El ciclo de vida de un servicio Web compuesto: virtudes y carencias de las soluciones actuales[*]

Mauricio Espinoza-Mejía[**] y Pedro Álvarez

Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza
María de Luna 1, 50018 Zaragoza, España
{mespinoz, alvaper}@unizar.es

Resumen: La composición de servicios Web en un nuevo servicio más complejo y útil es un importante reto. Actualmente existen algunas técnicas que permiten componer servicios Web, sin embargo aún existen algunas limitaciones. En este trabajo se analizan las virtudes y carencias de los diferentes estándares, técnicas, modelos y lenguajes presentes en distintas plataformas de composición, propuestas por grupos de investigación relevantes que permiten cubrir todo el ciclo de vida de un sistema de composición de servicios.

Palabras Clave: Servicios Web, composición de servicios, ciclo de vida, plataformas no comerciales.

1. Introducción

Los servicios Web prometen la interoperabilidad de aplicaciones diversas en plataformas heterogéneas, y así posibilitan la automatización y conexión dinámica de procesos comerciales dentro y a través de las empresas mediante aplicaciones *EAI* (*Enterprise Application Integration*) y *B2B* (*Business-to-Business*). Particularmente, la composición de servicios Web ha experimentado un gran interés debido a la posibilidad de desarrollar nuevos servicios a través de la composición de servicios individuales. Desafortunadamente dado que los servicios individuales son desarrollados usando diferentes propuestas y tecnologías, se requiere actualmente una cantidad considerable de programación y esfuerzo para componer los servicios de forma que sea posible obtener una aplicación libre de errores y en un tiempo de desarrollo prudencial. Estas limitaciones han provocado un gran esfuerzo en investigación y desarrollo enfocado al área de composición de servicios Web, para contribuir al desarrollo de estándares, herramientas, modelos y lenguajes que permitan efectuar el diseño y ejecución de servicios satisfactoriamente.

En este trabajo se analizan los estándares, técnicas, modelos y lenguajes presentes en distintas plataformas para composición de servicios Web propuestas por grupos de investigación relevantes. El resto del trabajo está organizado de la siguiente forma: en la sección 2, se presenta una arquitectura genérica para la composición automática de servicios Web, la cual permite definir los requerimientos mínimos para cubrir todo el ciclo de vida de un sistema de composición de servicios: *descripción y modelado de*

[*] Este artículo es el resultado del trabajo realizado como parte de la línea de investigación titulada "Integración y Composición de Servicios Web mediante workflows" del programa de doctorado en Ingeniería de Sistemas e Informática de la Universidad de Zaragoza.
[**] Trabajo soportado por una beca de la Universidad de Zaragoza-Banco Santander Central Hispano.

servicios, diseño y definición de la composición, generación del modelo de proceso, evaluación del servicio compuesto y ejecución y control de la composición. Diferentes técnicas, modelos y lenguajes (usadas en diferentes propuestas de composición de servicios) son analizadas en las secciones 3 al 7. En la sección 8, comparamos algunas plataformas de composición de servicios y relacionamos éstas a las diferentes técnicas, modelos y lenguajes analizados.

2. Arquitectura Genérica

En esta sección presentamos una arquitectura genérica para la composición automática de servicios Web. Esta estructura no pretende reflejar la utilización de un lenguaje particular, plataforma u algoritmo de composición. El objetivo de la arquitectura es presentar los requerimientos mínimos necesarios para cubrir todo el ciclo de vida de un sistema de composición de servicios. Para que se inicie un proceso de composición de servicios Web, un *proveedor de servicio* debe *describir y publicar un servicio* (etapa 1 en la Figura 1); por otra parte un *solicitante del servicio* debe efectuar una solicitud para que se inicie el proceso de composición. El proceso de *definición y composición de servicios* (2) puede soportar la traducción de un lenguaje de diseño a un lenguaje más formal, el cual es utilizado por el *generador del proceso de composición* (3). Debido a que algunos servicios pueden tener una funcionalidad muy similar, es posible que se genere más de un modelo del servicio compuesto que cubra los requisitos. En este caso los servicios compuestos son *evaluados* (4) usando la información provista por los atributos no funcionales; finalmente el servicio compuesto seleccionado es *ejecutado y controlado* (5).

