DESCRIPCIÓN, LOCALIZACIÓN Y COMPOSICIÓN SEMÁNTICA DE SERVICIOS WEB COMO SOLUCIÓN PARA EL SISTEMA DE GESTIÓN DE CUENTAS DE USUARIO DE LA RED DE DATOS DE LA UNIVERSIDAD DEL CAUCA

Trabajo de grado presentado como requisito para optar el título de Ingeniero en Electrónica y Telecomunicaciones

Javier Ernesto Burbano Sandoval
Jaime Andrés Cubillos Patiño

Director: Ing. Juan Carlos Corrales Muñoz

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Sistemas Distribuidos y Servicios Web
Popayán, 2.005

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La aparición de Internet causó una revolución mundial que aún persiste en su auge. Partiendo de las primeras iniciativas encaminadas a construir una red de comunicación de computadores, el desarrollo y la expansión de la red de redes es un proceso que hasta hoy permanece ininterrumpido.

En un sentido amplio, el más notable impacto producido por Internet se deriva de la increíble disponibilidad de información de cualquier tipo en la *World Wide Web*. Ante un mundo con fuerte orientación hacia el avance tecnológico, es fácil reconocer que aún tal ventaja es susceptible de ser ampliada y mejorada.

En esa tendencia de desarrollo y adaptabilidad, la tecnología de los servicios web se abre paso como una alternativa propicia para llevar a cabo una nueva revolución, que apunta a dotar a la web de las capacidades de una plataforma de prestación de servicios. Se busca también que sobre esta plataforma, se establezca un nuevo campo de acción para la computación distribuida, la cual sin duda aprovechará la enorme difusión de Internet, los modelos estándar de comunicación y la independencia de plataforma que ostenta.

En un campo afín a los mencionados, la iniciativa de la *web semántica* propende por mejorar la usabilidad de los recursos disponibles en la red por medio de un nuevo modelo conceptual en el cual, cada uno de estos recursos presenta no sólo información de contenido y localización, sino también información estructurada que lo describe a sí mismo, a su contenido o modo de uso, y además lo relaciona con otros. Este modelo toma como base algunos conceptos de la inteligencia artificial y los adapta para su uso en el entorno de Internet.

Un área específica del desarrollo de los servicios web es la *composición de servicios*. En términos generales, componer servicios significa emplear mecanismos que les permiten colaborar en la consecución de metas que van más allá de sus capacidades individuales. La importancia de la composición de servicios y de su estandarización se evidencia

gracias al gran potencial de los servicios web para servir como alternativa tecnológica para la integración de aplicaciones que se ejecutan en ambientes heterogéneos y se comunican por medio de protocolos de red estándar.

El presente documento refleja el trabajo de investigación realizado en el área originada por la convergencia entre las dos iniciativas mencionadas antes, conocida como *servicios web semánticos*. Uno de los propósitos principales de este proyecto fue traer al contexto investigativo y tecnológico de la Universidad del Cauca y la FIET, las alternativas emergentes en cuanto a la explotación y composición de servicios web por medio de las tecnologías de la web semántica. Se utilizó una de estas alternativas para desarrollar una aplicación prototipo para la migración de la aplicación de Gestión de Cuentas de Usuario de la Red de Datos de la Universidad a una soportada en una arquitectura basada en servicios web.

La temática del documento inicia con una revisión de la arquitectura de los servicios web, su significado y su potencialidad como tecnología de integración de aplicaciones. A continuación se trata el concepto de composición de servicios y sus dos principales perspectivas. Se presentan las ideas generales alrededor de la web semántica y su punto de unión con la composición de servicios.

El informe continúa con la descripción general de cada una de las alternativas tecnológicas disponibles, resaltando sus ventajas, desventajas y expectativas de evolución. Presenta luego un análisis comparativo en el cual se establecen los criterios tecnológicos que fueron utilizados como base en la selección de una alternativa para el desarrollo de la aplicación prototipo. Finalmente incluye el análisis efectuado para la implantación de los diferentes servicios componentes de la aplicación y se describe la forma en la cual colaboran para ofrecer la funcionalidad requerida.

1. SERVICIOS WEB

1.1 ARQUITECTURA

En términos generales, los servicios web proporcionan una forma estandarizada para facilitar tanto la **integración** como la **interacción** entre aplicaciones informáticas que se ejecutan en plataformas heterogéneas y en entornos distribuidos.

El propósito de la Arquitectura de los Servicios Web es establecer una definición genérica de servicio web y un modelo conceptual, así como las tecnologías y las especificaciones relacionadas con los servicios web. La arquitectura no especifica ni impone restricciones en cuanto a la implementación de los servicios ni el modo en el cual ellos deberían ser combinados. En ella se describen las características mínimas que son comunes a todos los servicios web.

1.1.1 Definición de Servicio Web

"Un servicio web es un componente software diseñado para soportar y ejecutar la interacción entre sistemas informáticos heterogéneos a través de una red. Este tiene una interfaz descrita en un formato procesable por la máquina. Otros sistemas interactúan con un servicio web de la forma indicada en su descripción utilizando mensajes estándar. Los mensajes son típicamente llevados combinando un protocolo de transporte de Internet y un mecanismo de serialización ampliamente aceptado." [WSAR04]

A continuación se enuncian conceptos clave que permitirán más adelante asimilar claramente la idea general detrás de la utilización de un servicio web.

1.1.1.1 Agentes

Un servicio web en sí mismo es un *concepto* o una *noción abstracta*. La funcionalidad que se pretende ofrecer a través de la publicación de un servicio web debe ser implementada en forma tangible por un ***agente***.

Un agente es un programa informático que envía y recibe mensajes, mientras que un servicio es un *recurso* caracterizado por el conjunto abstracto de funcionalidades que ofrece. Un mismo servicio web puede ser implementado por agentes diferentes. Cada agente podría estar desarrollado en un lenguaje de programación diferente.

1.1.1.2 Proveedores y Solicitantes

Uno de los propósitos buscados al separar la noción de servicio de la implementación concreta, es decir, del agente, es poder asociar en forma clara un servicio con su *propietario*. El propietario de un servicio puede ser una **persona** o una **organización**. La distinción entre el propietario de un servicio y la organización o persona que hace uso del servicio se logra a través de los siguientes conceptos:

Entidad Proveedora: es la persona u organización que provee un agente que materializa un servicio en particular. En consecuencia, la lógica detrás del servicio se encuentra en esta entidad.

Entidad Solicitante: es la persona u organización que pretende utilizar el servicio web ofrecido por la entidad proveedora. La entidad solicitante debe tener un *agente solicitante* que intercambie mensajes con el *agente proveedor* de la entidad proveedora.

1.1.1.3 Descripción de un Servicio Web

La *Descripción de un Servicio Web* es la especificación de su interfaz escrita en un lenguaje de descripción de interfaces. El Lenguaje de Descripción de Servicios Web *WSDL (WSDL: Web Services Description Language)* es actualmente el estándar de industria utilizado para este propósito.

La descripción del servicio define *el formato de los mensajes*, *los tipos de datos*, *los protocolos de transporte* y *los mecanismos de serialización* que deben ser utilizados por los agentes solicitante y proveedor para interactuar correctamente.

1.1.1.4 Semántica de un Servicio Web

Hablar de la *semántica de un servicio web* implica hablar de una expectativa, específicamente, de lo que *"se espera"* del comportamiento del servicio como respuesta a los mensajes que recibe.

Mientras que *la descripción de un servicio* representa un *acuerdo* que dirige la mecánica de la interacción con ese servicio, la *semántica* representa un *acuerdo* que gobierna el significado, el propósito y las consecuencias de esa interacción. El acuerdo sobre la semántica no necesariamente está escrito o se negocia en forma manifiesta, es decir, este podría ser explícito o implícito, orientado a la máquina u orientado a personas.

Un aspecto para resaltar es que la línea de división entre los dos tipos de acuerdo no es totalmente rígida. La utilización de *lenguajes semánticos* para describir la mecánica de la interacción con un servicio permitiría consignar la semántica de un servicio en su propia descripción, esto es, permitirá la creación de *descripciones semánticas*.

1.1.2 Utilización de un servicio web

Expuestos los conceptos relevantes de la arquitectura y su definición, por medio de la Figura 1-1 se ilustra que para lograr la utilización de un servicio web se requiere:

1. La entidad solicitante conozca a la entidad proveedora y viceversa (por lo menos lo primero). Este proceso puede lograrse por medio de un *mecanismo de localización de servicios*. El estándar de industria para este propósito es el Registro Universal de Descripción, Localización e Integración *(UDDI: Universal Description, Discovery and Integration)*.

2. La entidad solicitante y la entidad proveedora lleguen a un acuerdo acerca de la descripción y la semántica del servicio que dirigirá la interacción entre los agentes de cada entidad.

3. La descripción y la semántica del servicio sean realizadas por los agentes de cada entidad.

4. El agente solicitante y el agente proveedor intercambien mensajes, es decir, ejecuten alguna tarea a nombre de la entidad solicitante y la entidad proveedora, respectivamente. El estándar de industria para el intercambio de mensajes es el Protocolo Simplificado de Acceso a Objetos *(SOAP: Simple Object Access Protocol)*.

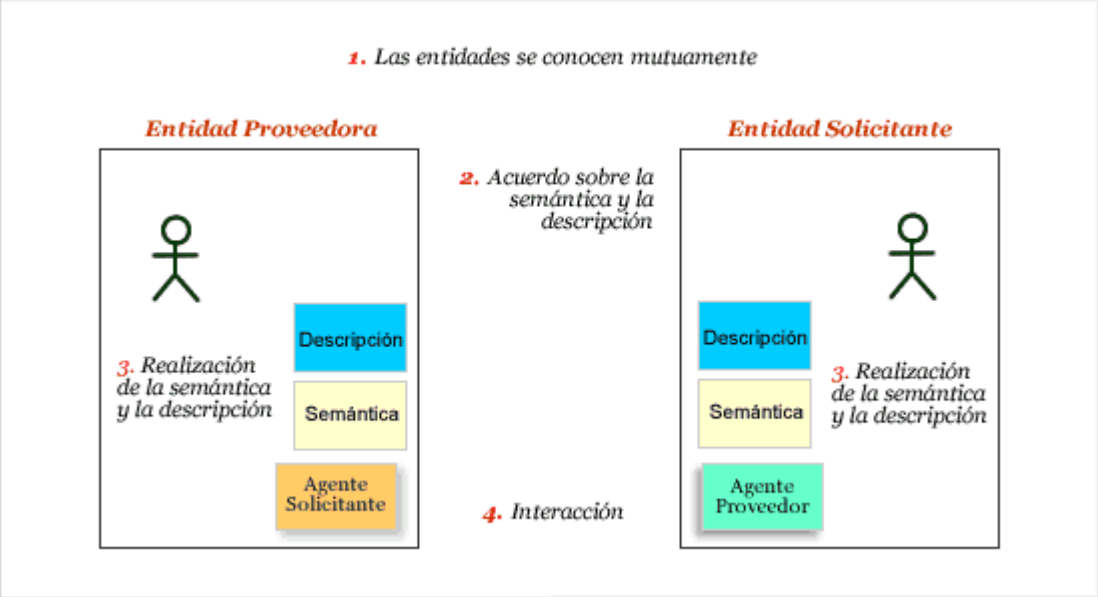

Figura 1-1 Utilización de un Servicio Web [WSAR04]

1.1.3 Vista por Niveles de la Arquitectura de los Servicios Web

En la Figura 1-2 se muestra una vista por niveles de la arquitectura de los servicios web así como las diferentes tecnologías asociadas.

1. Mensajería y Transporte de Datos: en este grupo se enmarcan las tecnologías y protocolos que permiten el intercambio de datos entre servicios web.

2. Descripción: contiene los lenguajes para la descripción de interfaces.

3. Publicación y Localización: incluye los lenguajes y mecanismos para la publicación y localización de servicios web.

4. Composición y Coreografía: es un nuevo grupo que abarca los lenguajes de descripción de flujos, coreografías y composiciones de servicios.

5. Elementos transversales: involucra mecanismos que tratan características adicionales de relevancia en todos los niveles de la arquitectura. Se resaltan Seguridad, Gestión y Calidad de Servicio.

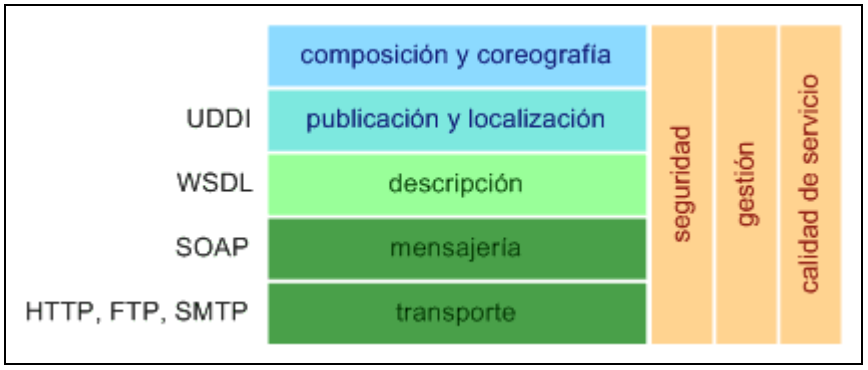

Figura 1-2 Vista por Niveles de la Arquitectura [WSCA01]

1.2 TECNOLOGÍAS DE SERVICIOS WEB

1.2.1 SOAP

En su definición más simple, SOAP es un protocolo liviano que utiliza el lenguaje extensible de marcado *(XML - Extensible Markup Language)* para efectuar intercambio de información estructurada en ambientes distribuidos.

Las metas de diseño de SOAP fueron la **simplicidad** y la **extensibilidad**. En virtud de la primera, la especificación del protocolo omite características del intercambio de mensajes que frecuentemente se requieren en sistemas distribuidos, por ejemplo *seguridad*, *confiabilidad*, *correlación de mensajes*, *enrutamiento* o *patrones de intercambio de mensajes*. La segunda exige que estas características y probablemente otras sean cubiertas por *otras especificaciones* que cumplan los requisitos para ser *extensiones* de SOAP.

La especificación del protocolo SOAP está dividida en dos grandes partes:

- Una **infraestructura de mensajería** diseñada para ser independiente de cualquier modelo de programación o de otras semánticas específicas de implementación. Esto quiere decir que las aplicaciones que se ejecuten en ambientes distribuidos pueden usar a SOAP como medio transparente de transporte de mensajes [SOMF03].

- Un **grupo de adjuntos** que pueden ser utilizados en conjunto con la infraestructura de mensajería. De estos se pueden resaltar: un modelo de datos específicos de aplicación, un conjunto de reglas de codificación para este modelo de datos, una convención para representar llamados y respuestas a procedimientos remotos entre otros [SOMF03].

