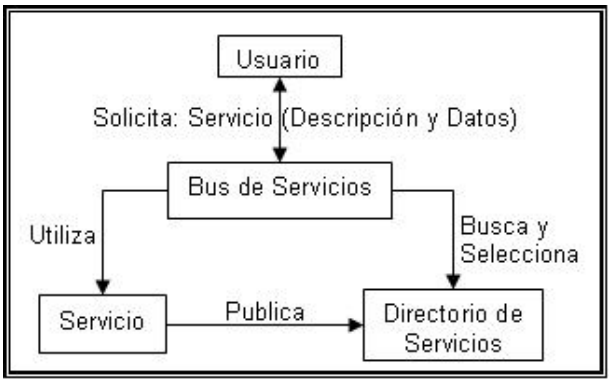

Figura 2: Uso del bus de servicios

Mejoramiento de los procesos de negocio

El mejoramiento de los negocios se da gracias a la flexibilidad que ofrece SOA, permitiendo la evolución de los procesos para su adaptación a las nuevas necesidades del mercado en el cual se ejecute. La capacidad de delegación de operaciones incluida en la plataforma brinda la posibilidad, que en dicha delegación, sea posible el cambio del proveedor de la solución, obteniendo mejores resultados en el proceso general. De igual forma, es posible la realización de cambios en la organización de las actividades involucradas dentro de un proceso, permitiendo así la obtención de nuevos y mejores resultados bajo la organización de dichos procesos.

2 Tecnologías para implementar SOA

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 y WS-CDL para desarrollar dichas tareas y permitir la elaboración de procesos bien estructurados con la propiedad de ser una arquitectura de servicios.

WS-CDL (Web Services - Choreography Description Language)[3]

Este es un lenguaje utilizado para la definición de servicios dentro de la plataforma SOA, basado en XML y cuyo objetivo es la descripción del comportamiento de cada uno de los servicios establecidos para lograr un objetivo común. Sin embargo, antes de WS-CDL ya existía un lenguaje que permitía la descripción de las funcionalidades de un servicio, este era WSDL (Web Services Definition Language), el cual podía describir el conjunto de funciones ofrecidas, con los posibles fallos que podrían ocurrir; estas cualidades son denominadas como comportamientos observables. Adicionalmente, con WS-CDL, fue posible conocer o describir algunos comportamientos no observables, definidos como 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.

Lo que permite en esencia WS-CDL es la definición de las restricciones de ordenamiento (secuencias, dependencias, etc.), y el establecimiento de reglas y comportamientos que permitan la colaboración adecuada entre los servicios. Así, este lenguaje permite la descripción sin ambigüedades de las colaboraciones establecidas entre servicios, determinando un protocolo de negociación. De esta manera, cada organización puede desarrollar de manera independiente su propio papel en la colaboración siempre y cuando se respete el "contrato global" para que de esta manera se garantice la interoperabilidad.

WS-CDL es importante dentro de SOA porque es una tecnología escalable, garantiza la interoperabilidad efectiva y segura de servicios, permite tener servicios mas robustos reduciendo el tiempo de implementación de los mismos

Estructura de WS-CDL

WS-CDL es un lenguaje organizado por capas, que permiten diferentes niveles de expresión de las coreografías de un servicio (Figura 3). 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.

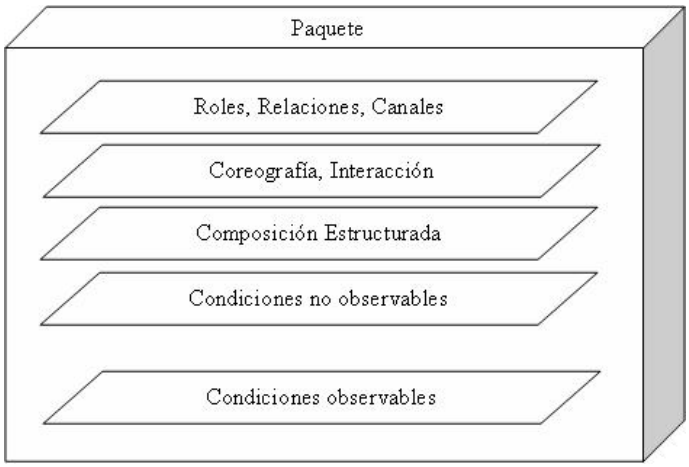

Figura 3: Estructura por capas de WS-CDL

WS-BPEL (Web Services - Business Process Execution Language)

WS-BPEL o especificación del lenguaje de ejecución de procesos de negocio en servicios Web, pertenece a la capa de componentes establecida en SOA y es, en conjunto con WS-CDL, una alternativa para la implementación y manejo de Servicios Web. WS-BPEL surge como necesidad de ser el integrador o el engranaje para las diversas tecnologías que funcionan bajo SOA, pero que no logran una interoperabilidad al 100%, lo que restringe su funcionamiento y adaptabilidad. BPEL tiene como objetivo primordial la "orquestación" de servicios Web, es decir, la definición de un nuevo servicio Web a partir de servicios Web existentes.