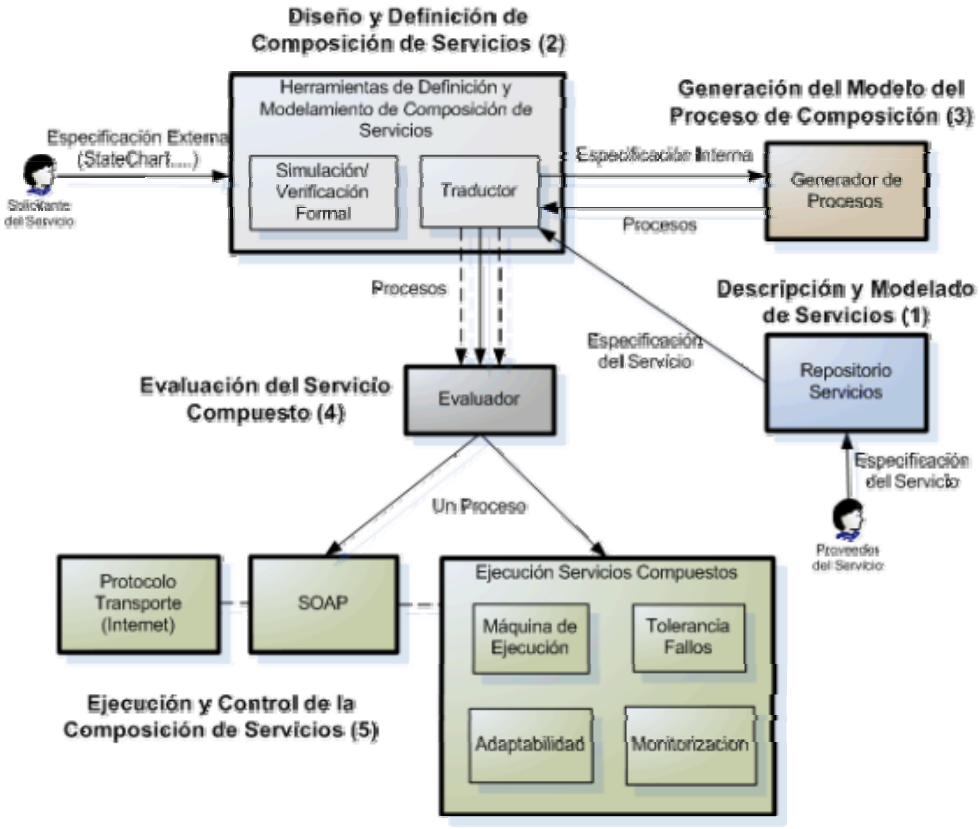

Figura 1. Arquitectura genérica para un sistema de composición de servicios Web

La idea de partir de una estructura genérica para analizar el estado del arte sobre composición de servicios no es nueva [3, 8]. En otros trabajos además se hace un estudio de diferentes estrategias de composición [1, 2, 4-7]; sin embargo este trabajo a diferencia de los anteriores no se centra únicamente en una etapa del ciclo de vida sino cubre todo el proceso. Para cada una de las etapas de la arquitectura de la Figura 1, extraemos un conjunto de parámetros, técnicas, modelos y lenguajes. Los parámetros pueden ser utilizados como un mecanismo de comparación para evaluar distintas soluciones para composición de servicios Web; mientras que el vistazo general que ofrecemos de las diferentes técnicas, modelos y lenguajes esperamos ayuden a los diseñadores y desarrolladores de composición de servicios a enfocar sus esfuerzos y liberar soluciones mas útiles y perdurables, al mismo tiempo que dirija las necesidades con respecto a la tecnología existente.

En las próximas secciones identificamos un conjunto de parámetros, técnicas, modelos y lenguajes con los cuales es posible manejar todo el ciclo de vida de una herramienta dedicada a la composición de servicios Web.

3. Descripción y Modelado de Servicios

Usando la tecnología actual, un proveedor de servicios, puede desplegar y exponer la interfaze de un servicio usando *WSDL* (*Web Services Description Language)*. La descripción WSDL de un servicio contiene una especificación de las operaciones que un servicio expone y los enlaces de información detallando como invocar las operaciones en términos de protocolos y direcciones. Aunque este nivel de detalle es suficiente para construir aplicaciones de Servicios Web simples, éste no es suficiente cuando se requiere crear servicios complejos y razonar acerca de su composición [9]. WSDL no es la única manera de describir un servicio, dado la enorme cantidad de lenguajes que están siendo propuestos como estándar por diferentes grupos de investigación.

3.1 Lenguajes de Descripción de Servicios Web

A continuación se describen brevemente algunas características de los principales lenguajes de descripción de servicios Web utilizados en diferentes plataformas para composición de Servicios Web.

- **WSDL:** Permite describir las operaciones que ofrece el servicio, los mensajes de entrada y salida que soporta y los tipos de datos usados, los cuales son definidos en términos de esquemas XML. WSDL define lo que hace el servicio y no cómo lo hace, por lo que no es posible expresar la semántica de los mensajes intercambiados ni el orden correcto de llamada. Una opción es utilizar WSCI el cual permite describir el orden de invocación y cómo las operaciones pueden ser coreografiadas en el contexto de intercambio de mensajes en el cual los Servicios Web participan. Como WSCI, la intención de WS-CDL es definir un lenguaje para describir escenarios de interacción multi-partes (o coreografías).

- **WSCI/WSDL:** WSDL solamente especifica a nivel sintáctico sus interfaces lo que limita la creación de servicios Web complejos. En el trabajo presentado en [7] se propone describir los servicios Web en función de su comportamiento y no

únicamente como una perspectiva de entradas/salidas (como propone WSDL). Una alternativa puede ser describir un servicio utilizando WSDL junto con *WSCI (Web Service Choreography Interface)* o *WS-CDL (Web Services Choreography Description Language)*.

- **WSDL anotado con Ontologías:** Otra propuestas para intentar mejorar las limitaciones de una propuesta tan rígida como WSDL es la de añadir semántica a la descripción de los servicios Web. La descripción semántica de un servicio puede ser alcanzada usando ontologías que soporten vocabularios y modelos de dominio compartidos. Usando ontologías de dominio específico, la semántica implícita en las estructuras de una descripción de servicio, las cuáles son conocidos sólo por el escritor de la descripción (proveedor del servicio Web), pueden ser hechas explícitas a otros usuarios.

- **ebRIM**: Para la descripción y descubrimiento de servicios, los estándares Web usan WSDL y UDDI, respectivamente. En el caso de ebXML (un estándar para la comunicación entre negocios), los mecanismos de descripción y descubrimiento forman parte del *ebXML registry*. ebXML usa un modelo de información de registros denominado ebRIM (*eBusiness Registry Information Model*), el cual permite definir qué tipos de objetos son almacenados en el registro y cómo éstos son organizados. El modelo de información de registros puede ser implementado dentro de *ebXML Registry* en forma de un esquema de bases de datos relacional, de objetos o algún otro esquema físico.

4. Diseño y Definición de Composición de Servicios

En la arquitectura genérica introducida en la sección 2, se observa que una solicitud externa efectuada por un *solicitante del servicio* es necesaria para que se inicie el proceso de composición; el cual tiene sentido cuando la solicitud no puede ser satisfecha por un servicio simple preexistente, sino por la combinación apropiada de algunos servicios disponibles.

La mayoría de las herramientas que soportan el diseño y definición de composición de servicios, hacen una distinción entre lenguajes de especificación de servicios internos y externos. Los lenguajes externos (llamados además lenguajes de diseño) permiten representar (usualmente en forma gráfica) la composición de servicios de forma que pueda ser fácilmente entendida por los interesados. Por otra parte los lenguajes internos (lenguajes de composición de servicios) son generalmente lenguajes más precisos y formales, los cuales son usados para generar el proceso de composición. Algunas herramientas pueden soportar la traducción de un diseño de servicio compuesto en un lenguaje de descripción.