Un mensaje SOAP se asemeja a una carta, puesto que está formado por un **sobre** que lleva información del destinatario y un **cuerpo** que lleva su contenido. Así mismo, el mensaje puede tener **encabezados** que contienen información adicional que puede o no ser dirigida al destinatario del mensaje. Concretamente, el mensaje SOAP es un documento XML formado por los elementos obligatorios que conforman su estructura y por los elementos que constituyen el contenido del mensaje, los cuales deben cumplir con las reglas XML pero son contenido arbitrario.

El transporte de un mensaje SOAP puede hacerse sobre una gran variedad de protocolos de transporte de amplia distribución en Internet. El conjunto de reglas que define los mecanismos de transmisión de un mensaje SOAP sobre cualquier protocolo se conoce con el nombre de *enlace de protocolo*. Algunos de los protocolos que pueden ser usados

son HTTP, SMTP y FTP, de los cuales el primero es el más empleado. Las reglas definidas en el enlace de SOAP para HTTP aprovechan las características de interacción del tipo solicitud-respuesta soportado por HTTP.

La interacción entre un agente solicitante y agente proveedor de servicios web puede lograrse usando un patrón de intercambio de mensajes del tipo *Respuesta* o *Solicitud-Respuesta*. Esto es consistente con la característica de extensibilidad del SOAP, puesto que tales patrones se definen en el enlace de protocolo y no en el protocolo en sí.

El patrón *Respuesta* implica una invocación a una URL de un agente proveedor a través del método GET. Se conoce como patrón respuesta porque la interacción involucra la generación y transporte de un único mensaje SOAP.

El patrón *Solicitud-Respuesta* implica una invocación a una URL de un agente proveedor a través del método POST. En este caso, la interacción involucra la generación y transporte de dos mensajes SOAP, uno en el sentido solicitante-proveedor y otro en el sentido opuesto.

1.2.2 WSDL

El *Lenguaje de Descripción de Servicios* Web *(WSDL: Web Services Description Language),* es un lenguaje de marcado escrito en XML para describir en forma estándar la interfaz de un servicio web. Dicho de otro modo, WSDL es un Lenguaje de Descripción de Interfaces *(IDL: Interface Description Language)*.

Un documento de descripción de un servicio web define el formato de los mensajes, los tipos de datos, los protocolos de transporte y los formatos de serialización a utilizar entre el agente proveedor y el agente solicitante. El documento establece un *acuerdo* que gobierna la mecánica de la interacción con el servicio web. Un documento WSDL correctamente formado *describe* un servicio web desde el punto de vista de la *entidad proveedora*. [WSDL01].

La descripción de un servicio web escrita en un documento WSDL está dividida en dos partes: una ***abstracta*** y una ***concreta***.

Las premisas de la *descripción abstracta* de un servicio web son las siguientes:

- La *funcionalidad* de un servicio web está descrita en términos de los mensajes que envía y recibe.
- Los *mensajes* son descritos sin dependencia del protocolo de transporte, utilizando un sistema de tipos de datos, típicamente esquemas XML (*XMLS: XML Schema*).
- Una *operación* agrupa uno o más mensajes, describiendo una porción de la funcionalidad del servicio.
- Una *interfaz* describe los mensajes que un servicio web envía o recibe. Logra esto agrupando los mensajes relacionados en operaciones, es decir, una interfaz es un conjunto de operaciones.

Las premisas de la descripción concreta de un servicio web son las siguientes:

- Un *enlace* describe **cómo** acceder a un servicio web definiendo detalles de transporte para una o más de sus interfaces.
- Un *punto final* describe **dónde** acceder a un servicio web asociando un enlace con una dirección física.

Todas estas premisas se consignan en un documento XML de definiciones cuyos elementos principales son:

ELEMENTOS	DESCRIPCION
<definitions>	Elemento raíz del documento.
<part>	Define las partes de un mensaje en particular. El sistema de tipos de datos usado es XMLS.
<message>	Cada instancia de esta etiqueta define un mensaje, agregando las diferentes partes definidas para este.
<operation>	Es una **operación** abstracta.
<portType>	Agrega elementos <operation> formando la *definición abstracta* del servicio.
<binding>	O **enlace**, define el mecanismo de transporte concreto para un determinado elemento <portType>.

<port>	O **punto final**, define la dirección de red (una o más) específica para acceder a un servicio.
<service>	Agrega los dos elementos anteriores formando la definición concreta del servicio.

Tabla 1-1 Elementos WSDL

El siguiente ejemplo simplificado demuestra la estructura de un documento WSDL.

```
<?xml version="1.0"?>
<definitions>
   <message name="NombreMensaje">
      <part name="NombreParte" type="unTipodeDato">
   </message>
   <portType name="NombrePortType">
      <operation name="NombreOperacion" pattern="UnMEP">
         <!-- Mensajes relacionados con esta operación -->
      </operation>
      <operation name="NombreOperacion" pattern="UnMEP">
         <!-- Mensajes relacionados con esta operación -->
      </operation>
   </portType>
   <binding name="NombreEnlace" portType="NombrePT">
      <!-- Definición del enlace -->
   </binding>
   <service name="NombreServicio" interface="NombreInterfaz">
      <port name="NombrePunto" binding="NombreEnlace" address="unaURL" />
</service>
</definitions>
```

1.2.3 UDDI

1.2.3.1 Publicación y Localización de Servicios

Un servicio web no puede ser invocado si de antemano no se conoce su descripción. Esto quiere decir que para poder utilizar un servicio web, una entidad debe primero localizarlo y esto depende de que otra entidad lo haya publicado. Tanto los mecanismos para publicar servicios como los mecanismos para encontrarlos son variados. Básicamente, cualquier mecanismo que permita a una entidad tener acceso al documento de descripción de un servicio web se considera como un *mecanismo de localización*.

El problema implícito asociado con la localización de servicios es conocer, en un momento dado, qué entidades ofrecen cuáles servicios para luego escoger entre ellos aquel que es de utilidad.

1.2.3.2 Generalidades de UDDI

UDDI es una especificación para la creación de registros distribuidos de información de servicios web, precisamente basada en Internet. Por otro lado UDDI es también un conjunto de implementaciones de esta especificación, las cuales están disponibles para uso público. [UDDI01].

Un registro UDDI es algo así como un directorio electrónico que almacena en un formato XML común, información de organizaciones y de los servicios web que ellas ofrecen. En la mayoría de las ocasiones, las organizaciones utilizan este registro para localizar información de servicios que pueden ser útiles en el desarrollo de sistemas internos; por otro lado también pueden emplear el registro para publicar aquellos servicios web que pretenden ofrecer a otros.

El componente clave de UDDI es la *inscripción o matrícula de negocio*. La inscripción de negocio es un documento XML que describe a una entidad de negocio (una organización) y los servicios web que ella ofrece. Conceptualmente, la información que se publica en una inscripción de negocio está compuesta por tres partes:

- **Páginas Blancas:** incluyen nombre de la organización, dirección de ubicación e información de contacto como número telefónico, número de fax o dirección de correo electrónico.

- **Páginas Amarillas:** incluyen información de las organizaciones catalogada según taxonomías internacionales como el Sistema de Clasificación de la Industria Norteamericana *(NAICS: North American Industry Classification System)* o la Clasificación Industrial Estándar *(SIC: Standard Industrial Classificaciton)*. A manera de ejemplo, el código NAICS 3341 identifica a los fabricantes de equipos de cómputo.

- **Páginas Verdes:** incluyen información técnica de los servicios ofrecidos por la organización. En esta se presenta la categoría de cada servicio (basada en un *conjunto de tipos de servicio*) y referencias a las especificaciones de cada servicio.

1.3 INTEGRACIÓN DE APLICACIONES CON SERVICIOS WEB

Teniendo en mente la descripción de la arquitectura de los servicios web, es importante recalcar su papel como tecnología orientada a la interoperabilidad e integración de aplicaciones heterogéneas que se ejecutan en ambientes distribuidos.

Un sistema distribuido está constituido por diversos componentes software que deben actuar juntos para lograr objetivos comunes. La característica obvia y a la vez principal de este tipo de sistemas es que sus diversos componentes están distribuidos en una serie de nodos de cómputo que usan una infraestructura de red y algún mecanismo de comunicación. [WSA04].

En años anteriores surgieron diversas tecnologías como solución al requerimiento de *distribuir* la funcionalidad ofrecida por un sistema en un ambiente descentralizado de cómputo. Todas estas tecnologías se basaron en la filosofía de *objetos remotos* y lograron resolver el requisito planteado por medio de modelos complejos de interacción y protocolos de comunicación desarrollados a la medida. Sin embargo, cuando el requisito se modificó para requerir interacción entre sistemas distribuidos de diferente naturaleza se vio que todos ellos tenían una importantísima falencia: no eran compatibles. [WWS02].

En otro contexto y en forma casi paralela, Internet fue creciendo en un ritmo considerable, logrando un nivel de penetración considerable en universidades, empresas y organizaciones. A pesar de que la arquitectura en la cual se soporta Internet tiene grandes restricciones, el soporte a su modelo de trabajo y sus protocolos asociados es obligatorio en cualquier sistema de cómputo actual.

La convergencia entre las tecnologías de computación distribuida y la gran extensión de Internet es precisamente el punto de origen de los servicios web. Estos plantean una

Arquitectura Basada en Servicios que deliberadamente abstrae los detalles de implementación de los diferentes componentes software en una interfaz estándar descrita con datos procesables por la máquina y que confía los procesos de comunicación a mecanismos ampliamente aceptados y difundidos.

Como puede notarse en la definición de servicio web consignada en este documento, estos surgieron como una alternativa de interacción entre diferentes aplicaciones. La gran ventaja con respecto a los modelos de computación distribuida propietarios radica precisamente en la búsqueda de una interoperabilidad real, puesto que se apuesta por un modelo estándar de comunicación y por una independencia real de la plataforma de ejecución.

Las ventajas claves de los servicios web se relacionan en la siguiente tabla.

VENTAJAS CLAVE DE LOS SERVICIOS WEB	
Desligamiento de Plataforma	Es irrelevante la plataforma de cómputo, el modelo y el lenguaje de de programación usado para crear un servicio web, siempre y cuando se conserven las reglas de publicación, comunicación y descripción.
Accesibilidad	Los servicios web pueden ser completamente descentralizados y estar distribuidos en Internet y pueden ser consumidos desde una gran variedad de dispositivos de red.
Eficiencia	Las empresas pueden concentrarse en tareas que ofrezcan valor agregado y en tareas críticas, puesto que a partir del software ya disponible pueden publicar funcionalidades específicas en forma de servicios web, evitándose así el proceso de desarrollo completo de nuevas aplicaciones software.
Estandarización	Los servicios web se basan en tecnologías estandarizadas de comunicación, representación de datos y descripción de interfaces. Esto constituye un aporte significativo a la interoperabilidad.
Amplio Despliegue	Numerosas organizaciones y empresas líderes en el desarrollo de software han apostado por el rápido esparcimiento de los servicios web.

Tabla 1-2 Ventajas Clave de los Servicios Web

Además de las características ya mencionadas hasta aquí, es necesario tener en cuenta otros aspectos especiales en el desarrollo de nuevos tecnologías que complementen aquellas que hacen posible el desarrollo y publicación de servicios. Básicamente son tres los aspectos que recibirán gran atención:

- Seguridad: dado que el modelo de comunicación de los servicios web está basado principalmente en el protocolo HTTP, han surgido interrogantes en cuanto a la posibilidad de garantizar privacidad en los intercambios de información.

- Calidad de Servicio: se refiere a la posibilidad de dotar a los servicios de características especiales que identifiquen requisitos puntuales de calidad.

- Gestión: se refiere a la posibilidad de monitorear y controlar características de la provisión de servicios, por ejemplo, la misma calidad.

La conclusión inmediata a la que se puede llegar es que la tecnología de servicios web está todavía en una etapa temprana de desarrollo. Sin embargo, existen pistas importantes que permiten determinar el importante papel que tendrá en la evolución de las tecnologías de interacción e integración de aplicaciones.

El resto de este documento presenta el resultado de la investigación desarrollada en el aspecto del soporte a la colaboración de servicios web basada en tecnologías de la web semántica, enfocado principalmente al punto de vista de la organización que publica servicios para dar soporte a sus propios procesos de negocio.

2. COMPOSICIÓN DE SERVICIOS WEB

Componer servicios significa establecer mecanismos que permitan a dos o más de ellos cooperar entre sí para resolver requisitos que van más allá del alcance de sus capacidades individuales.

En un documento publicado por la Corporación para las Ciencias de la Computación (*CSC – Computer Sciences Corporation*) en el 2.002 [COSC02], se identificó que el gran reto en cuanto a la gestión de los sistemas de información de las compañías era el poder **conectarse electrónicamente con proveedores, clientes y socios** como mecanismo para incrementar la eficiencia de procesos de negocio y lograr aumento general de los beneficios.

Con el advenimiento de la tecnología de servicios web como alternativa novedosa de *computación distribuida* y su importante crecimiento y expansión, fue evidente la necesidad de preparar los sistemas de información para hacer uso y sacar provecho a esta nueva generación de aplicaciones basadas en Internet.

Si bien los servicios web nacen de una *Arquitectura Orientada a Servicios*, ellos por sí solos no representan una alternativa robusta y estandarizada para lograr una interacción eficiente entre aplicaciones distribuidas que se ejecutan en ambientes heterogéneos; sin embargo, son un primer paso invaluable.

Poder retomar los conceptos de Integración de Aplicaciones Empresariales y de interacciones B2B (*Business to Business*) en forma de mecanismos estandarizados basados en servicios web es precisamente el objetivo perseguido por la composición de servicios.

Para lograr tal cosa, la composición de servicios debe cumplir algunos requerimientos técnicos, de los que destacamos [WSOR03]:

- La posibilidad de efectuar interacciones asincrónicas con servicios es indispensable, puesto que de esta forma se habilita la posibilidad que un proceso de negocio use varios servicios en forma concurrente. Así por ejemplo, un sistema de compra podría solicitar múltiples cotizaciones en forma simultánea y no en forma secuencial. Tal capacidad tiene que ver directamente con el rendimiento en la ejecución.

- La posibilidad de garantizar un adecuado manejo de excepciones, puesto que un proceso que se base en múltiples servicios puede fallar en su ejecución si al menos uno sus servicios componentes no está disponible. La declaración de un proceso debe tener en cuenta la posibilidad de ocurrencia de errores y además debe incluir mecanismos de manejo de errores para tales escenarios.

- La posibilidad de garantizar integridad transaccional, debido a que durante su ejecución, un proceso puede encontrarse ante una situación que implique deshacer parte del trabajo ya completado. Así por ejemplo, un proceso que cubra una reservación de tiquetes de avión puede estar en estado de espera de una confirmación por parte de una aerolínea (interacción asincrónica), lo cual puede tomar un tiempo considerable bajo condiciones especiales. Si durante ese tiempo de espera el viajero desea cancelar su reservación, deben tomarse acciones para liberar lo que se haya logrado reservar y cobrar las multas correspondientes. Este tipo de acciones se conocen como acciones de compensación.