WS-BPEL nace tomando propiedades de dos lenguajes propietarios con características diferentes pero complementarias, estos lenguajes son: WSFL (Web Services Flow 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 extendible, 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[1, Sección 14.2. Architectural Concepts]
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 (Figura 4).

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.

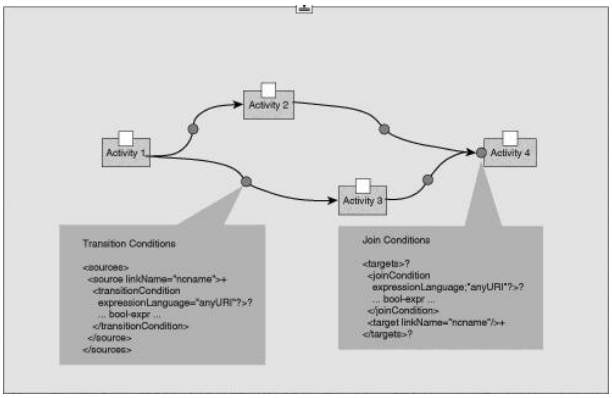

Figura 4: Proceso con WS-BPEL

3 Cálculo Pi

El cálculo Pi, es un lenguaje formal que hace parte de la familia del álgebra de procesos, cuyo objetivo es la descripción y análisis de propiedades del procesamiento concurrente. Fue desarrollado por Robin Milner, como una interpretación de los procesos concurrentes, móviles y su comunicación; por lo tanto, esta herramienta provee una infraestructura para la representación, simulación, análisis y verificación de sistemas móviles de comunicación.

Un sistema representado en el cálculo Pi, consta de múltiples procesos concurrentes de los cuales hay parejas de procesos que interactúan entre ellos, enviando y recibiendo mensajes de manera sincronizada. Esta comunicación es realizada sobre canales complementarios de entrada y salida, así cuando un proceso receptor recibe un mensaje, recibe también un canal, el que puede utilizar para comunicar sus respuestas. Esta característica, llamada movilidad, permite que las conexiones de la red cambien con las interacciones realizadas; es decir, se producen cambios dinámicos en la topología de la comunicación entre procesos.[6]

Dentro del cálculo Pi, un proceso es una entidad autónoma que posee puertos, a través de los cuales podrá comunicarse con otros procesos, esta estructura es manejada de la siguiente forma:

$$NombreDeProceso(Lista\ de\ Puertos) = ComportamientoDelProceso$$

En la descripción del comportamiento del proceso, los nombres de los puertos con barras horizontales sobre ellos serán interpretados como puertos de salida y aquellos que no tienen barras como puertos de entrada (Figuras 5 y 6).

Dentro de esta definición, es posible describir procesos con comportamientos secuenciales, repetitivos y comportamientos alternativos.

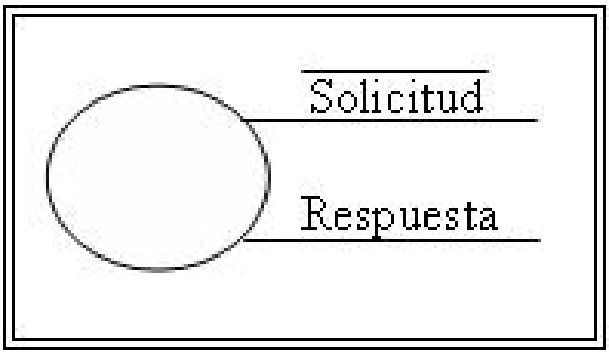

Figura 5: Notación Gráfica

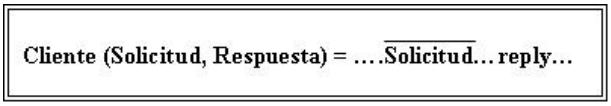

Figura 6: Notación Algebraica

También es posible describir la comunicación entre procesos, permitiendo la composición de estos, al establecer comunicación a través de puertos con nombres complementarios. Estos procesos de comunicación concurrente pueden sincronizar sus comportamientos dependiendo de su disposición para la comunicación.

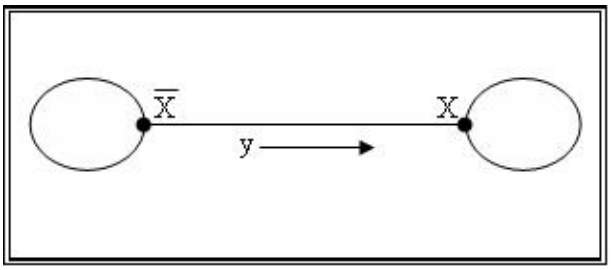

Figura 7: Procesos Complementarios

Otra de las características de este cálculo, es la descripción de movilidad, referida al cambio dinámico de la topología de comunicación entre procesos, efectuada por la adquisición y pérdida de puertos de un proceso, por los cuales se comunicará. Este cambio es realizado transmitiendo el nombre del puerto como el valor de alguna comunicación entre dos procesos, permitiendo que el proceso receptor conozca el puerto a utilizar.