4.1 Lenguajes de Composición de Servicios Web

En esta sección detallamos brevemente algunas características de los lenguajes que permiten describir la composición de servicios.

- **BPEL4WS:** *Business Process Execution Language for Web Services*, es un lenguaje con una sintaxis basada en XML y soporta la especificación de procesos que

involucran operaciones provistas por uno o algunos Servicios Web. BPEL4WS soporta la descripción de dos tipos de procesos: abstractos y ejecutables. El lenguaje hace uso de los conceptos desarrollados en el área de administración de workflows y se puede decir que es relativamente expresivo, comparándolo con algunos lenguajes soportados por sistemas de workflows existentes y estándares relacionados.

- **ebXML BPSS:** *Electronic Business XML,* es una serie de estándares que intentan proveer una plataforma de implementación para colaboración entre negocios. ebXML adopta una propuesta basada en coreografía para la composición de servicios. Específicamente una colaboración de negocio es descrita como un conjunto de Perfiles de Protocolos de Colaboración (CPP).

- **BPML:** La iniciativa de Administración de Procesos de Negocio (BPMi) es un consorcio que intenta contribuir al desarrollo de estándares de descripción de procesos. El consorcio ha publicado una especificación para un lenguaje de descripción de procesos orientados al servicio llamado *BPML (Business Process Modeling Language)*, el cual es similar en algunas formas a BPEL4WS y está orientado para describir orquestación. BPML se beneficia de un estándar previo llamado WSCI desarrollado por los interesados en BPMi. WSCI integra algunos de los modelos encontrados en BPML y BPEL4WS (por ejemplo, secuencia, ejecución paralela, primitivas para envío/recepción, etc.) y es usado para describir coreografía.

4.2 Lenguajes Formales de Composición de Servicios

Los lenguajes de composición de servicios descritos en la sección anterior, presentan limitaciones al momento de diseñar y analizar una composición de servicios, permitir técnicas de simulación, asegurar la verificación de propiedades como seguridad y administración de recursos o comprobar la consistencia de datos, y la satisfacción de las restricciones de negocio. Por este motivo, algunos grupos de investigación han propuesto algunos lenguajes formales de composición de servicios que cubra estas necesidades.

- **OWL-S (previamente conocida como DAML-S):** *OWL-based Web service ontology,* es una ontología de servicios que permite el descubrimiento, invocación, composición, inter-operación y monitorización de la ejecución en forma automática. OWL-S modela los servicios usando una ontología, la cual consta de tres partes: (1) *service profile:* describe lo que el servicio requiere de parte de los usuarios; (2) *service model:* especifica cómo trabaja el servicio; y (3) service *grounding:* ofrece información de cómo usar el servicio.
  El modelo de procesos es una subclase de *service model,* el cual describe un servicio en términos de entradas, salidas, precondiciones, post-condiciones y si es necesario sus propios subprocesos. OWL-S distingue tres tipos de procesos: *atómicos*, los cuales no tienen sub-procesos, *simples*, los cuales no pueden ser invocados directamente y son usados como un elemento de abstracción para procesos atómicos o compuestos; y *compuestos*, los cuales constan de subprocesos. Para proveer una base formal para su interacción con el mundo real OWL-S toma la propuesta del *cálculo de situación*, un formalismo basado en la lógica que modela explícitamente el hecho de que sobre el tiempo, diferentes "situaciones" pueden surgir.

- **Componentes Web:** Esta propuesta trata a los servicios como componentes para dar soporte a los principios de desarrollo de software básicos tales como reutilización, especialización, y extensión. La idea principal es encapsular la información lógica compuesta dentro de una definición de clase, lo cual representa un componente Web. La interfaz pública de un componente Web puede entonces ser publicada y usada para descubrimiento y reutilización.

- **Composición de Procesos Algebraica:** La composición algebraica de servicios tiene como meta introducir descripciones mucho más simples que otra propuesta, y modelar los servicios como procesos móviles para asegurar la verificación de propiedades como: la seguridad y administración de recursos. La teoría de los procesos móviles esta basada en el π-cálculo, en el cual la entidad básica es un proceso, el cual puede ser un proceso vacío; o una elección entre varias operaciones de entrada/salida; una composición paralela; una definición recursiva; o una invocación recursiva.

- **Redes de Petri:** Las redes de Petri son una propuesta de modelado de procesos bien establecida. Una red de Petri es un grafo dirigido, conectado y bipartito en el cual los nodos representan ubicaciones y transiciones, y existen tokens que ocupan los estados. Cuando existe al menos un token en cada ubicación conectada a una transición, ésta es sensibilizada. Es posible modelar servicios como redes de Petri asignando transiciones a métodos y lugares a estados [16, 24]. Cada servicio tiene una red de Petri asociada que describe su comportamiento y tiene dos puertos: una ubicación de entrada y de salida. En cualquier momento, un servicio puede estar en uno de los siguientes estados: no instanciado, listo, ejecutado, suspendido, o completado.

- **Inspección de Modelos:** La inspección de modelos es usado formalmente para verificar sistemas concurrentes de estado finito. La especificación del sistema es descrita usando lógica temporal, y para ver si la especificación tiene aplicación se efectúa una comprobación del modelo. Podemos aplicar la inspección de modelos a la composición de servicios Web, verificando la exactitud de una especificación de servicios. Entre las propiedades que se pueden comprobar están la consistencia de datos, y la satisfacción de restricciones de negocio [17].

- **Roman Model (Acciones con Modelo de Procesos basado en Autómatas):** Con esta propuesta los servicios son modelados de una forma muy abstracta, basados en la noción de actividades. Básicamente existe un alfabeto (finito) de nombres de actividades, pero no se modela la estructura interna (entradas, salidas o interacción con el mundo). En [20] se especifica el flujo de procesos internos de un servicio Web, a través de sistemas de transición. En el caso más general, éstos son árboles potencialmente infinitos, dónde cada rama corresponde a una secuencia permitida de ejecuciones de las actividades.