Lograr un mecanismo estandarizado para la composición de servicios web que cumpla con estos requerimientos posibilitará mejorar notablemente las capacidades de Internet para actuar como plataforma de integración de aplicaciones.

2.1 ORQUESTACIÓN Y COREOGRAFÍA DE SERVICIOS WEB

Los dos términos usados por excelencia para referirse a la colaboración entre varios servicios web son *orquestación* y *coreografía*. Ambos están relacionados directamente con la composición de servicios pero se enfocan en porciones diferentes de la interacción

entre servicios web, es decir, son complementarios. El resultado de la composición de servicios web se conoce como **proceso web** o simplemente proceso.

El primer punto de vista para analizar la composición de servicios es el interno de una organización, la cual, puede publicar una serie de servicios web para lograr un objetivo de negocio interno. El segundo punto de vista es el de procesos (o servicios individuales) de dos o más organizaciones distintas y que colaboran entre sí para lograr un objetivo de negocio complementario de más alto nivel. Son estos dos escenarios los que se explican a continuación para entender claramente los conceptos de orquestación y coreografía.

2.1.1 Orquestación

Un proceso web es de *orquestación de servicios* cuando es controlado totalmente por una única entidad u organización. El proceso sin embargo, puede usar servicios que están al interior de la organización o servicios que son ofrecidos por entidades externas.

En un proceso de orquestación de servicios se definen completamente las interacciones del proceso con los servicios componentes a nivel de los mensajes, el orden de tales interacciones y la lógica del negocio asociada, todo desde el punto de vista de uno de los participantes en la interacción. [WSOR03].

Un proceso de orquestación de servicios se puede entender como uno **privado** y **ejecutable**. Es privado porque la definición de la lógica del proceso es hecha enteramente por un participante en la interacción, aún cuando éste vaya a hacer parte de uno más grande. Por otro lado es **ejecutable** porque tiene un comportamiento de conversión de entradas en salidas y tiene efectos en el mundo real.

En la figura 2-1 se muestran dos procesos de orquestación de servicios. Se supone un escenario donde participan una **Entidad Compradora** y una **Entidad Proveedora**. Cada una de ellas cuenta con un sistema ERP (*ERP: Enterprise Resource Planning*) para la gestión de recursos y expone funcionalidad de servicios web para interactuar con su contraparte.

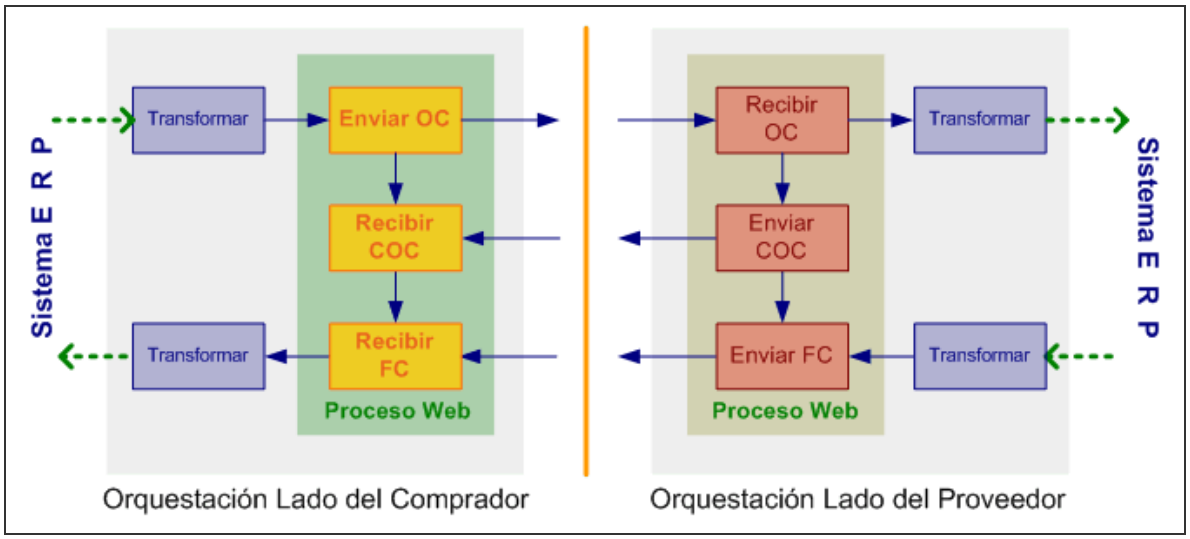

Figura 2-1 Orquestación de Servicios

En la Entidad Compradora el sistema ERP nota la falta de algún producto y notifica al proceso de orquestación. Éste genera una orden de compra (OC), la envía a la Entidad Proveedora y queda a la espera de una confirmación de recepción de la orden (COC). Una vez recibida la confirmación, el proceso espera la respuesta a la orden de compra, es decir, la Factura de Compra y Notificación de Entrega (FC), para ser retornada al sistema ERP.

En el otro lado, el proceso de orquestación de la Entidad Proveedora también se comunica con su propio sistema ERP para efectuar la compra. En este lado, el proceso inicia recibiendo una orden de compra, respondiendo con una confirmación de recepción al tiempo que envía la orden al sistema interno y quedando a la espera de la respuesta a la orden para ser reenviada a la Entidad Compradora.

Lo que es realmente importante en esta figura es notar cómo cada organización participante en la interacción ajusta un proceso que usa servicios web para interactuar con su contraparte. Cada entidad implementa y controla su propio proceso al tiempo que ignora totalmente cómo está implementado el de su contraparte.

2.1.2 Coreografía

Un proceso web es de *coreografía de servicios* cuando define las colaboraciones entre cualquier tipo de aplicaciones componentes, independientemente del modelo de programación o de la plataforma de soporte de cada una de ellas. Un proceso de coreografía no es controlado por uno solo de los participantes de la interacción y representa más bien un acuerdo para la implementación de una interacción entre aplicaciones.

En el contexto de la composición de servicios web, la coreografía puede entenderse como un proceso **público** y **no ejecutable**. Es público porque define el comportamiento mutuamente visible, en el nivel de los mensajes, entre los diferentes participantes en una interacción. Por otro lado es no ejecutable o abstracto porque no está pensado para ser llevado a cabo, sino que está hecho para actuar como un *protocolo de negocio* que dicta reglas de interacción que deben ser cumplidas por las entidades participantes y por aquellas que deseen participar en un futuro.

La Figura 2-1 expone la interacción entre dos entidades que soportan parte de su lógica de negocio con un proceso web, enfocando en la parte privada de la interacción para cada una de ellas. En la Figura 2-2 se muestra la misma interacción pero resaltando la parte pública del intercambio de mensajes.

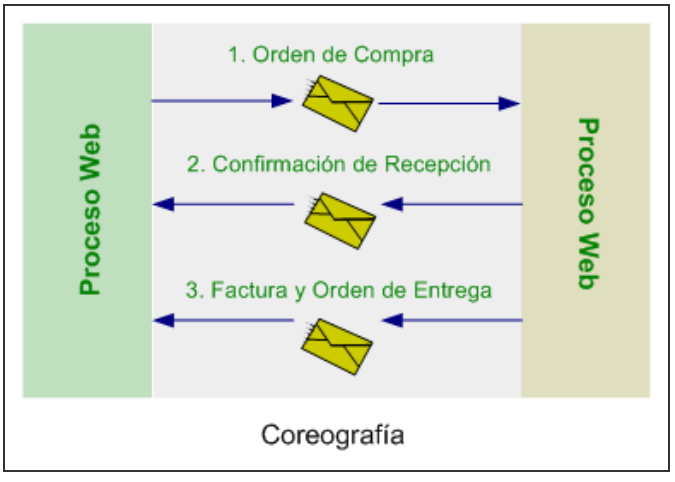

Figura 2-2 Coreografía de Servicios Web

Como puede verse, la interacción entre estas dos entidades sigue un modelo: primero se envía la orden de compra en un sentido, luego se envía una confirmación a la orden y finalmente se envía una factura de compra, estas últimas en el otro sentido. Con base en este modelo y la representación de datos para cada mensaje requerido, cualquier otra entidad compradora podría comunicarse con la entidad proveedora. Dicho de otro modo, el orden en el flujo de la información se constituye en un *protocolo de negocio* que dicta premisas a las entidades que deseen participar en una interacción de negocio. Nótese que para la coreografía no es relevante la forma en la cual cada entidad implementa su participación en la interacción, simplemente exige que se rija por ella.

Junto con la iniciativa que busca convertir a la web en una *web de servicios*, se encuentra la iniciativa que pretende también convertirla en una *web de conceptos*. Uno de los creadores de lo que hoy conocemos como Internet, ha propuesto dotar a la web de *semántica formal* entendible por las máquinas, de tal forma que la información disponible sea *entendida* y *usada* por estas en forma más eficiente.

El punto interesante es la convergencia entre los servicios y la semántica en la web. Una de las propuestas de esta iniciativa conocida como **Web Semántica** es precisamente ampliar la descripción de los servicios web con semántica formal, de tal modo que no sólo esté disponible la información de acceso, sino también una descripción formal y procesable de las capacidades, precondiciones, poscondiciones, entradas y salidas que éste exhibe.

La visión más ambiciosa de la web semántica vislumbra la posibilidad de componer servicios automáticamente, mediante agentes inteligentes que logren localizar y poner a hablar servicios entre sí, mediante inferencia y procesamiento de las capacidades de los mismos. A pesar de los avances logrados, tal escenario todavía no es posible, puesto que no se cuenta con un mecanismo lo suficientemente completo de descripción y de inferencia que permita desarrollar agentes capaces de realizar estas tareas. Dicho de otro modo, **localizar** y **componer** servicios **automáticamente** aún no es posible.

Sí es posible en cambio, realizar **composiciones** partiendo del hecho que el desarrollador conoce de antemano los servicios que va a utilizar **(localización)** y además cuenta con una **descripción** de los mismos.

2.2 LA WEB SEMÁNTICA

2.2.1 La Web Actual y sus Limitaciones

El enfoque de los últimos ocho años en los desarrollos de Internet, ha estado centrado en conectar computadores en redes corporativas y en la puesta de varios tipos de contenido en la red. Estos desarrollos han permitido a los negocios comunicarse de manera eficiente con consumidores, otros negocios y sus propios empleados y a las personas acceder a información variada y a servicios.

El gran problema que evidencia la web actual es la falta de capacidad de las representaciones en las cuales ésta se basa para expresar significados. Cualquier página de contenido disponible en la web presenta únicamente información adicional de formato (HTML) pero no información de descripción que establezca claramente sobre qué aspectos trata. Cualquier servicio web publicado en Internet presenta información completa acerca del modo en el cual un potencial usuario debe interactuar con él, pero presenta poca información acerca de su capacidad.

Como ejemplo, imagínese que existe un sitio web que presenta información meteorológica actualizada de una ciudad en particular. Los datos de temperatura y humedad pueden ser útiles para una gran variedad de propósitos, sin embargo, tal información sólo puede ser consumida por un navegador web. Cualquier programa de computador que quisiera obtener el valor de la temperatura actual debería conocer de antemano el formato HTML empleado en el sitio para luego poder aislarlo y extraer lo requerido. Si se trata de un servicio meteorológico, quien quiera usarlo obtendrá datos de valor, pero primero habrá de encontrarlo, tarea que puede llegar a ser lenta y complicada.

El problema mayúsculo consiste en que no es posible usar la web para encontrar información acerca de algo en particular a partir de la comprensión de su significado (contenido), sus capacidades (servicios) y de sus relaciones con otros conceptos. Los computadores por sí solos no están habilitados para comprender el significado de los documentos de manera inteligente como lo hacen los seres humanos. La web semántica es una solución por medio de la cual los equipos (máquinas) pueden intercambiar información y estar de acuerdo en el significado de los conceptos.

2.2.2 ¿Qué es la Web Semántica?

Básicamente, la web semántica se fundamenta en una gran colección de información etiquetada de los recursos disponibles en Internet, la cual permite representarlos fácilmente. Tal información se conoce como metadatos y es utilizada para describir documentos existentes, páginas web, conceptos, bases de datos, servicios y otros recursos que se encuentran en la web, para que las aplicaciones software tengan una comprensión adecuada de lo que significa su contenido.

Si bien esta idea resulta un poco abstracta, los humanos la aplican en la vida diaria al asociar cosas, palabras, lugares o personas con conceptos. A manera de ejemplo, la palabra “perro” esta almacenada como metadatos en el cerebro humano, de tal forma que si una persona la piensa o la ve, la asocia con un mamífero de cuatro patas en forma inequívoca. En otras palabras, el cerebro genera la información correspondiente que permite describir y asociar un objeto de manera única.

En la Figura 2-3 se puede apreciar la interpretación del concepto “*Perro”* tanto para una persona como para un computador.

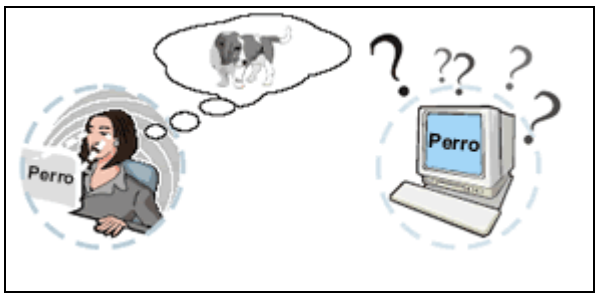

Figura 2-3 Concepto de Perro para una Persona y para un Computador

Los humanos están en la capacidad de comunicarse porque la comprensión de la palabra “perro” es siempre la misma, incluso si se habla en diferentes idiomas y las palabras son diferentes.