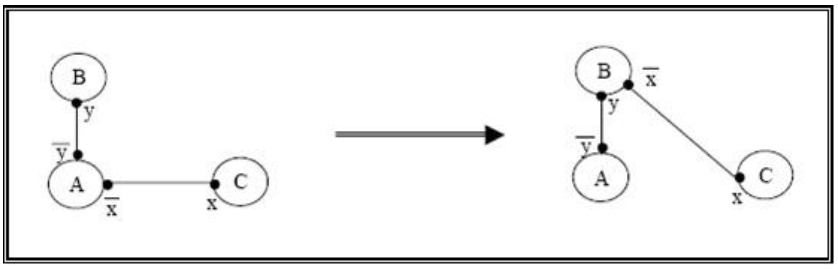

Figura 8: Movilidad

Aplicaciones del Cálculo PI.

PICT[7]

Es un lenguaje de programación diseñado con base en el cálculo Pi, y con adición de primitivas de alto nivel utilizadas comúnmente en la construcción de programas, tales como estructuras de datos, protocolos para el retorno de resultados, comunicación selectiva y objetos concurrentes. Adicionalmente, integra algunas características de los lenguajes orientados a objetos, tales como el polimorfismo.

NOMADIC PICT[8]

Este lenguaje es una extensión de PICT con las nociones de ubicación, agentes, migración, distribución y fallas, basado en el diseño y primitivas de comunicación para la computación móvil.

Este lenguaje propone un cálculo simple de agentes que permiten la implementación de los algoritmos para una infraestructura distribuida como codificaciones o compilaciones del cálculo en un fragmento, que emplea la primitiva de comunicación dependiente de la ubicación, es decir aquellas que requieren el conocimiento por parte del programador de la ubicación actual de un agente móvil para poder comunicarse con él.

Este cálculo esta equipado con una semántica precisa, entregando una base sólida para el entendimiento de algoritmos y para el estudio de su correctitud y robustez.

PI4SOA[13]

Es un proyecto que provee una implementación de código libre del lenguaje WS-CDL, configurado para aplicar los beneficios del cálculo Pi en la arquitectura orientada a servicios. Esta herramienta permite la especificación de las interacciones entre un conjunto de servicios colaborativos desde una perspectiva global o neutral, permitiendo la proyección de las responsabilidades de cada servicio desde su representación, sin hacerlo desde la perspectiva de un servicio. Adicionalmente, el concepto de cálculo Pi es incluido para asegurar

que las coreografías descritas estén libres de "bloqueos" y asegurar que estas usen correctamente los servicios ya definidos por otras descripciones.

El proyecto tiene una caracteristicas esenciales que lo definen, orientan y estructuran:

- Editor de Coreografía: Será provisto como un plugin para el entorno de desarrollo Eclipse, permitiendo a través de este la importación - exportación de la representación desde/hacia el estándar WS-CDL
- "Endpoint Projection": Permitirá proyectar una coreografía en base al rol o al participante
- "Endpoint Monitoring": Usará la descripción del comportamiento de un "endpoint" involucrado dentro de una coreografía para de esta manera determinar si se esta comportándose correctamente.
- "Endpoint Generation": Generadores de "endpoint" serán implementados para poducir una versión estructural de un "endpoint" basado en el comportamiento proyectado.

Bibliografía

[1] Weerawarana, Sanjiva. Curbera, Francisco. Leymann, Frank. Storey, Tony. Ferguson, Donald F. Web Services Platform Architecture:SOAP,WSDL,WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall. Marzo 22 de 2005.

[2] Taylor, Hugh. Pulier, Eric. Understanding Enterprise SOA. Manning Publications. 2006.

[3] Ross, Steve. Talbot. Understanding WS-CDL. Versión 4/8/05

[4] Ross, Stephen. Orchestration and Choreography: Standards, Tools and Technologies for Distributed Workflows.

[5] Oliva, Enrico. Web Services Choreography. Diciembre 22 de 2005.

[6] http://www.fairdene.com/picalculus/pi-c-tutorial.pdf. Junio 01 de 2006.

[7] http://www.cis.upenn.edu/~ bcpierce/papers/pict/Html/Pict.html, PICT, Marzo 01 de 2006.

[8] http://www.cl.cam.ac.uk/users/pes20/nomadicpict.html, Nomadic PICT, Marzo 01 de 2006.

[9] http://www.ebpml.org/bpel4ws.htm, BPEL for Web Services, Marzo 01 de 2006.

[10] http://www-128.ibm.com/developerworks/webservices/library/ws-bpelwp/, Bussiness Processes in a Web Services World, Marzo 01 de 2006.

[11] http://www-128.ibm.com/developerworks/webservices/library/ws-bpelcol1/, Business Process with BPEL4WS, Parte 1, Marzo 01 de 2006.

[12] http://www-128.ibm.com/developerworks/webservices/library/ws-bpelcol2/, Business Process with BPEL4WS, Parte 2, Marzo 01 de 2006.

[13] http://www.pi4tech.com/index.php PI4SOA Project, Marzo 01 de 2006.

[14] Business Process Execution Language for Web Services Specification. Version 1.1. Mayo 5 de 2003.

[15] https://sourceforge.net PI4SOA Project, Junio 01 de 2006