- **Máquinas Mealy (Máquinas de estado finito):** Otra alternativa son las *máquinas Mealy*, las cuales son máquinas de estado finito con entradas y salidas. Los servicios se comunican enviando mensajes asíncronos, donde cada servicio tiene una cola. Un "observador" global sigue la pista a todos los mensajes. La conversación es introducida como una secuencia de mensajes. Estudiando y entendiendo las

propiedades de la conversación, este método provee nuevos mecanismos para diseñar y analizar una composición de servicios "bien estructurada". La composición automática de servicios es el objetivo de la plataforma presentada en [18], la cual describe el comportamiento de los servicios Web como un árbol de ejecución y entonces traduce éste en una maquina de estados finito. Ellos proponen un algoritmo que comprueba la existencia de la composición y retorna una si existe

4.3 Modelo de Compatibilidad

Un asunto fundamental al definir un servicio compuesto es conocer si sus servicios componentes pueden formar una composición [5]. Por ejemplo, puede ser difícil invocar una operación si no existe una correspondencia entre los parámetros solicitados para su operación (ej. tipos de datos, número de parámetros) y aquellos transmitidos por el servicio del cliente. Del trabajo presentado en [10], es posible identificar que al menos es posible comprobar la compatibilidad de los servicios a dos niveles:

- **Sintáctico:** A nivel sintáctico la comparación es muy simple y puede efectuarse a: (1) modo de operación y (2) enlaces, lo cual permite comparar los protocolos de enlace de los servicios que interactúan.

- **Semántico:** A nivel semántico la comparación es mucho más compleja y entre otras cosas, las reglas semánticas pueden incluir: (1) *mensajes*, tomando en cuenta el número de parámetros de los mensajes, sus tipos de datos, roles de negocio y unidades; (2) *operaciones semánticas*, esta regla asegura que las operaciones interconectadas tienen categorías y propósitos "compatibles*";* (3) *propiedades cualitativas*, los mecanismos de composición generalmente tienen preferencias referente a la calidad de operaciones que les gustaría subcontratar. Las reglas cualitativas comprueban las propiedades cualitativas de las operaciones que interactúan, y (4) *validez de la composición*, lo cual verifica que cierta combinación de servicios de una forma específica sea válida y además agregue un valor añadido. La última regla semántica se verifica a nivel de composición, mientras que las anteriores tratan la compatibilidad a niveles de servicio y operación. En otro trabajo del mismo autor se extiende el modelo propuesto en [10] y se introduce los conceptos de grado de compatibilidad y ד-compatibilidad para atender la compatibilidad total y parcial. Un mayor detalle de esta propuesta puede ser consultada en [11].

4.4 Fiabilidad de la Composición

Existe una amplia diversidad de investigación en esta área, sin embargo no existe una definición clara de fiabilidad que establezca su alcance en el contexto de un servicio Web compuesto. En [17] por ejemplo proponen una definición de fiabilidad que toma en consideración diferentes aspectos que afectan la fiabilidad de un servicio Web compuesto durante su composición y ejecución. En esta sección describiremos los aspectos relacionados a la fiabilidad de un servicio Web compuesto durante la etapa de composición (antes de la ejecución del servicio compuesto). Mas adelante describiremos los aspectos de fiabilidad que deben ser considerados durante la ejecución del servicio.

- **Verificación de la Especificación del Servicio Compuesto:** Este aspecto requiere que la notación para especificar la composición de servicios Web sea correcta, y que la especificación de servicios compuesto, cuando éste es puesto en ejecución, traduzca correctamente el flujo de procesos que representa. Una posible solución para esta actividad es descrita en [7], la cual usa técnicas de chequeo de modelos para verificar la especificación creada de acuerdo a un estándar de composición de servicios Web particular.

- **Acuerdo entre la Especificación del Servicio Compuesto y los servicios individuales:** Es necesario que exista un acuerdo entre los requerimientos del flujo de procesos representados por el servicio compuesto y las restricciones esperadas por los servicios Web componentes. Existen algunos esfuerzos que permiten a un servicio Web especificar sus políticas con relación a la participación en un servicio compuesto [10-12].

- **Soporte de cambios en la definición de interfaces:** Dada la naturaleza autónoma de los proveedores de servicios, es posible que un proveedor pueda cambiar la interfaze de un servicio que forma parte de un servicio Web compuesto; esto no debería causar un fallo en el proceso representado por el servicio compuesto o conducir a errores inesperados. Una solución es construir una capa de funcionalidad sobre la pila de estándares de composición, como se describe en [25]. Esta propuesta usa un modelo conversacional entre servicios Web para manejar los cambios de interfaces en los servicios individuales.

4.5 Estrategias para expresar los Esquemas de Composición

En el trabajo presentado en [5] se identifican al menos cinco estrategias de composición de servicios: *estática o dinámica, manejada por modelos, composición manejada por reglas de negocio, declarativa y composición basada en ontologías*. En nuestra opinión, no consideramos que exista una estrategia de composición estática o dinámica, pues esto hace referencia al tiempo en que los servicios Web son compuestos.

La composición estática toma lugar durante el tiempo de diseño, cuando la arquitectura y el diseño del software son planeados. Los componentes a ser usados son escogidos, ligados y finalmente compilados y desplegados. La composición estática, puede ser demasiado restrictiva especialmente cuando se requiere que los componentes automáticamente se adapten a cambios impredecibles.

En la composición dinámica, los servicios son ligados durante el tiempo de ejecución y el ambiente de servicio es altamente flexible y dinámico. Nuevos servicios están disponibles diariamente y el número de proveedores crece constantemente. Una herramienta que soporte una composición dinámica de servicios, debería poder adaptarse transparentemente a los cambios del ambiente, y a los requisitos del cliente con intervención mínima del usuario. A continuación describimos algunas estrategias para expresar los esquemas de composición, en donde los servicios Web son compuestos dinámicamente.