2.2.3 Web Actual y Web Semántica

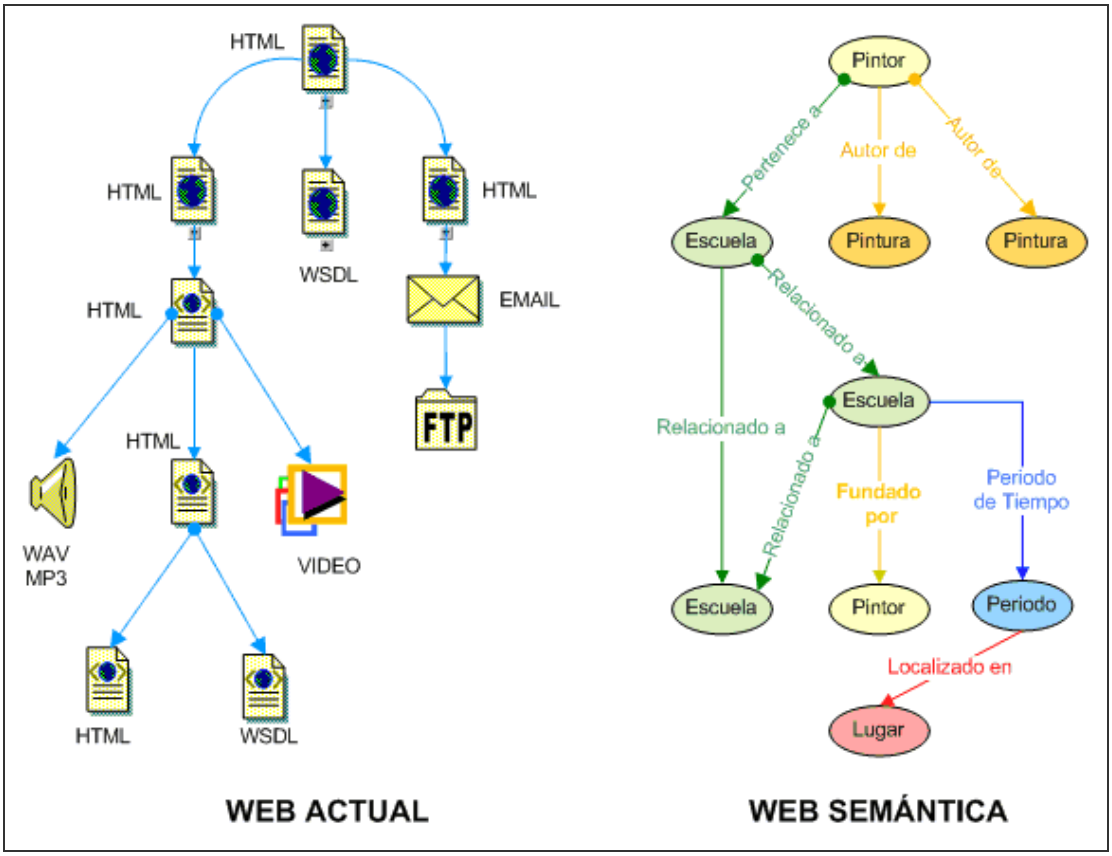

Figura 2-4 Estructura de Web Actual y de Web Semántica

En la figura 2-4 puede verse, en el lado izquierdo, la representación gráfica de la estructura actual de Internet, organizada como páginas web desarrolladas en diferentes lenguajes (HTML, PHP, JSP y ASP) y enlazadas entre sí, permitiendo el acceso a documentos, recursos multimedia y servicios como transferencia de archivos o correo electrónico. La estructura de la web semántica, mostrada en el lado derecho de la figura está basada en conceptos y las relaciones que existen entre ellos.

Actualmente Internet se asemeja a un grafo formado por nodos del mismo tipo, y arcos (vínculos) igualmente indiferenciados. Por ejemplo, no se hace distinción entre el sitio personal de un profesor y el portal de una tienda en línea, como tampoco se distinguen explícitamente los enlaces a las asignaturas que imparte ese profesor de los enlaces a sus publicaciones. En la web semántica cada nodo tiene un tipo (periodo, escuela, pintor, pintura) y los arcos representan relaciones explícitamente diferenciadas (pintor – pintura, pintor – escuela, escuela – periodo). Este tipo de representación también será útil para identificar servicios web disponibles, sus características, entradas y salidas y sus capacidades.

La nueva web es un espacio conceptual de información en el cual cada recurso es representado por medio de un Identificador Universal de Recursos *(URI: Universal Resource Identifier),* y descrito por metadatos que pueden ser procesado por máquinas. Una vez que se asigna el URI a los recursos, estos pueden ser referidos por cualquiera y se habilita la posibilidad de establecer relaciones complejas entre ellos, cuestionarlas y procesarlas.

Frente el crecimiento caótico de recursos y la ausencia de una organización clara de la web actual, la web semántica aboga por clasificar, dotar de estructura y anotar los recursos con semántica explícita procesable por máquinas.

2.2.4 ¿Cómo trabaja la Web Semántica?

Hay dos perspectivas distintas en la web semántica. La primera de ellas es un esfuerzo para complementar el contenido existente con significados semánticos del mismo; la

segunda consiste en crear un conjunto de aplicaciones que harán uso de los nuevos metadatos creados. Ambos componentes son necesarios para hacer realidad el concepto de la nueva web.

El corazón de la web semántica son los vocabularios descentralizados conocidos como *Ontologías*. Estas son el equivalente humano a la colección especial de neuronas que permiten a las personas comprender cuál es el significado de la palabra “perro”. Las ontologías se encargan de definir las restricciones específicas del significado de las palabras y conceptos de un dominio en particular.

En la figura 2-3 se muestra cómo, mediante metadatos, se relaciona el contenido de un documento web con vocabularios que ayudan a los computadores a usarlo en forma más eficiente. Para los programas, la palabra perro ya no sería plana e inconexa sino un concepto previamente definido en una ontología de acuerdo con el contexto en el que se esté trabajando.

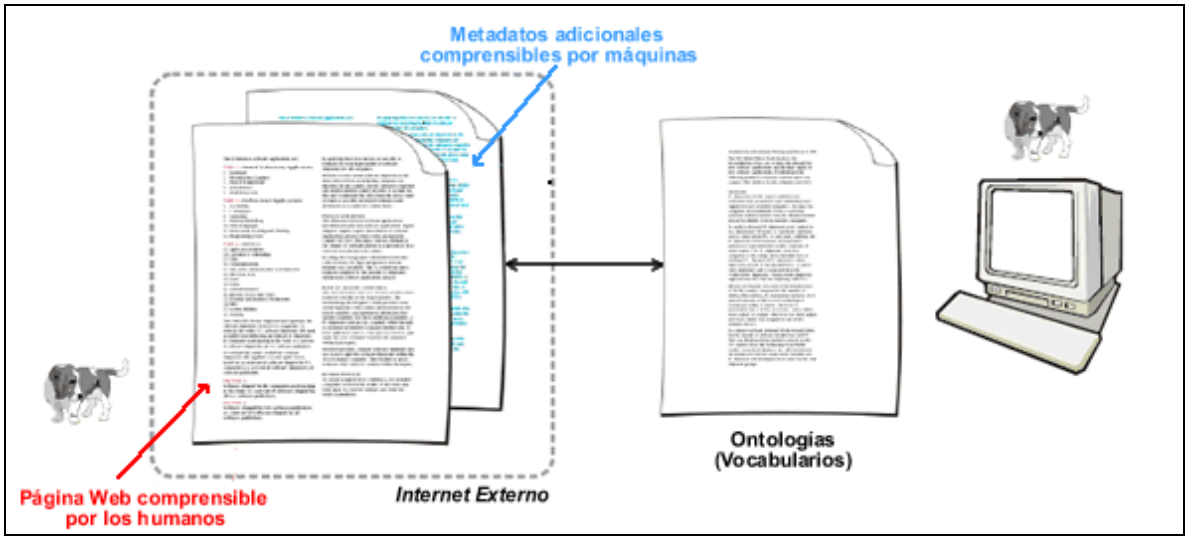

Figura 2-5 Metadatos y Ontologías en la Web Semántica

El nuevo contenido semántico, permitirá a los computadores comunicarse de manera más inteligente, hacer peticiones, responder peticiones y deducir información.

La segunda parte de la web semántica son las aplicaciones que usan el contenido semántico. Los expertos creen que las aplicaciones que usan este tipo de contenido llegarán a ser ubicuas en un futuro cercano.

2.2.5 Ontologías

Una ontología es un conjunto de términos, común y compartido, que describe los conceptos de un dominio, sus propiedades y las relaciones entre dichos conceptos. Las ontologías proporcionan una semántica procesable por las máquinas y entendible por los humanos, necesaria para permitir que las aplicaciones y sistemas empresariales colaboren de una manera inteligente y escalable**.**

Una ontología puede permitir a los programas "razonar semánticamente" sobre determinados conceptos, sin haber sido programados explícitamente para ello, utilizando una serie de primitivas generales en términos de las cuales están definidos los conceptos que forman las ontologías. Pueden también actuar como modelo de referencia común entre sistemas diferentes que utilizan conceptos similares, facilitando la interoperabilidad.

2.2.5.1 Componentes de una Ontología

Las ontologías tienen los siguientes componentes que sirven para representar el conocimiento de algún dominio:

- *Clases o Conceptos:* son las ideas básicas que se intentan formalizar.
- *Instancias:* se utilizan para representar objetos determinados de un concepto.
- *Propiedades:* especifican características o atributos de las clases o instancias.
- *Relaciones:* representan la interacción y enlace entre los conceptos del dominio.

- *Funciones:* son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología.

- *Axiomas:* son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología.

Estos últimos componentes, los axiomas, permiten junto con la herencia de conceptos, inferir conocimiento que no esté indicado explícitamente.

Ventajas de las Ontologías

- Brindan capacidad de procesamiento e interpretación a los computadores, lo que les permite realizar algunas tareas en forma semejante a como las haría un ser humano.

- Permiten almacenar información de un dominio de manera ordenada, sin lugar a ambigüedades puesto que el significado de las cosas que allí se definen es único y conceptualmente claro.

- Disminución del esfuerzo y el tiempo en la búsqueda de información en Internet. La inminente masificación de contenidos semánticos conducirá a la reducción del costo de la adquisición de la información.

- Se constituyen en la base conceptual para la migración de la web actual a la web semántica y la evolución de la Inteligencia Artificial.

2.3 SERVICIOS WEB SEMÁNTICOS

Los servicios web semánticos son una línea importante de la web semántica, que propone describir no sólo la información de acceso a un servicio, sino definir vocabularios de funcionalidad y procedimientos para describir servicios web. Dicha descripción abarca aspectos como: entradas, salidas, procesos, condiciones necesarias para que se puedan ejecutar, los efectos que producen y la información para localizarlos.

Se habla de servicios web semánticos porque se agrega semántica explicita a la descripción de los mismos, por medio de la adición de metadatos, utilizando ontologías. Aunque la actual especificación se servicios web agrega metadatos a su descripción, no son considerados como semánticos puesto que no están relacionados con ontologías.

La adición de semántica a la descripción de los servicios incrementa la eficiencia en la realización de tareas como: *Localización*, *Composición* y *Monitoreo de Ejecución*.

Usando UDDI para localizar servicios, es posible pero difícil efectuar una adecuada coincidencia entre lo que desea encontrar un solicitante de servicio y los diferentes proveedores de servicios. Si bien pueden existir varios servicios relacionados, también es posible que ninguno de ellos satisfaga exactamente los requisitos del solicitante.

La localización de servicios web debe estar basada en la *coincidencia semántica* entre una *descripción declarativa del servicio buscado* y *la descripción del servicio ofrecido*. Esto es, debe buscarse que la coincidencia entre los anuncios y los requisitos no sólo se de en el nivel lexicográfico (con base en palabras claves), sino también en el nivel de las *capacidades* de los servicios. Con el advenimiento de la web semántica, la localización de servicios podrá lograrse en forma automática por medio de agentes que hagan inferencias basándose en la capacidades de los servicios descritos; en la actualidad, tales procesos se ejecutan en forma manual.

Si se efectúa la búsqueda de servicios comprendiendo la semántica de las descripciones y de las necesidades, es posible reconocer el grado de desigualdad entre ellas. De esta manera se incrementa la posibilidad de encontrar un servicio que se ajuste aproximadamente a los requisitos del solicitante.

La composición de servicios web es una iniciativa reciente que se está abriendo paso rápidamente. Como las alternativas emergentes para lograr composición no son estándares de industria, es crucial buscar puntos de coincidencia entre la composición pura de servicios y las descripciones semánticas de los mismos. Si un usuario de servicios los busca con base en coincidencias semánticas entre las capacidades ofrecidas y los requisitos por resolver, igualmente podrá usar información adicional sobre

precondiciones, poscondiciones y modelos de proceso para hacer trabajar juntos a tales servicios en la resolución de requisitos de mayor nivel.

Finalmente, en razón a la dinámica que rige el mundo en general, la composición de servicios debe ser lo suficientemente flexible y robusta para adaptarse fácilmente a los cambios y a la complejidad en los requisitos a ser resueltos por medio de servicios web.

2.4 TÉCNOLOGÍAS PARA LA WEB SEMÁNTICA

Las tecnologías que empiezan a hacer posible la web semántica se pueden dividir en dos grandes grupos. El primero de ellos agrupa herramientas que establecen mecanismos para la representación, almacenamiento, acceso, actualización, visualización y conversión de ontologías. El segundo recopila tecnologías relacionadas con la localización, descripción y composición de servicios web semánticos.

Las tecnologías pertenecientes al primer grupo permiten agregar contenido semántico a los diferentes recursos web. Para este fin, básicamente se cuenta con dos lenguajes: Lenguaje para Ontologías Web (*OWL*: *Web Ontology Language)* e Infraestructura para la Descripción de Recursos *(RDF: Resource Description Framework)*. Por medio de estos lenguajes los desarrolladores pueden enlazar los elementos de sus sitios web con documentos y bases de datos de vocabularios descentralizados (ontologías) que definen su significado.

En la actualidad existen una gran cantidad de tecnologías y herramientas relacionadas con la web semántica y específicamente con ontologías, sin embargo, las de mayor peso y aceptación a nivel comercial e investigativo son las mencionadas con anterioridad, puesto que son recomendaciones del W3C.

2.5 TÉCNOLOGÍAS PARA COMPONER SERVICIOS WEB

Existen diferentes alternativas tecnológicas para efectuar composición de servicios web. Éstas pueden clasificarse de acuerdo con el tipo de servicios web que permitan

componer, esto es, servicios web semánticos y no semánticos. A continuación se muestra un corto resumen de estas alternativas y su explicación detallada, figura en los siguientes tres capítulos.

2.5.1 Tecnologías para componer servicios web no semánticos

A partir de los estándares actuales que definen los servicios web es posible realizar composición se servicios mediante el Lenguaje para la Ejecución de Procesos de Negocio para Servicios Web *(BPEL4WS: Bussines Process Execution for Web Services).*

Aunque no incluyen una relación estricta con ontologías y por tanto la composición no puede ser considerada como un servicio web semántico, es una alternativa de gran valor para componer servicios web. BPEL4WS no permite componer servicios web semánticos.

2.5.2 Tecnologías para componer servicios web semánticos

Como se mencionó antes, éstas pertenecen al segundo grupo de tecnologías de la web semántica, es decir, están estrechamente relacionadas con ontologías y se basan en RDF y OWL.

Las propuestas más representativas en este campo, sin convertirse ninguna de ellas en un estándar todavía, son: Lenguaje para Ontologías de Servicios Web *(OWL-S Ontology Web Language - Services)* y la Ontología para Modelar Servicios Web *(WSMO: Web Services Modeling Ontology).*

3. LENGUAJE DE EJECUCIÓN DE PROCESOS DE NEGOCIO PARA SERVICIOS WEB

3.1 INTRODUCCIÓN

Como se mencionó en el Capítulo 1, los servicios web proporcionan una forma estandarizada para facilitar la interacción entre aplicaciones que se ejecutan en plataformas heterogéneas y en entornos distribuidos. Los servicios web usan un modelo poco acoplado de integración, el cual provee una gran flexibilidad en la creación de sistemas distribuidos basados en componentes heterogéneos.

No obstante lo anterior, el modelo de intercambio de mensajes que soporta WSDL en forma directa, sólo permite lograr interacciones sincrónicas o asincrónicas en las que no se soporta el almacenamiento del estado de la interacción.

Para lograr una integración de aplicaciones más flexible y robusta, se requiere la capacidad para definir y ejecutar: interacciones sincrónicas y asincrónicas con persistencia de estado entre múltiples servicios participantes; acciones que permitan resolver condiciones de excepción presentadas durante este tipo de interacciones y acciones que sirvan de compensación a situaciones donde ocurren fallas que interrumpen o impiden la ejecución satisfactoria de una interacción. [BPEL03].

3.2 CONCEPTOS BÁSICOS

BPEL4WS es un lenguaje basado en XML para describir procesos de negocio basados en servicios web. Un proceso de negocio descrito con este lenguaje, usa o importa funcionalidad de servicios web y publica o exporta funcionalidad como si fuera un servicio web. Los procesos de negocio pueden ser descritos en dos formas diferentes, cada una de las cuales tiene su objetivo particular. Estas formas son: procesos de negocio ejecutables, los cuales se corresponden con procesos de orquestación de servicios;

procesos de negocio abstractos o protocolos de negocio, los cuales se corresponden con procesos de coreografía de servicios. [BPEL03].

BPEL4WS no incluye mecanismos de inclusión de semántica explícita, sin embargo, más adelante se menciona en qué modo este lenguaje podría llegar a ser combinado con otros lenguajes que sí lo hacen. Por otro lado, de las posibles representaciones de procesos, este trabajo de investigación se centra en aquellos que son ejecutables, por lo que el resto del capítulo sólo trata este tipo de procesos y su representación con este lenguaje.

Un proceso de negocio ejecutable descrito con BPEL4WS es en esencia un servicio compuesto por otros. Para materializar tales descripciones, el lenguaje proporciona un conjunto de elementos de sintaxis para soportar:

- **Definición de socios**, a través de construcciones de lenguaje que permiten declarar qué socios participarán en la definición del proceso.
- **Interacciones con socios**, a través de actividades para la invocación a servicios provistos por socios, recepción de invocaciones y envío de respuestas.
- **Manejo de datos**, a través de actividades para almacenamiento y acceso a variables que representan mensajes WSDL, manipulación de variables de control y manejo de expresiones.
- **Implementación de lógica de negocio**, a través de actividades para efectuar secuencias, repeticiones condicionales, selecciones condicionales y concurrencia de otras actividades.
- **Correlación de mensajes**, a través de construcciones de lenguaje que habilitan la posibilidad de que exista más de una instancia de proceso ejecutándose simultáneamente.
- **Manejo de fallas**, a través de construcciones de lenguaje que permiten definir condiciones de excepción y su manipulación.
- **Compensación**, a través de construcciones de lenguaje que permiten definir actividades de compensación que deshagan total o parcialmente acciones de un proceso en ejecución.

BPEL4WS se basa en otras especificaciones XML, específicamente: **XML Schema 1.0** para la definición de tipos de datos, **WSDL 1.1** para la definición de tipos de datos (mensajes) y para importar y exportar funcionalidad de interfaces de servicios, y **XPath 1.0**[1] para dar soporte a la manipulación de datos. El lenguaje cuenta también con mecanismos de extensibilidad para acomodarse a futuras versiones de estas especificaciones [BPEL03].

3.3 DEFINICIÓN DE PROCESOS EJECUTABLES CON BPEL4WS

A continuación se profundiza acerca de los detalles técnicos del lenguaje en cuanto a las estructuras XML que permiten la descripción de procesos de negocio ejecutables. Dado que un documento BPEL4WS es un documento XML, este debe cumplir con sus reglas en cuanto el uso de espacios de nombres y al seguimiento a un esquema definido[2]. La siguiente tabla muestra convenciones usadas en los ejemplos XML.

CONVENCIONES XML PARA BPEL4WS	
ncname	Nombre No Calificado, se aplica en atributos
qname	Nombre Calificado, se aplica en atributos
?	Puede estar presente o ausente, se aplica en atributos y en elementos
+	Aparece 0 o más veces, se aplica en atributos y en elementos
*	Aparece 1 o más veces, se aplica en atributos y en elementos
A|B	A o B, mutuamente excluyente. Se aplica en atributos y valores de atributos

Tabla 3-1 Convenciones XML para BPEL4WS

3.3.1 Proceso

El papel de BPEL4WS es el de definir un nuevo servicio web a través de la composición de un conjunto de servicios web existentes. La interfaz del servicio compuesto es descrita como una colección de interfaces WSDL. El servicio compuesto se denomina **proceso**.

---

[1] XPath es un lenguaje para acceder a partes de documentos XML.
URL: http://www.w3.org/TR/1999/REC-xpath-19991116

[2] Mayor información acerca del uso de XML y sus espacios de nombres en el documento W3C “Espacios de Nombres en XML 1.1”. URL: http://www.w3.org/TR/2004/REC-xml-names11-20040204/

Un proceso BPEL4WS está definido por un documento WSDL que relaciona el conjunto de interfaces y de operaciones que el proceso ofrece como un servicio web tradicional y un docuemtno BPEL4WS en el cual se consigna la definición del proceso de negocio ejecutable.

3.3.2 Definición de Socios

Un requisito importante para declarar procesos es modelar la relación entre éste y sus socios. Con WSDL ya se tiene una descripción funcional del servicio provisto por un socio, tanto a nivel abstracto como a nivel concreto. Un **socio** de un proceso es un servicio web que usa su funcionalidad o que le provee funcionalidad. En consecuencia, un proceso y un socio mantienen una relación igual-a-igual y su interacción ocurre en el nivel de los mensajes que ellos intercambian.

3.3.2.1 Tipos de Enlace de Socio

Un tipo de enlace de socio caracteriza la relación conversacional entre dos servicios, definiendo el rol que juega cada uno de ellos juega dentro de la conversación y las interfaces que cada uno ofrece para recibir mensajes dentro del contexto de esa conversación. Cada rol debe contener únicamente una interfaz WSDL.

```
<partnerLinkType name="ncname" xmlns ="http://schemas.xmlsoap.org/ws/2003/05/partner-link/">
      <role name="ncname">
            <portType name="qname"/>
      </role>
      <role name="ncname">?
            <portType name="qname"/>
      </role>
</partnerLinkType>
```

3.3.2.2 Enlace de Socio

Cada uno de los servicios que efectivamente interactúan con el proceso se modela como un **enlace de socio**. Cada enlace de socio proviene de un tipo de enlace de socio, a partir del cual declara el rol que va a jugar tanto el proceso como el servicio en la interacción.

Cada enlace de socio tiene un nombre único el cual se usa en todas las interacciones del proceso con ese socio en particular. La definición de los enlaces de socio se ubica en el archivo de descripción de proceso. La definición de este elemento tiene la siguiente forma.

```
<partnerLinks>
    <partnerLink name="ncname" partnerLinkType="qname" myRole="ncname"?
        partnerRole="ncname"?>+
    </partnerLink>
</partnerLinks>
```

El atributo myRole identifica el rol del proceso mientras que el atributo partnerRole identifica el rol del socio.

3.3.2.3 Puntos finales de Acceso

Representan un concepto similar al definido por el elemento port de WSDL. Cada rol de cada enlace de socio recibe un **punto final de acceso** bien sea en el momento de publicar el proceso o en forma dinámica a través de una asignación directa.

3.3.3 Actividades Básicas

Un componente primario de un proceso de negocio es la interacción con sus socios. Existe un conjunto de actividades básicas que definen este tipo de interacciones, las cuales son detalladas a continuación.

3.3.3.1 Variables

En un proceso BPEL4WS, los datos que se envían o reciben se almacenan en **variables**. Una variable aloja una instancia de un mensaje WSDL o una parte de este.

```
<variables>
    <variable name="ncname" messageType="qname"? type="qname"? element="qname"? />+
</variables>
```

3.3.3.2 Invocación de Operaciones

Permite invocar una operación de una interfaz WSDL provista por un socio. En la sintaxis básica debe especificarse un nombre para la actividad y la información del socio (enlace de socio, interfaz y operación). También se declaran las variables a usar para almacenar los mensajes.

```
<invoke name="ncname" partnerLink="ncname" portType="qname" operation="ncname"
        inputVariable="ncname"? outputVariable="ncname"? />
```

3.3.3.3 Provisión de Operaciones

Un proceso de negocio ofrece funcionalidad a través de las actividades marcadas con receive y reply. Una actividad receive especifica una interfaz y una operación sobre la cual el proceso espera recibir una invocación desde el socio perteneciente al tipo de enlace de socio especificado.

```
<receive name="ncname" partnerLink="ncname" portType="qname" operation="ncname"
        variable="ncname"? />
```

Para responder una solicitud se usa la actividad reply. Esto sólo tiene sentido cuando la interacción iniciada desde el socio externo es de tipo sincrónica. Para relacionar la actividad que recibe la invocación con la actividad que envía la respuesta existe una restricción, la cual dice que de las posibles interacciones sincrónicas definidas, sólo una puede estar activa en un instante dado. Finalmente, la presencia de un atributo faultName identifica una falta WSDL que está definida en el espacio de nombres del proceso y que puede ocurrir como resultado de la invocación recibida con receive. En tal caso, la variable definida en la actividad reply alojará el mensaje de falta.

```
<reply name="ncname" partnerLink="ncname" portType="qname" operation="ncname" variable="ncname"?
        faultName="qname"? />
```

3.3.3.4 Notificación de Fallas

Si en un proceso se necesita definir una falla interna en forma explícita se utiliza la actividad throw. Esta debe tener un nombre calificado único y puede estar en cualquiera de los espacios de nombres usados en el proceso. Por otro lado, la actividad puede contener una variable que alojará información adicional sobre la falla. Un **manejador de fallas** puede usar los datos que se alojen en tal variable para ejecutar alguna acción extra.

```
<throw faultName="qname" faultVariable="ncname"? />
```

3.3.3.5 Esperar

La actividad marcada con wait permite que la ejecución un proceso espere a que pase un tiempo determinado antes de continuar. El manejo de estas condiciones temporales se hace por medio de expresiones XPath. En el ejemplo, **duración** declara una medida de tiempo durante la cual es proceso espera, mientras que **límite** declara una condición temporal (usualmente una marca de tiempo o *timestamp*).

```
<wait name="ncname" (for="duración" | until="límite") />
```

3.3.3.6 Actividad Vacía

La actividad marcada con empty permite definir una acción vacía. Esta sirve como acción por defecto en algunas de las actividades de estructura mostradas más adelante.

```
<empty name="ncname" />
```

3.3.4 Lógica del Proceso

Por medio de las actividades de estructura se describe la lógica de negocio del proceso. La combinación de las actividades de estructura y de las actividades básicas permite una gran flexibilidad en la implementación de algoritmos simples o complejos.

3.3.4.1 Secuencia

Una actividad de secuencia es marcada con sequence y aloja una o más actividades (marcada como actividad) que deben ser ejecutadas en estricto orden. La actividad de secuencia termina cuando termine la última de las actividades que encierra.

```
<sequence name="ncname">
      actividad+
</sequence>
```

3.3.4.2 Bifurcación

La actividad de bifurcación se marca con switch y soporta el comportamiento condicional en el cual, una y sólo una de varias opciones disponibles debe ser ejecutada. Cada opción es marcada con case y aloja una actividad básica. La ejecución está condicionada a una expresión lógica. La opción especial marcada con otherwise aloja una actividad que se ejecutará sólo cuando ninguna de las condiciones en las ramas case fue verdadera.

```
<switch name="ncname">
      <case condition="expresión-lógica">+
            actividad+
      </case>
      <otherwise>?
            actividad+
      </otherwise>
</switch>
```

3.3.4.3 Ciclo

La actividad marcada con while ejecuta en forma iterativa la actividad básica que encierra hasta que su condición lógica se vuelva negativa.

```
<while name="ncname" condition="condición-lógica" >
      actividad+
</while>
```

3.3.4.4 Escogencia

La actividad de escogencia pick define una serie de **eventos** que pueden ocurrir y una actividad básica asociado a cada uno de ellos. Cuando la ejecución del proceso llega a una actividad de escogencia, queda a la espera que uno de los eventos ocurra. Cuando ocurra uno de los eventos, se ejecuta la actividad que este encierra.

El evento puede ser la recepción de un mensaje hacia una operación publicada por el proceso o una alarma basada en un temporizador.

```
<pick createInstance="yes|no"? >
      <onMessage partnerLink="ncname" portType="qname" operation="ncname" variable="ncname"?>+
            actividad
      </onMessage>
      <onAlarm (for="duración" | until="límite")>*
            actividad
      </onAlarm>
</pick>
```

3.3.4.5 Concurrencia

La ejecución de actividades en paralelo se logra por medio de la actividad marcada con flow.

```
<flow name="ncname">
      actividad+
</flow>
```

3.3.4.6 Sincronismo

Si se requiere que dos actividades encerradas por una actividad flow se ejecuten una después de la otra, se dice que existe una **dependencia de sincronización** entre ellas. Para soportar esto, la actividad flow debe ser complementada con los elementos enlace, fuente y destino (link, source y target). Un **enlace** relaciona dos actividades, una **fuente** y una **destino**, expresando una relación de sincronización entre ellas. Adicionalmente, la

transición entre una actividad fuente y una actividad destino puede estar restringida por una condición lógica definida como **condición de transición**.

```
<flow name="ncname">
      <links>?
            <link name="ncname">+
      </links>
      actividad+
</flow>
```

En el siguiente ejemplo, la actividad receive es fuente de un enlace A y destino de un enlace B.

```
<flow name="ncname">
      <links>
                  <link name="enlaceA" />
      <link name="enlaceB" />
      </links>
...
      <receive name="ncname" partnerLink="ncname" portType="qname" operation="ncname"
      variable="ncname"?>
            <source linkName="enlaceA" />
            <target linkName="enlaceB" />
      </receive>
...
</flow>
```

El significado de esta definición es el siguiente:

- La actividad receive es el destino del enlace B, lo cual significa que existe una actividad que es la fuente de este enlace. La actividad receive sólo puede ejecutarse cuando su *actividad fuente* haya finalizado su ejecución.

- Por otro lado, la actividad receive es fuente del enlace A. Esto quiere decir que existe una actividad que es el destino de este enlace y que sólo puede ejecutarse cuando esta actividad receive se haya ejecutado.

Cuando se define que una actividad es fuente de un enlace, el elemento source puede llevar un atributo de **condición de transición** (transitionCoindition), cuyo valor es una

condición lógica. Si la evaluación de la condición lógica es negativa, el enlace entre la actividad fuente y la actividad se vuelve negativo, es decir, la actividad destino **no se ejecuta**.

Puesto que una actividad puede ser el destino de uno o más enlaces (enlaces de entrada), debe existir un mecanismo para decidir si la actividad se ejecuta o no dependiendo del valor de dichos enlaces de entrada. Este mecanismo consiste en evaluar una **condición de unión** (joinCondition). La condición de unión es un atributo de las actividades que son destino de por lo menos un enlace. Una condición de unión explícita es una operación lógica definida por el usuario. Una condición de unión implícita es proporcionada por la implementación del ambiente de ejecución y es siempre una O exclusiva, con la cual, una actividad se ejecutará cuando al menos uno de los enlaces de entrada tiene un valor positivo.