- **Manejada por Modelos:** Facilita la administración y desarrollo de composiciones dinámicas de servicios. *UML (Unified Modelling Language)* es usado para proveer un alto nivel de abstracción y posibilitar el mapeo directo con otros estándares como *BPEL (Business Process Execution Language)*. Algunas propuestas que utilizan esta estrategia de composición, utilizan *OCL (Object Constraint Language)* como lenguaje para definir las reglas de negocio. Las reglas de negocio pueden ser usadas para estructurar y programar la composición de servicios, y describir la selección y enlaces de los servicios.

- **Reglas de Negocio**: La composición de servicios manejada por reglas de negocio es muy similar a la manejada por modelos. Las reglas de negocio son clasificadas para facilitar la especificación de un servicio Web compuesto. Por ejemplo las reglas relacionadas con roles controlan los participantes involucrados en el proceso de composición, las reglas relacionados con mensajes regulan el uso de la información, o las reglas relacionadas con eventos controlan el comportamiento del servicio compuesto en reacción a un evento inesperado.

- **Declarativa:** Esta propuesta es dinámica y es diferente a otras estrategias de composición de servicios, en el sentido de que los servicios son creados dinámicamente para ejecutar las solicitudes del cliente. La propuesta declarativa consta de dos fases: la primera toma como entrada una situación inicial y una meta y construye un plan genérico para alcanzar el objetivo. La segunda escoge un plan genérico, descubre los servicios apropiados y construye un workflow sobre éstos. Esta estrategia utiliza la planeación como mecanismo para resolver la composición.

- **Semántica:** Permite componer los servicios en forma automática y utilizan ontologías tanto para describir los servicios cuanto para acceder a estos.

5. Generación del Modelo de Proceso de Composición

El generador de proceso (descrito en la arquitectura de la Figura 1) intenta resolver el requerimiento proporcionado por el solicitante de servicio, componiendo los servicios atómicos anunciados por los proveedores de servicios. Este módulo usualmente toma las funcionalidades de los servicios como entrada, y su salida es un modelo que describe el servicio compuesto. El modelo del proceso generado contiene un conjunto de servicios atómicos seleccionados y el flujo de control y datos entre éstos servicios. A continuación se describen las diferentes técnicas que puede utilizar el *generador del proceso de composición* para descubrir los servicios existentes que formarán parte de la composición de servicios.

5.1 Descubrimiento de Servicios

Es el proceso de localizar qué servicios están disponibles para tomar parte en una composición. Hemos identificado al menos dos criterios para analizar las técnicas de descubrimiento de servicios: 1) por los criterios de búsqueda aplicados; 2) por la topología de información de soporte a la búsqueda y descubrimiento de servicios.

Por los criterios de búsqueda

- **UDDI (sintáctica):** UDDI (*Universal Description, Discovery, and Integration*) es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros. UDDI se ha enfocado sobre detalles operacionales y sintácticos para la implementación y ejecución de servicios Web, lo cual limita los mecanismos de búsqueda a solo basadas en palabras clave.

- **Preferencias usuario:** En [13] por ejemplo se propone una estructura para el descubrimiento y selección de Servicios Web basado en evaluaciones de calidad personalizadas. Ellos proponen mejorar el registro UDDI a través de un componente que además de ejecutar el filtrado de servicios en base a términos específicos, ejecute una búsqueda cooperativa entre descripciones de servicios y las preferencias del usuario

- **Descubrimiento Semántico:** Algunas propuestas han sugerido agregar semántica a los servicios Web. La semántica puede ser agregada a los actuales estándares de servicios Web como UDDI o WSDL o los servicios pueden ser descritos usando algún lenguaje descripción basado en ontologías como DAML-S [19]. En el trabajo publicado en [12] se mejoran las técnicas actuales de composición de procesos Web usando *Plantillas de Procesos Semánticos* para capturar los requisitos semánticos del proceso. Usan ontologías en la definición de las plantillas lo que permite una descripción mucho más rica de los requisitos de actividad y una forma más efectiva de localizar servicios. Para el descubrimiento de servicios, no solamente se considera la funcionalidad, sino también la Calidad del Servicio (QoS) de las actividades correspondientes. Esta propuesta combina el poder expresivo de las normas presentes de composición de servicio Web y la ventaja de las técnicas de Web semántica para la definición de plantilla de proceso y descubrimiento de servicios Web.

Por la Topología

Centralizada y *Peer-to-Peer*: Las propuestas basadas en esta arquitectura argumentan que un diseño no centralizado para la búsqueda de Servicios Web es más escalable, tolerante a fallos y eficiente. En [14] se propone *WSPDS (Web Services Peer-to-peer Discovery Service)* un servicio de descubrimiento inter-operable y completamente descentralizado, con capacidad de descubrimiento a nivel semántico. WSPDS esta implementado como un servicio cooperativo en donde una red de WSPDS resuelven las consultas de descubrimiento lanzadas por cada *Peer*. [15] propone una arquitectura basada en ontologías para organizar los registros y permitir la clasificación semántica de todos los servicios Web. Cada uno de estos registros soporta publicación semántica, la cual es usada durante el proceso de descubrimiento, además esta propuesta esta apoyada en una arquitectura *peer-to-peer* lo que permite acceder a múltiples registros.

6. Evaluación del Servicio Compuesto

Es muy común que algunos servicios tengan la misma funcionalidad o al menos muy similar. Por lo tanto es posible que el *generador del proceso de composición* genere más de un modelo de composición que cubra los requisitos del usuario. En este caso los servicios compuestos son evaluados de acuerdo a todas sus funciones, usando la información provista por los atributos no funcionales. El método mas común es utilizar funciones, por ejemplo se puede asignar pesos para cada atributo no funcional y el mejor servicio compuesto será aquel que obtenga más puntuación.