3.3.5 CICLO DE VIDA DE UN PROCESO Y CORRELACIÓN

3.3.5.1 Ciclo de Vida de un Proceso

BPEL4WS da soporte a interacciones entre procesos manteniendo el estado de las mismas. Para ello, cada interacción entre un proceso y sus socios es ejecutada por una **instancia** independiente del proceso. Cada instancia de proceso se ejecuta en forma independiente de las demás y maneja sus propios datos.

Las instancias en BPEL4WS se crean implícitamente cuando un proceso recibe un mensaje en uno de sus puntos de inicio. Las actividades receive y pick pueden ser configuradas para crear una nueva instancia de proceso. Específicamente, la actividad receive que puede iniciar un proceso debe ser la que inicia lógicamente el comportamiento del proceso, es decir no debe tener actividades antecedentes. La actividad pick puede iniciar un proceso a través de un evento onMessage.

Una instancia de proceso puede finalizar en forma normal o anormal, lo cual indica la compleción o no del objetivo del proceso. Una vez se ha creado, una instancia de proceso se finaliza en alguno de los siguientes casos:

- La actividad que define el comportamiento del proceso (la más externa) se finaliza. En este caso el proceso finaliza en forma normal
- Cuando en el contexto del proceso aparece un mensaje de falla WSDL, independientemente de si hay un manejador para tal falla. La terminación del proceso es anormal
- Cuando se alcanza una actividad terminate. La finalización del proceso es también anormal en este caso.

3.3.5.2 Correlación

El soporte a la persistencia en el estado de las interacciones entre dos o más procesos no sólo requiere del mecanismo de múltiples instancias. El mecanismo conocido como **correlación de mensajes** permite garantizar la entrega correcta de mensajes desde y hacia distintas instancias de un mismo proceso. Este mecanismo se vale de los datos de aplicación de una interacción y es indispensable cuando el transporte se realiza por medio de protocolos livianos como SOAP. Es posible que un ambiente de ejecución con mecanismos de transporte sofisticados no haga uso de esta característica del lenguaje y maneje la correlación entre los mensajes y las instancias de proceso en un nivel inferior al de los datos de aplicación.

BPEL4WS maneja los escenarios de correlación proporcionando un método para declarar que un proceso tiene **grupos de operaciones** que deben estar mutuamente relacionadas (correlacionadas) en cada una de sus instancias. En una interacción entre varios socios de negocio, cada uno de ellos debe ajustar sus procesos declarando el grupo de operaciones con correlación.

De los mensajes definidos en cada interfaz de proceso se extraen aquellas partes que son de valor para la persistencia del estado de una interacción. Tales partes son definidas

como propiedades para luego ser unidas en **conjuntos de correlación**. Los datos que forman los conjuntos de correlación son los que van a simular el comportamiento de un identificador de instancia de proceso, es decir, por medio de esos datos se va a lograr el enrutamiento adecuado de los mensajes.

3.3.6 CONTEXTOS, MANEJO DE FALLAS Y COMPENSACIÓN

Los procesos ejecutables usualmente representan aplicaciones de negocio o de industria que están habilitadas para interactuar con otras a través de Internet. Por tal razón, es normal que un proceso de negocio maneje datos sensibles e interactúe con bases de datos y con aplicaciones internas.

Cuando se ejecuta un proceso de negocio definido con BPEL4WS, pueden ocurrir errores durante la interacción con otros procesos o dentro del proceso mismo. Por otro lado, puede ser necesario deshacer o compensar los efectos de una actividad que ya ha sido completada si acaso hace parte de una larga transacción que deba ser abortada, por ejemplo, por una petición explícita de alguno de los socios del proceso o porque la ejecución degeneró en un error irrecuperable. En razón a lo anterior, BPEL4WS soporta la definición de actividades para el manejo de fallas y la definición de actividades de compensación. En esta sección se explica la forma en la cual es lenguaje da soporte a tales características.

3.3.6.1 Contextos

En la definición más simple, un **contexto** es un grupo arbitrario de actividades que representa el comportamiento de un proceso de negocio o de una porción de éste. El proceso en sí mismo define un contexto de comportamiento global para las actividades que encierra, no obstante, BPEL4WS otorga la posibilidad de definir otros contextos de comportamiento para porciones del proceso mismo.

Sobre un contexto pueden definirse las siguientes cosas:

- Uno o más manejadores de fallas, que se encargan de manejar fallas externas o internas de la ejecución del proceso.
- Un manejador de compensación, que se encarga de efectuar actividades de compensación.
- Uno o más manejadores de eventos.
- Variables para la manipulación de datos.
- Conjuntos de Correlación, para soportar la persistencia de estado en las interacciones entre varios socios de un proceso de negocio.

La definición de un contexto se hace por medio del elemento scope.

```
<scope variableAccessSerializable="yes|no">
      <variables />?
      <correlationSets />?
      <faultHandlers />?
      <compensationHandler />?
      <eventHandlers />?
      actividad+
</scope>
```

3.3.6.2 Manejo de Fallas

Manejar una falla implica efectuar una o más actividades como respuesta a la ocurrencia de una falla. En BPEL4WS las fallas pueden ser producidas fuera del proceso o dentro del mismo. Las fallas externas a un proceso son aquellas resultantes de la invocación de una operación en uno de los socios. Esto implica que la falla está definida dentro de la operación invocada como una de sus posibles respuestas.

Las fallas internas pueden ocurrir a nivel local durante la ejecución del proceso de negocio. Estas fallas nativas son diversas y de ellas pueden destacarse, el intento de uso de una variable que no existe, el intento de ejecutar una actividad reply que no tiene una actividad receive correspondiente, la evaluación negativa de una condición de unión. El listado completo de fallas nativas puede consultarse en la especificación del lenguaje.
Finalmente, es posible **lanzar** una falla programática o intencional.

```
<faultHandlers>?
      <catch faultName="qname"? faultVariable="ncname"?>*
            actividad
      </catch>
      <catchAll>?
            actividad
      </catchAll>
</faultHandlers>
```

Cada falla posible debe estar definida con un nombre calificado en el atributo faultName. El atributo faultVariable es la variable que permite almacenar información relativa al fallo. En el caso de una falla definida con WSDL, esta variable almacenará el mensaje de error. La presencia de tal variable no es obligatoria.

El elemento faultHandlers permite definir todos los manejadores de faltas que pertenecen a un contexto. Cada manejador se marca con el elemento catch y encierra una actividad. Lo común es usar una actividad reply para notificar a alguno de los socios la ocurrencia del error; sin embargo, el manejo dado a la falla es arbitrario.

Los manejadores de fallos pueden marcarse con la actividad catchAll. Si se hace de esta manera, se está indicando que la actividad que encierra es la que debe ejecutarse por defecto cuando ocurre una falla cualquiera.

3.3.6.3 Compensación

Se entiende compensación como la ejecución de actividades tendientes a deshacer el efecto de un grupo previo de actividades enmarcadas en un contexto que ya finalizó su ejecución. Sólo puede definirse un manejador de compensación para cada contexto de la definición del proceso.

La sintaxis para definir un manejador de compensación dentro de un contexto el la siguiente.

```
<compensationHandler>?
      actividad
</compensationHandler>
```

La invocación del manejador de compensación se declara de forma explícita. Para hacerlo debe usarse un elemento compensate dentro de un manejador de compensación o manejador de fallas que esté en un contexto inmediatamente superior al contexto que quiere ser compensado.

```
<compensate scope="ncname"? />
```

Finalmente, en el Anexo C – Ejemplo de Composición de Servicios web con BPEL4WS, se puede encontrar información adicional de cómo describir y ejecutar un servicio compuesto con BPEL4WS.

4. LENGUAJE PARA ONTOLOGÍAS DE SERVICIOS WEB

4.1 FUNDAMENTOS DEL LENGUAJE PARA ONTOLOGIAS WEB

El Lenguaje para Ontologías Web *(OWL: Ontology Web Language)*, permite definir e instanciar ontologías específicas de Internet. OWL esta diseñado para ser usado por aplicaciones que necesiten procesar el contenido de la información en lugar de sólo presentar dicha información en un formato comprensible por los seres humanos.

OWL se estructura a partir de los conceptos básicos de RDF[3], por lo tanto es posible hacer uso de los diferentes elementos definidos en RDF para maximizar las definiciones hechas en las ontologías OWL.

4.1.1 Cabeceras de la Ontología

OWL permite definir cabeceras para las ontologías, donde se puede incluir información importante para otros usuarios, por ejemplo: comentarios, control de versiones e inclusión de otras ontologías. La definición de la cabecera debe estar situada al inicio del documento OWL y representada mediante el elemento owl:Ontology.

Es posible definir algunos atributos dentro de la cabecera de una ontología, ellos son:

ATRIBUTOS	DESCRIPCIÓN
rdf:about	Provee un nombre o referencia para la ontología.
rdfs:comment	Permite realizar comentarios o anotaciones acerca de la ontología.
owl:priorVersion	Etiqueta estándar para especificar la versión anterior de la ontología.
owl:imports	Proporciona un mecanismo para incluir otras ontologías.
owl:AnnotationProperty	Permite representar propiedades que son utilizadas como anotaciones.
rdfs:label	Es una forma alternativa para expresar en lenguaje común información útil.

Tabla 4-1 Atributos Cabecera OWL

[3] Más información acerca de RDF en http://www.w3.org/RDF/

Para la demostración del uso de estos atributos y de otros elementos OWL se hará uso de un ejemplo de ontología acerca de vinos. Los ejemplos se enmarcan en bloques como el mostrado a continuación. Este en particular muestra la cabecera de la ontología.

```
<rdf:RDF>
<owl:Ontology rdf:about="http://www.w3.org/owl/vino">
<rdfs:comment>Ejemplo de una Ontología OWL</rdfs:comment>
  <owl:priorVersion rdf:resource="http://www.w3.org/owl/vino" />
  <owl:imports rdf:resource="http://www.w3.org/owl/comida" />
  <rdfs:label>Ontología Vino</rdfs:label>
 ...
</owl:Ontology>
</rdf:RDF>
```

4.1.2 Espacio de Nombres

Para poder utilizar conjuntos predefinidos de términos, clases, propiedades y atributos al implementar ontologías, es necesario precisar los vocabularios que definen dichas características al principio de los documentos por medio de referencias de espacios de nombres.

Después de la cabecera de ontología se sigue con la declaración de espacios de nombres. Puede haber cero, uno o muchos espacios de nombres definidos dentro de un documento. De existir, estos deben estar contenidos entre el elemento rdf:RDF.

```
<rdf:RDF>
      xmlns      ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/vino#"
      xml:base   ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/vino#"
      xmlns:owl  =http://www.w3.org/2002/07/owl#"
      xmlns:rdf  ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:xsd  ="http://www.w3.org/2001/XMLSchema#">
</rdf:RDF>
```

En el recuadro, la primera declaración identifica el espacio de nombres por defecto y por tanto no requiere prefijo. La segunda declara que el URI de la ontología es la base para los elementos definidos en este documento OWL. La tercera, indica que en este documento los elementos con el prefijo owl: deben ser referenciados y comprendidos de

acuerdo con las definiciones realizadas en http://www.w3.org/2002/07/owl#. De manera similar, los prefijos rdf:, rdfs:, xsd: deben ser asociados con el respectivo URI, para su correcto uso e interpretación.

De forma análoga los lenguajes de programación, se importan clases para poder utilizar instancias de ellas. En la definición de ontologías se utilizan espacios de nombres para poder utilizar elementos definidos en otras ontologías o lenguajes [OWL04].

4.1.3 Elementos Básicos

Permiten definir clases, propiedades, instancias de clases y relaciones entre instancias. A continuación se presentan dichos elementos.

4.1.3.1 Clases Simples

La mayor parte de los conceptos en un dominio pueden representarse por medio de clases y las posibles relaciones que existen entre ellas.

1. owl:Class

Las clases se definen mediante la etiqueta owl:Class. El nombre de las clases se especifica por medio de la sintaxis rdf:ID=" ".

```
<owl:Class rdf:ID="BodegaVino" />
<owl:Class rdf:ID="Region" />
<owl:Class rdf:ID="CosaConsumible" />
```

Para utilizar o hacer referencia a una clase dentro del mismo documento donde se la definió, debe utilizarse la sintaxis rdf:resource="#Region". Sin embargo, las clases pueden ser referenciadas por otras ontologías diferentes que no incluyan la definición de la clase en cuestión; una manera de hacerlo es especificando por medio de un URI la ruta completa de la ontología que contiene la clase, así: rdf:resource="http://www.w3.org/TR/2004/REC-owl-guide/vino#Region".

Una forma alternativa para poder utilizar clases que se encuentre definidas en otras ontologías, tal vez la más adecuada, es por medio de espacios de nombres. Suponiendo la definición de este último como se muestra a continuación, se puede hacer referencia a una clase **Region** definida en la ontología **vino** usando la etiqueta XML <vin:Region> o el atributo vin:Region="Region". En esta forma, **vin** es el prefijo asignado al espacio de nombres que hace referencia a la ontología **vino**.

```
<rdf:RDF  xmlns:vin=http://www.w3.org/TR/2004/REC-owl-guide/vino#>
```

Otra forma de referencia usa la sintaxis rdf:about="#Region" para ampliar o extender las definiciones de un recurso. Es decir, si se quiere utilizar una clase definida en otra ontología y además es necesario agregar algunas cosas o incluso redefinir otras (extenderla) sin modificar el documento original, debe hacerse uso de la sintaxis rdf:about="&vin;#Region", de esta manera, se puede extender la definición de la clase **Region** en la nueva ontología que se este creando, sin modificar la ontología **vino** pero haciendo uso de algunas de las definiciones realizadas en esta.

2. rdfs:subClassOf

El principal constructor taxonómico que sirve para organizar jerárquicamente las clases es rdfs:subClassOf. Este relaciona una clase más específica con una clase más general. Si X es una subclase de Y, entonces toda instancia de X es también una instancia de Y. La relación rdfs:subClassOf es transitiva, es decir, si Z es una subclase de X y además X es una subclase de Y entonces Z es una subclase de Y.

En este punto es posible crear una definición simple e incompleta de la clase **Vino**, pues sólo se establece el nombre de la clase y que es un subconjunto de la clase **LiquidoPotable.**

```
<owl:Class rdf:ID="Vino">
  <rdfs:subClassOf rdf:resource="&comida;LiquidoPotable" />
  ...
</owl:Class>
```

4.1.3.2 Individuos

Se entiende por *individuos* a los diferentes miembros de una clase. De manera análoga a los entornos de programación, los *individuos* pueden asociarse con las instancias de una clase [OWL04].

Suponiendo que se ha definido previamente la clase **Region**, mediante la propiedad rdf:type es posible asociar un individuo llamado **RegionCostaCentral** como miembro de la clase **Region.** Primero se define el *individuo* con sus respectivas propiedades y luego se asocia a una *clase* en particular.

```
<owl:Thing rdf:ID="#RegionCostaCentral">
...
</owl:Thing>

<owl:Thing rdf:about="#RegionCostaCentral">
  <rdf:type rdf:resource="#Region"/>
</owl:Thing>
```

4.1.3.3 Propiedades Simples

Las propiedades nos permiten representar características propias de los miembros de una clase. Se pueden distinguir dos tipos de propiedades.

- *Propiedades de Tipos de Datos:* Relaciones entre instancias de clases y tipos de datos RDF o esquema XML.

- *Propiedades de Objetos:* Relaciones entre instancias de clases.

Una definición de una clase tiene dos partes: un nombre o referencia y un conjunto de restricciones. Así, las instancias de las clases pertenecen a la intersección de las restricciones o propiedades.

PROPIEDADES	DESCRIPCIÓN
owl:ObjectProperty	Permite definir propiedades entre instancias de clases.
owl:DatatypeProperty	Permite definir propiedades entre una instancia de una clase y tipos de datos definidos en XML Esquema o RDF.
rdfs:subPropertyOf	Define una especialización o subpropiedad de una propiedad existente.
rdfs:domain	Especifica el campo de aplicación de la propiedad, la clase o el conjunto de clases para los cuales la propiedad aplica.
rdfs:range	Especifica los posibles valores que puede tener la propiedad dependiendo del contexto donde se aplique la propiedad.

Tabla 4-2 Propiedades Simples OWL

A continuación se demuestra una ***propiedad de objeto*** que relaciona una instancia de la clase **Vino** con una instancia de la clase **VinodeUva**, mediante la propiedad **HechodeUva**. También se indica que la propiedad **HechodeUva** es una especialización de la propiedad **CosaConsumible.**

Aunque puede resultar obvio para un *ser humano*, esta propiedad permitirá a las maquinas y agentes software deducir y comprender que un *Vino de Uva* está hecho de Uva.

```
<owl:ObjectProperty rdf:ID="HechodeUva">
  <rdfs:subPropertyOf rdf:resource="#CosaConsumible" />
  <rdfs:domain rdf:resource="#Vino" />
  <rdfs:range  rdf:resource="#VinodeUva" />
</owl:ObjectProperty>
```

De igual forma se muestra una ***propiedad de tipo de datos*** por medio de la clase **AñoCosecha.** Esta se relaciona con el tipo de datos e*nteros positivos* definidos en XML esquema mediante la propiedad **ValordeAño**.

```
<owl:Class rdf:ID="AñoCosecha" />
...
</owl:Class>

<owl:DatatypeProperty rdf:ID="ValordeAño">
  <rdfs:domain rdf:resource="#AñoCosecha" />
  <rdfs:range  rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

4.2 LENGUAJE PARA ONTOLOGÍAS DE SERVICIOS WEB

La estructura del Lenguaje para la Ontología de Servicios Web *(OWL-S: Ontology Web Language-Service)*, también conocido como ontología para la descripción semántica de servicios web, está basado en tres partes que representan características esenciales de un servicio en lo que respecta a su *descripción*, *funcionamiento* y *modo de acceso*.

De esta forma, cada parte de la ontología puede asociarse con una pregunta y su respuesta así: cada pregunta representa una característica que debe ser conocida del servicio para poder utilizarlo correctamente; cada respuesta representa la parte de la ontología que proporciona esa característica. Las preguntas son:

- *¿Qué hace el servicio desde la perspectiva del cliente?* OWL-S cuenta con una clase llamada *ProfileService*.

- *¿Cómo funciona el servicio?* OWL-S cuenta con una clase llamada *ModelService.*

- *¿Cómo acceder o interactuar con el servicio?* OWL-S cuenta con una clase llamada *ServiceGrounding.*

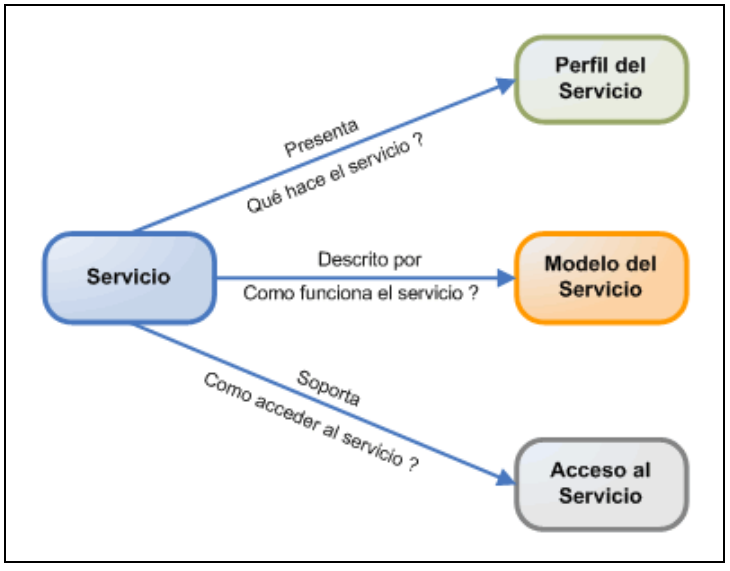

Figura 4-1 Estructura de la Ontología

Como puede verse en la anterior figura, la estructura que provee OWL-S está subordinada a una clase *Service*, la cual está directamente relacionada con el servicio web descrito. Debe existir una instancia de esta clase por cada servicio publicado.

La clase *Service* cuenta con las propiedades *presents*, *describedBy* y *supports* por medio de las cuales se puede relacionar con las clases *ServiceProfile*, *ServiceModel* y *ServiceGrounding* definidas por OWL-S para describir servicios. Cada instancia de *Service* presenta (*presents*) una instancia de *ServiceProfile*, es *descrita por* (*describedBy*) una instancia de *ServiceModel* y soporta (*supports*) una instancia de *ServiceGrounding* [OWLS04].

El *Perfil del Servicio* (S*erviceProfile*), especifica qué hace el servicio. Esto es, las capacidades del servicio como tal, lo que ofrece a los usuarios, lo que requiere de ellos y las posibles limitaciones de aplicabilidad. Además, permite a personas o agentes software determinar si el servicio satisface o no sus necesidades.

El M*odelo de Servicio (ServiceModel)*, indica al usuario cómo usar el servicio. Es decir, especifica cómo se pueden realizar peticiones al servicio y lo que debe suceder o esperarse una vez éste se haya ejecutado.

El *Acceso al Servicio (ServiceGrounding),* especifica las características de cómo un agente puede acceder al servicio. Entre ellas puede encontrarse el protocolo de comunicación y el formato de los mensajes. Entre otros detalles disponibles también está el punto final de acceso del servicio. Además, se especifica el patrón de intercambio de datos y los tipos abstractos de datos definidos en el *modelo de servicio,* es decir, las técnicas de serialización empleadas.

Resumiendo, el *perfil del servicio* suministra la información necesaria para que los agentes puedan localizar un servicio en particular. En cambio, el *modelo de servicio* y el *acceso al servicio* proveen información suficiente para que un agente pueda hacer uso de éste, una vez encontrado.

OWL-S establece las siguientes premisas respecto de la cardinalidad entre la clase *Service* y las tres partes de la ontología que permiten describirlo.

- Un servicio puede tener cero o muchos *perfiles del servicio*.
- Un servicio puede tener como máximo un *modelo del servicio* (puede tener cero)
- Un servicio puede tener cero o muchos *accesos de servicio*.

Es de anotar que las relaciones en el otro sentido, es decir, desde la ontología hacia servicio, deben tener correspondencia uno a uno. Así, por ejemplo un *perfil*, un *modelo* o un *acceso* de servicio sólo pueden ser asociados con un único servicio.

4.2.1 Perfil del Servicio

El *perfil de servicio,* describe tres tipos básicos de información acerca del mismo: el proveedor del servicio, las funcionalidades que el servicio ofrece y un conjunto de características que ayudan a detallarlo.

La información del proveedor, se refiere a los datos de contacto de la entidad que suministra el servicio. La descripción funcional del servicio es expresada en términos de las *entradas* recibidas y las *salidas* generadas. Además, se describen las *precondiciones* requeridas por el servicio y los *efectos* esperados como resultado de la ejecución del mismo.

Finalmente, el *perfil del servicio* permite describir un conjunto de propiedades que son útiles para representar características propias del servicio.

4.2.2 Construyendo un Perfil: La Relación con el Modelo del Servicio

El *perfil* proporciona una descripción concisa para los registros que actúan como repositorios de servicios, sin embargo, una vez se el servicio ha sido encontrado, el *perfil*

ya no tiene utilidad alguna. Más bien, el cliente utilizará el *modelo del servicio* para controlar la interacción con el mismo.

Las *entradas*, *salidas*, *precondiciones* y *efectos (IOPE)*[4] de un servicio web deben definirse en el *modelo* del servicio y describirse en el *perfil* del servicio.

OWL-S no impone ninguna limitación o restricción de correspondencia entre los *perfiles* y los *modelos de servicios*, de hecho las dos descripciones pueden no tener ninguna relación y aún así ser válidas. Sin embargo, si el *perfil* representa un servicio que no es consecuente con el descrito en el *modelo de proceso (modelo de servicio),* la interacción se romperá en algún punto ante la muy segura inconsistencia de las IOPE.

A manera de ejemplo supóngase que un servicio publicado como agente de viajes (perfil) adopta el *modelo de proceso* de un agente que vende libros. Con base en el perfil, el servicio será seleccionado para reservar viajes o alguna otra tarea relacionada, pero la interacción será errónea, ya que no indagará por destinos o itinerarios, sino por títulos o códigos de libros. De otro lado, el servicio nunca será localizado por otros que quieran comprar libros, puesto que en el *perfil* se indica su relación con viajes y no con libros.

4.2.3 Propiedades del Perfil

4.2.3.1 Perfil del Servicio

Describe las propiedades que enlazan la clase *Perfil del Servicio (ServiceProfile)* con las clases *Servicio (Service)* y *Modelo de Servicio (ServiceModel)*. Hay dos relaciones entre un *servicio* y un *perfil* que pueden ser expresadas mediante las propiedades *presents* y *presentedBy.*

[4] Entradas, Salidas, Precondiciones y Efectos se resumen como **IOPE** por su sigla en inglés: *inputs, outputs, preconditions, effects*.

- *presents:* describe una relación entre una instancia del *Servicio* y una instancia de *Perfil*, básicamente expresa que el *servicio* esta *descrito por* el *Perfil*.

- *presentedBy:* es el inverso de *presents*; esta especifica que un *Perfil* dado *describe* un *Servicio*.

4.2.3.2 Nombre, Contactos y Descripción del Servicio

El objetivo de estas propiedades es brindar información de contacto del proveedor al usuario del servicio en un formato completamente entendible por las personas. Estas son *serviceName*, *textDescription* y *contactInformation*. Un perfil puede tener a lo sumo un nombre de servicio y una descripción textual, pero puede tener tanta información de contacto como el proveedor lo desee.

- *serviceName:* hace referencia al nombre del servicio que esta siendo ofrecido.

- *textDescription:* proporciona una pequeña descripción del servicio en la que se incluye las siguientes características: las funcionalidades que se ofrecen, lo que requiere el servicio para funcionar correctamente e información adicional que puede ser útil para los usuarios finales.

- *contactInformation:* provee la información necesaria para poder contactar a los responsables del servicio.

4.2.3.3 Descripción Funcional

El *perfil* describe dos aspectos fundamentales de la funcionalidad del servicio, la transformación de la información (representado por *entradas* y *salidas*) y el cambio en el mundo real producido por la ejecución del servicio (representado por *precondiciones* y *efectos*).

Por ejemplo, para completar una transacción, un servicio de venta de libros espera como *entradas* un número de tarjeta de crédito y una fecha de expiración; como *precondiciones* que la tarjeta de crédito exista y tenga el cupo necesario para realizar la operación. Como resultado de la venta, es decir *salidas*, se tiene la generación y retorno de un recibo de pago que confirma la correcta ejecución de la transacción y como *efectos* en el mundo real, la correspondiente transacción bancaria y el traslado del libro desde los almacenes del vendedor a la dirección del comprador.

La ontología del *perfil del servicio* no proporciona un esquema para describir instancias de IOPE, en cambio, la ontología del *modelo del proceso* sí cuenta con un esquema para realizar esta tarea. Por lo tanto, el *modelo*, creará todas las IOPE necesarias y una instancia del *perfil* podrá apuntar a ellas.

La ontología del *perfil* define las siguientes propiedades de la *clase perfil* para representar las IOPE, que deben estar definidas en el *modelo* del servicio.

- *hasParameter:* describe *parámetros* del *modelo* del servicio.

- *hasInput:* describe *entradas* del servicio.

- *hasOutput:* describe *salidas* del servicio.

- *hasPrecondition:* es posible precisar precondiciones del servicio de acuerdo con las condiciones necesarias para ejecutarlo.

- *hasResult:* describe los cambios producidos en el mundo real durante la ejecución del servicio. Esto de acuerdo con las definiciones hechas en el *modelo del servicio* por medio de la clase *Result.*

4.2.3.4 Atributos Adicionales del Perfil

Además de la descripción funcional del servicio, hay otros aspectos importantes que los usuarios deben conocer. Entre ellos se encuentra la categoría del servicio, las garantías de calidad que se ofrecen y algunos otros parámetros adicionales que el proveedor podría especificar a sus usuarios para evitar ambigüedades.

La categoría del servicio, se refiere a la clasificación de acuerdo con sus funcionalidades y el contexto donde se pretenda ejecutar el servicio. Un ejemplo de esta clasificación puede ser: ambientales, educativos, comerciales, financieros entre otros.

Respecto de la evaluación de calidad, los servicios pueden ser muy buenos, confiables, rápidos o incluso dañinos. Es por ello que antes de usar un servicio, un solicitante puede querer verificar con qué clase de servicio se dispone a interactuar. Por consiguiente, los proveedores de servicios deben publicar la evaluación de calidad del servicio que ofrecen para que pueda ser consumido de manera confiable.

Los parámetros adicionales del servicio incluyen características como: el tiempo máximo esperado para la respuesta, la ubicación geográfica del servicio y otros que pueden ser definidos por el proveedor del servicio de acuerdo con los requerimientos del contexto donde se ejecute el servicio.

- *serviceParameter:* permite representar propiedades adicionales definidas por el creador del servicio de acuerdo con requerimientos específicos.

- *serviceCategory:* describe categorías del servicio con base en alguna clasificación que puede no estar relacionada con OWL-S e incluso con OWL.

4.2.4 El Modelo del Servicio: Modelando Servicios como Procesos

Por medio de esta parte de la ontología de servicios se describen las operaciones que el servicio realiza para el cliente, al igual que las IOPE necesarias para ejecutar cada uno de los pasos de dichas operaciones.

Un proceso puede tener dos tipos de propósito. El primero, generar y devolver nueva información basada en las entradas ingresadas y en el cumplimiento de las condiciones del mundo real impuestas con anterioridad. El segundo propósito, consiste en producir cambios en el mundo real, de acuerdo con las acciones que debe realizar el servicio.

Considerando los servicios como procesos, las *entradas* representan la información que se procesa en la ejecución del mismo; las *salidas* en cambio, la información que el proceso devuelve al usuario. Respecto de las *precondiciones* y *efectos*, se puede decir que los primeros son las condiciones necesarias para la ejecución del servicio y los segundos los cambios efectuados en el mundo real.

Para servicios no triviales (compuestos de múltiples pasos o procesos complejos), el modelo del servicio puede ser usado por un agente buscador de servicios al menos de cuatro formas diferentes: realizar análisis exhaustivos de si un servicio encontrado satisface los requerimientos de la búsqueda, componer servicios a partir de descripciones semánticas, coordinar las actividades de los diferentes participantes y monitorear la ejecución de servicios.

4.2.4.1 Parámetros y Expresiones

Las entradas y salidas pueden asociarse con los *parámetros* del servicio. Para su representación se cuenta con las clases *Input* y *Output*, que a su vez son subclases de la clase *Parameter*.

De otro lado, las precondiciones y efectos pueden considerarse como las expresiones del servicio. La representación de estas no abarca el campo de acción de OWL-S ya que

deben ser representadas mediante expresiones lógicas, reglas de inferencia y conceptos propios de inteligencia artificial. Para este fin han surgido otra familia de lenguajes, entre los que se destacan: Lenguaje para las Reglas de la Web Semántica *(SWRL: Semantic Web Rule Language)*[5], Formato para el Intercambio de Conocimiento *(KIF: Knowledge Interchange Format)*[6] y el Lenguaje para la Planeación de Definición de Dominios *(PDDL: The Planning Domain Definition Language)*[7].

4.2.4.2 Parámetros del Proceso y Resultados

Por medio de las siguientes propiedades se enlazan o interconectan los procesos con sus respectivas IOPE.

PROPIEDADES	RANGO	TIPO
hasParticipant	Participant	Cosa
hasInput	Input	Parámetro
hasOutput	Output	Parámetro
hasLocal	Local	Parámetro
hasPrecondition	Condition	Expresión
hasResult	Result	Expresión

Tabla 4-3 Propiedades para el manejo de IOPE

El dominio estas propiedades es únicamente el proceso en el que se encuentran definidas. En los siguientes tres ejemplos se definen procesos, cuya sintaxis, clasificación y conceptualización se explicará más adelante.

1. Participantes

Un proceso normalmente involucra dos o más agentes. Uno es el *cliente*, el agente desde el cual se solicita el servicio. El otro es el *servidor*, que es el elemento principal del servicio con el que interactúa el cliente. Sin embargo, es posible que esté involucrado otro agente.

---

[5] Más información acerca de SWRL en: http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/

[6] Más información acerca de KIF en: http://logic.stanford.edu/kif/dpans.html

[7] Más información acerca de PDDL en: http://ls5-www.cs.uni-dortmund.de/~edelkamp/ipc-4/

Cualquiera de los anteriores puede ser referenciado en el proceso mediante la propiedad hasParticipant.