7. Ejecución del Servicio Compuesto

Después de que un único modelo de composición de servicios ha sido seleccionado, el servicio esta listo para ejecutarse. La ejecución de un servicio Web compuesto puede pensarse como una secuencia de mensajes o conversaciones con los servicios Web integrados al modelo de procesos. El flujo de datos de un servicio compuesto es definido como las acciones que los datos de salida de un servicio ejecutado anteriormente transfieren a las entradas de un servicio ejecutado más adelante. Durante la ejecución del servicio compuesto es necesario tomar en cuenta algunos parámetros como los que describimos a continuación.

7.1 Mecanismo de Ejecución de Procesos

Desde las plataformas de composición de servicios Web existentes es posible identificar al menos tres mecanismos para coordinar la ejecución de un servicio Web compuesto:

- **Centralizada:** En este modelo, la responsabilidad para coordinar la ejecución de un servicio compuesto es confiando a un simple "planificador". Este planificador interactúa con cada uno de los servicios componentes procesando y despachando los mensajes. La arquitectura interna del planificador central es similar a la de un sistema administrativo tradicional de workflows excepto que los recursos son todos servicios en lugar de actores humanos, y que no existe una base de datos compartida a través de la cual la información puede estar implícitamente pasada de un actor a otro. En lugar de eso, la información debe ser explícitamente pasada a través de intercambio de mensajes.

- **Distribuida**: El paradigma distribuido en contraste espera que los servicios Web participantes compartan su contexto de ejecución. Cada uno de los *host* tienen su propio coordinador, el cual tiene que colaborar con los coordinadores de otros *hosts*, para garantizar una ejecución ordenada de los servicios.

- **Peer-to-Peer:** En este modelo, la responsabilidad para coordinar las ejecuciones de un servicio compuesto es distribuida a través de los proveedores de los servicios componentes, el cuál interactúa de una forma peer-to-peer sin encaminar mensajes a través de un planificador central. El ambiente de ejecución del servicio compuesto por consiguiente se manifiesta en forma de una colección de módulos interconectados los cuales se comunican a través de un protocolo convenido. Esta modelo de ejecución soporta algunas similitudes con los modelos de ejecución de workflows distribuidos.

7.2 Fiabilidad de la Ejecución

Dado que un servicio compuesto puede comprometer una mezcla de varios servicios autónomos, la fiabilidad del servicio compuesto resulta significativa. Como ya explicamos en la sección 4.4 no existe una definición clara de fiabilidad que establece el alcance en el contexto de un servicio compuesto; sin embargo a continuación describimos los aspectos de fiabilidad que deben ser considerados durante la ejecución del servicio, basados en el trabajo presentado en [17].

- **Soporte de transacciones:** Los mecanismos de tolerancia a fallos pueden tratar con errores que pueden ocurrir durante la ejecución del servicio compuesto de una manera apropiada. La forma de tratar el error puede depender de su clase y su efecto sobre la ejecución del servicio compuesto. Una propuesta general para manejar tales requerimientos es usar mecanismos transaccionales. WS-CAF y WS-Transactions son los estándares por referencia en los servicios Web. Además, otras propuestas pueden ser usadas para permitir transacciones sobre los servicios Web [14,15]. En [18] se propone una arquitectura multi-capa y un modelo de transacciones para construir servicios compuestos confiables. [17] propone una estructura para construir composición transaccional de servicios Web.

- **Intercambio fiable de mensajes:** La entrega fiable de mensajes entre los servicios Web es un factor importante que influye en la ejecución correcta del servicio compuesto, por consiguiente tiene relación directa con la fiabilidad global del servicio compuesto. Existen algunos esfuerzos que permiten entregar en forma fiable mensajes entre Servicios Web. Por ejemplo, *WS-ReliableMessaging* es un modelo que garantiza que los mensajes enviados por el remitente inicial serán entregados al último receptor.

7.3 Escalabilidad

La escalabilidad se refiere a la capacidad del sistema para crecer en una o más dimensiones tales como: volumen de datos accesibles, número de transacciones soportadas por unidad de tiempo, y el número de relaciones que pueden ser soportadas. Dentro del contexto de la composición de servicios, la escalabilidad es un factor a tomar en cuenta, debido a que no es igual componer 2 servicios, que 10 ó 100. En un escenario real, una plataforma de composición tiene que interactuar con algunos servicios, por lo tanto un aspecto crítico es conocer como las propuestas escalan con el número de servicios envueltos. Los problemas (y soluciones) que podemos localizar en una plataforma de composición de servicios para escalar los servicios que soporta son similares a aquellos para escalar la infraestructura Web en general.

Una solución es utilizar intermediarios (agentes) que pueden ocuparse de estas necesidades haciendo disponible los servicios a partir de un número de dispositivos. Los intermediarios pueden clasificarse en dos categorías: funcionales y optimización. Los intermediarios funcionales hacen los servicios disponibles haciéndose responsable de (las partes de) éstos. Por otra parte los intermediarios de optimización enriquecen los servicios sacando provecho de ciertos aspectos de su semántica, para: 1) eliminar o retrasar las conexiones al servidor origen; 2) reducir el número de bytes enviados a/desde el servidor origen.

7.4 Seguridad

La seguridad es una preocupación fundamental en diferentes tipos de aplicaciones más aún cuando las aplicaciones requieren de servicios que provienen de diferentes orígenes como es el caso de la composición de servicios Web; por esta razón es necesario contar con mecanismos de seguridad en los que exista autenticación mutua, integridad de la comunicación, confidencialidad, no repudiación y autorización.

Una plataforma de composición de servicios debe poder determinar las seguridades soportadas y requeridas por un servicio individual. Los servicios Web deben por lo tanto documentar sus requerimientos y soportar transacciones, seguridad y fiabilidad de mensajes. WS-POLICY [26] permite que los servicios Web documenten qué transacciones, seguridad y funciones de fiabilidad soportan o requieren. El protocolo WS-Security por otra parte contiene las especificaciones de cómo la integridad y confidencialidad pueden ser obligados en el envío de mensajes entre servicios Web. WSS incluye detalles sobre el uso de SAML y Kerberos y formatos de certificación como X.509.

En lo referente al control de acceso, la mayor parte de las propuestas se basan en autenticación, a través de un control central (registro) y una prueba de identidad; lo cual presenta ciertas limitaciones. En [21] proponen la idea de otorgar derechos de acceso y decisión basados en estructuras de confianza independientes. Ellos muestran cómo las credenciales pueden ser usadas en la Web semántica y cómo *Simple Public Key Infrastructure/Simple Distributed Security Infrastructure (SPKI/SDSI)* puede ser usado para especificarlas. Además introducen un álgebra de políticas que permite un control de acceso más complejo y eficiente. En [22], se definen políticas y contratos que son aplicados a arquitecturas orientadas al servicio. Ellos introducen brevemente el lenguaje *Knowledgeable Agent-Oriented System (KAoS)*, el cual es utilizado en diferentes aspectos de seguridad de los servicios Web. En [23] proponen una arquitectura para incorporar anotaciones de seguridad en las descripciones de servicios OWL-S. Los autores ilustran cómo usar el lenguaje de políticas *Rei* para aumentar el modelo de proceso OWL-S, y ofrecen una estructura para poner en ejecución las políticas definidas en un ambiente de ejecución OWL-S.

7.5 Adaptabilidad

Este aspecto hace referencia a la habilidad del servicio compuesto para manejar cambios dentro de los servicios individuales sin producir errores inesperados. En [13] por ejemplo se tiene una capa de funcionalidad adicional sobre los estándares de composición que permiten manejar este aspecto. Este usa un modelo conversacional de interacción entre los servicios Web para el manejo de cambios en la interfase de los servicios individuales.

8. Comparación de Plataformas de Composición de Servicios Web

En esta sección, comparamos algunas plataformas de composición de servicios, adicionalmente relacionamos éstas con las diferentes técnicas, modelos y lenguajes estudiados en las secciones 3 al 7 (ver Tabla 1).

Tabla 1. Comparación de Plataformas de Investigación para Composición de Servicios Web

PARÁMETROS	SHOP2	MENTOR-LITE	EFLOW	METEOR-S	SELF-SERV	SYNTHY
Características generales	Utiliza HTN (Hierarchical Task Network) para la composición de servicios.	Los workflows son especificados en términos de diagramas de actividades y estados	Soporta la composición de e-services manejado como procesos.	Descubrimiento dinámico de servicios en base a la semántica de cada actividad	Utiliza un mecanismo de ejecución distribuido.	Desacopla la composición de servicios en una composición lógica y física.
Lenguaje de Descripción Servicios: -WSDL. -WSDL con WS-CDL ó WSCI. -WSDL Mejorado	WSDL anotado semánticamente con DAML-S	No utiliza WSDL	No utiliza WSDL	WSDL anotado semánticamente	Se asume que los servicios están especificados en WSDL, sin embargo no esta restringido a este lenguaje.	Los servicios son descritos mediante OWL-S
Lenguaje de Composición de Servicios: -BPEL4WS -ebXML BPSS -WSCI -BPML	No se especifica	La composición de servicios es modelado mediante diagramas de estado y actividades.	La composición de servicios es modelado por un grafo, definiendo el flujo de servicios de forma similar a un diagrama UML	BPEL		Para la ejecución el servicio es traducido a BPEL

PARÁMETROS	SHOP2	MENTOR-LITE	EFLOW	METEOR-S	SELF-SERV	SYNTHY
Lenguaje formal de Composición de Servicios: -OWL-S. -Componentes Web. -Composición de Procesos Algebraica. -Redes de Petri. -Diagramas de Estado. -Inspección de Modelos. -Roman Model. -Máquinas Mealy.	DAML-S (ahora conocido como OWL-S)	Diagramas de Estado y actividades	-	No utiliza ningún lenguaje formal, únicamente para describir los detalles de la QoS se utiliza WSEL (Web Services Endpoint Language).	Diagramas de Estado	OWL-S
Modelo de Compatibilidad: -Sintáctico. -Semántico.	No se especifica	No se especifica	Sintáctico	Sintáctico	Sintáctico	Sintáctico y un nivel muy básico semánticamente.
Fiabilidad de la Composición: - Verificación de la Especificación del Servicio Compuesto. - Acuerdo entre la Especificación del Servicio Compuesto y los servicios individuales.	No se aborda	No se detalla ninguna característica.	Se controlan ciertos aspectos de la especificación del servicio compuesto.	No se aborda.	No se detalla ninguna característica.	Se controlan ciertos aspectos de la especificación del servicio compuesto.

PARÁMETROS	SHOP2	MENTOR-LITE	EFLOW	METEOR-S	SELF-SERV	SYNTHY
Estrategia de Composición: -Estática/ Dinámica -Manejada por Modelos y Reglas de negocio -Declarativa -Semántica (ontologías).	Declarativa: Planeación con IA y uso de Ontologías	Dinámica	Manejada por Modelos	Semántica	Manejada por Modelos	Declarativa: Planeacion con IA y uso de Ontologias
Descubrimiento de Servicios: -UDDI (Sintácticamente) -Descubrimiento semántico -Peer to peer	No se especifica	No se especifica	Propio mecanismo de búsqueda, analiza los requerimientos del usuario para construir parámetros de entrada para descubrir servicios.	Peer to peer, semántico.	UDDI, pero la búsqueda se efectúa en el marco de una comunidad de servicios.	Descubrimiento semántico mediante: Web Services Matchmaking Engine
Diferentes Técnicas de optimización	-	-	-	-	-	Utiliza técnicas de optimización para categorizar los workflows abstractos y concretos en base a los requerimientos no-funcionales.
Mecanismo de Orquestación: -Centralizada -Peer to peer	Centralizada	Distribuido	Centralizada	Centralizada	Distribuido	Distribuido
Confiabilidad de la ejecución: -Soporte de transacciones -Intercambio seguro de mensajes	No aborda	Soporta un *log manager* como mecanismo de recuperación	Soporte de transacciones a nivel de regiones, lo que permite que cierta región sea ejecutada como un proceso atómico	.No aborda	No aborda	No aborda
Escalabilidad: -Diferentes propuestas	No aborda	Los workflows son subdivididos en varios sub-workflows y entonces distribuidos	Motores distribuidos de anunciación de servicios	No aborda	Modelo de ejecución peer-to-peer	Modelo de ejecución peer-to-peer

PARÁMETROS	SHOP2	MENTOR-LITE	EFLOW	METEOR-S	SELF-SERV	SYNTHY
Seguridad	No aborda	No aborda	No aborda	No aborda	No aborda	No aborda
Adaptabilidad		El uso DBMS para conservar los registros del workflow provee ciertos niveles de escalabilidad.	Provee plantillas de proceso, nodos de servicio, y repositorio de datos de servicio para rehusar y controlar la adaptabilidad.	Semantic Process Templates pueden ser usados para manejar ciertos aspectos de adaptabilidad.		La utilización de servicios a nivel lógico y físico provee cierta adaptabilidad.
Manejabilidad Externa	No aborda	No aborda	Rastreo de eventos	No aborda	No aborda	No aborda

Bibliografía

[1] Chakraborty, D. and A. Joshi. (2001). Dynamic Service, Composition: State-of-the-art and Rsearch Directions. Technical Report TR-CS-01-19, CSEE, UMBC.

[2] Srivastava, B. and Köhler, J. (2003). *Web Service Composition – Current Solutions and OpenProblems*, http://www.zurich.ibm.com/pdf/ebizz/icaps-ws.pdf.

[3] Jinghai Rao and Xiaomeng Su (2003). A Survey of Automated Web Service Composition Methods, Norwegian University of Science and Technology, Department of Computer and Information Science, Trondheim, Norway

[4] Nikola Milanovic and Miroslaw Malek (2004). *Current Solutions for Web Service Composition*, Humboldt University, Berlin.

[5] Dustdar, S. and Schreiner, W. (2005). A survey on web services composition, *Int. J. Web and Grid Services*, Vol. 1, No. 1, pp.1–30.

[6] Brahim Medjahed (2004). Semantic Web Enabled Composition of Web Services, Dissertation submitted to the Faculty of the Virginia Polytechnic Institute and State University, January, Falls Church, Virginia, USA.

[7] Daniela Berardi (2005). Automatic Service Composition. Models, Techniques and Tools, Universit`a degli Studi di Roma "La Sapienza", Dottorato di Ricerca in Ingegneria Informatica.

[8] Boualem Benatallah, et al. (2005). Service Composition: Concepts, Techniques, Tools and Trends, Chapter III, pp 45-66.

[9] J. Hanson, P. Nandi, and S. Kumaran. Conversation Support for Business Process Integration. In *Proceedings of 6th Int'l Enterprise Distributed Object Computing (EDOC02)*, September 2002.

[10] Brahim Medjahed, et al. (2003). *Composing Web services on the Semantic Web.* VLDB Journal.

[11] Brahim Medjahed, et al. (2005). *A Multilevel Composability Model for Semantic Web Services.* IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 7,

[12] K. Sivashanmugam, J. Miller, A. Sheth, and K. Verma (2004), Framework for Semantic Web Process Composition, International Journal of Electronic Commerce, Winter 2004-5, Vol. 9(2) pp. 71-106

[13] Wolf-Tilo Balke, Matthias Wagner (2004), Through Different Eyes – Assessing Multiple Conceptual Views for Querying Web Services, *WWW 2004*, May 17-22, 2004, New York, NY USA.

[14] WSPDS: Web Services Peer-to-peer Discovery Service Farnoush Banaei-Kashani, Ching-Chien Chen, and Cyrus Shahabi, Computer Science Department, University of Southern California,

[15] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services," *Journal of Information Technology and Management*, under review.

[16] R. Hamadi and B. Benatallah. A Petri-Net-Based Model for Web Service Composition. *Proc. 14th Australasian Database Conf. Database Technologies*, ACM Press, 2003, pp. 191–200.

[17] X. Fu, T. Bultan, and J. Su. Formal Verification of Eservices and Workflows. *Proc. Workshop on Web Services,E-Business, and the Semantic Web* (WES), LNCS 2512, Springer-Verlag, 2002, pp. 188–202.

[18] D. Berardi et al., Automatic Composition of E-Services that Export Their Behavior, *Proc. 1st Int'l Conf. Service-Oriented Computing* (ICSOC 03), LNCS 2910, Springer-Verlag, 2003, pp. 43–58.

[19] A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara, "DAML-S: Web service Description for the Semantic Web", *Proceedings of the 1st International Semantic Web Conference* (ISWC 2002).

[20] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In Proc. 1st Int. Conf. on Service Oriented Computing (ICSOC), volume 2910 of LNCS, pages 43–58, 2003.

[21] S. Agarwal, B. Sprick and S.Wortmann. Credential Based Access Control for Semantic Web Services. American Association for Artificial Intelligence (www.aaai.org), 2004.

[22] A. Uszok, J. M. Bradshaw, M. Johnson, R. Jeffers, A.Tate, J. Dalton, and S.Aitken, KAoS Policy Management for Semantic Web Services, IEEE INTELLIGENT SYSTEMS, 2004.

[23] L. Kagal, T. Finin, M. Paolucci, N. Srinivasan, K. Sycara, and G. Denker. Authorization and Privacy for Semantic Web Services, IEEE INTELLIGENT SYSTEMS, 2004.

[24] P. Álvarez, J.A. Bañares, and J. Ezpeleta. Approaching Web Service Coordination and Composition by Means of Petri Nets. The Case of the Nets-Within-Nets Paradigm. ICSOC 2005, LNCS 3826, pp. 185–197, 2005.

[25] Santhosh Kumaran and Prabir Nandi. *Conversation Support for Web Services*. Accessed on: 15 June 2003.
http://www-106.ibm.com/developerworks/webservices/library/ws-conver/

[26] WS-Policy: http://www-128.ibm.com/developerworks/library/specification/ws-polfram/