```
<process:Participant rdf:ID="Editorial">
  …
</process:Participant>

<process:AtomicProcess rdf:ID="ProcesoPrecioLibro">
  <process:hasParticipant rdf:resource="#Editorial" />
</process:AtomicProcess>
```

2. Entradas y Salidas

Las entradas y salidas pueden ser asociadas con procesos mediante la propiedad hasParameter y sus subpropiedades hasInput, hasOutput y hasLocal. Esto quiere decir, especificar cuales son las entradas y salidas de un proceso en particular.

En el proceso únicamente se hace referencia a las entradas y salidas, pero su definición puede estar en el mismo documento o en uno diferente.

```
<process:AtomicProcess rdf:ID="ProcesoPrecioLibro">
  <process:hasInput rdf:resource="#NombreLibro" />
  <process:hasOutput rdf:resource="#Precio" />
</process:AtomicProcess>

<process:Input rdf:ID="NombreLibro">
  <process:parameterType rdf:resource="xsd:string" />
</process:Input>

<process:Output rdf:ID="Precio">
  <process:parameterType rdf:resource="xsd:float" />
</process:Output>
```

3. Precondiciones y Resultados

Si un proceso tiene una precondición, éste no puede ser ejecutado exitosamente a menos que la precondición sea verdadera. En OWL-S, si una precondición de un proceso es falsa (no se cumple) las consecuencias de la ejecución o inicialización del proceso son indefinidas.

Como se menciono antes, para poder definir precondiciones y efectos de un proceso es necesario utilizar lenguajes que permitan realizar descripciones lógicas.

De manera similar al caso anterior se puede hacer referencia a las precondiciones y efectos de un proceso mediante las propiedades hasPrecondition y hasResult respectivamente.

```
<process:AtomicProcess rdf:ID="ProcesoPrecioLibro">
  <process:Precondition rdf:resource="swrl:encontrado" />
  <process:Result rdf:resource="swrl:compra" />
</process:AtomicProcess>
```

4.2.4.3 Procesos Atómicos

Los procesos atómicos son directamente invocables, no tienen subprocesos y se ejecutan en solo paso en el cual el usuario del servicio suministra un mensaje de entrada y recibe un mensaje de salida. Estos siempre tienen sólo dos participantes: el *Cliente* y el *Servidor*.

Un mensaje puede involucrar cero, una o varias entradas o salidas, por lo que un proceso atómico, aunque conste de un sólo paso o interacción, puede tener varias entradas y salidas. Para este tipo de procesos las entradas provienen del cliente, en cambio las salidas pueden ser suministradas al cliente o a otro proceso ya sea atómico o compuesto.

En el siguiente ejemplo se detalla un proceso atómico sencillo con sus respectivas entradas y salidas.

```
<process:AtomicProcess rdf:ID="ProcesoPrecioLibro">
  <process:hasInput rdf:resource="#NombreLibro" />
  <process:hasOutput rdf:resource="#PrecioLibro" />
</process:AtomicProcess>
```

4.2.4.4 Procesos Simples

Los procesos simples no son invocables y además no tienen asociado un Acceso al Servicio (*Grounding*). Sin embargo, al igual que los procesos atómicos su ejecución consta de un sólo paso.

Comúnmente son utilizados como elementos de abstracción y también pueden ser utilizados para proveer una vista de un proceso atómico (una manera especializada de usarlo) o una representación simplificada de un proceso compuesto (para propósitos de planeación y razonamiento).

4.2.4.5 Procesos Compuestos

Un proceso compuesto puede implicar uno o más pasos en su ejecución. Es posible descomponer este tipo de procesos en otros más sencillos. Para procesos compuestos, algunas entradas pueden venir de los clientes y otras de procesos.

Básicamente pueden considerarse como una descripción concisa de la manera como interactúan varios procesos para alcanzar un objetivo común. Es importante tener en cuenta que para obtener el resultado final de un proceso compuesto es necesario completar exitosamente cada uno de los pasos que lo componen.

La representación sintáctica de un proceso compuesto en OWL-S se hace por medio de la propiedad process:CompositeProcess. Para definir sus componentes se debe utilizar la propiedad process:composedOf y un constructor de control [OWLS04].

Los diferentes tipos de constructores de control se explican en la siguiente sección. Para el ejemplo mostrado en seguida se utiliza process:Sequence. Este último define la forma de interactúan los procesos que componen el proceso compuesto, que se definen mediante la propiedad process:components.

```
<process:CompositeProcess rdf:ID="ProcesoCompraLibro">
<process:hasInput rdf:resource="#NombreLibro" />
<process:hasInput rdf:resource="#Moneda" />
<process:hasOutput rdf:resource="#Precio" />

<process:composedOf>
  <process:Sequence>
    <process:components rdf:parseType="Collection">
      <process:AtomicProcess rdf:about="http://www.ejemplo.com/Encontrar.owl#ProcesoEncontrar" />
      <process:AtomicProcess rdf:about="http:// www.ejemplo.com /Precio.owl#ProcesoPrecio" />
       <process:AtomicProcess rdf:about="http://www.ejemplo.com/Moneda.owl#ProcesoMoneda" />
    </process:components>
  </process:Sequence>
</process:composedOf>
```

4.2.4.6 Constructores de Control

Representan una de las partes más importantes y novedosas de la ontología para la descripción de servicios web OWL-S. Esto debido a que nos permiten agregar lógica a la interacción entre procesos de diferentes servicios web, es decir, componerlos de acuerdo con la lógica de negocio que se requiera.

De manera análoga a los lenguajes de programación, donde es posible mediante la manipulación del valor de variables realizar ciclos repetitivos, evaluar condiciones o sincronizar funciones, por medio de OWL-S y sus constructores de control es posible también hacerlo mediante la manipulación de las IOPE de diferentes procesos que se encuentren a su vez en distintos servicios web [OWLS04].

1. Sequence: agrupa una lista de constructores de control para ser ejecutados en orden.

2. Split: permite agrupar un conjunto de componentes de proceso que deben ser ejecutados de forma concurrente. *Split*, termina tan pronto todos sus componentes de proceso han sido fijados para ejecución.

3. Split+Join: De manera similar a *Split*, se agrupan componentes de proceso con el objetivo de sincronizarlos. Esto es, *Split+Join* termina cuando todos sus componentes de proceso han sido completados.

4. Any-Order: permite la ejecución de un grupo de componentes de proceso en un orden no especificado, pero no concurrentemente. Para la finalización de *Any-Order* se requiere que cada uno de los componentes haya terminado su ejecución.

5. Choice: habilita la posibilidad de seleccionar para ejecución un constructor de control simple de un grupo de constructores de control disponibles.

6. If-Then-Else: la semántica es intencional, esto quiere decir que si la condición del If es verdadera se ejecuta la parte del *Then*. En caso contrario se ejecuta la parte de *Else*.

7. Iterate: es una clase abstracta en el sentido de que no está suficientemente detallada para ser instanciada en un modelo de proceso, sino que está definida para servir como una superclase para *Repeat-While*, *Repeat-Until* y posiblemente otros constructores de control en un futuro.

8. Repeat-While and Repeat-Until: estas dos estructuras de control evalúan una condición para continuar o terminar un bucle repetitivo.

   Repeat-While existe si la condición que se evalúa es falsa, esto es, repite sus iteraciones hasta tanto la condición sea verdadera. Por el contrario *Repeat-Until* existe si la condición que se evalúa es verdadera y por lo tanto mantiene el ciclo de repetición hasta que la condición sea falsa.

   Es importante tener en cuenta que *Repeat-While* puede no ser ejecutada nunca, en cambio *Repeat-Until* se ejecutará al menos una vez.

4.2.4.7 Flujo de Datos y Enlace de Parámetros

En OWL-S se entiende por flujo de datos al traslado de las salidas de un componente de un proceso hacia las entradas de un componente subsecuente. Esto es, cuando una salida de un proceso (atómico o compuesto) se convierte en la entrada de otro proceso de

cualquier índole. Este procedimiento de mapear las entradas de un proceso en las salidas de otro o viceversa refleja la manera como los datos fluyen entre diferentes procesos.

Para ilustrar un proceso compuesto que utiliza un constructor de control y además establece un flujo de datos, se hacen las siguientes suposiciones:

El proceso compuesto se llama CP y está conformado por los procesos S1 y S2. El constructor de control utilizado para definir la composición es *Sequence*, esto es, CP estará conformado por la ejecución ordenada de S1 y después S2.

	Proceso Compuesto CP	**Proceso S1**	**Proceso S2**
Entradas	I1	I11, I12	I21
Salidas	O1	O11	O21

Tabla 4-4 Ejemplo de Entradas y Salidas de un Proceso Compuesto

En la Figura No 4-2, se ilustra cada uno de los procesos con sus respectivas entradas y salidas, además del flujo de datos entre ellos.

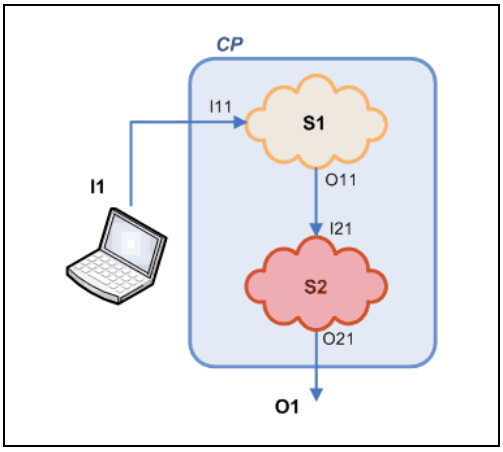

Figura 4-2 Ejemplo Composición de Servicios

El *modelo* de servicio del proceso compuesto que describe la figura anterior tiene la siguiente presentación.

```
<process:ProcessModel rdf:ID="BookPriceProcessModel">
  <service:describes rdf:resource="#BookPriceService" />
  <process:hasProcess rdf:resource="#BookPriceProcess" />
</process:ProcessModel>

<process:CompositeProcess rdf:ID="CP">
  <process:hasInput rdf:resource="#I1" />
  <process:hasInput rdf:resource="#O1" />

<process:composedOf>
  <process:Sequence>
    <process:components rdf:parseType="Collection">
      <process:AtomicProcess rdf:about="http://www.ejemplo.com/Proceso1.owl#P1" />
      <process:AtomicProcess rdf:about="http://www.ejemplo.com//Proceso2.owl#P2" />
    </process:components>
  </process:Sequence>
</process:composedOf>

<process:sameValues rdf:parseType="Collection">
  <process:ValueOf>
    <process:theParameter rdf:resource="#I1"/>
    <process:atProcess rdf:resource="#CP"/>
  </process:ValueOf>
  <process:ValueOf>
    <process:theParameter rdf:resource="http://www.ejemplo.com/Proceso1.owl#I11" />
    <process:atProcess rdf:resource="http://www.ejemplo.com/Proceso1.owl#P1" />
  </process:ValueOf>
</process:sameValues>

<process:sameValues rdf:parseType="Collection">
  <process:ValueOf>
    <process:theParameter rdf:resource="http://www.ejemplo.com/Proceso1.owl#O11" />
    <process:atProcess rdf:resource="http://www.ejemplo.com/Proceso1.owl#P1" />
  </process:ValueOf>
  <process:ValueOf>
    <process:theParameter rdf:resource="http://www.ejemplo.com/Proceso2.owl#I21" />
    <process:atProcess rdf:resource="http://www.ejemplo.com/Proceso2.owl#P2" />
  </process:ValueOf>
</process:sameValues>

<process:sameValues rdf:parseType="Collection">
  <process:ValueOf>
    <process:theParameter rdf:resource=" http://www.ejemplo.com/Proceso2.owl#O21" />
    <process:atProcess rdf:resource="http://www.ejemplo.com/Proceso2.owl#P2" />
  </process:ValueOf>
  <process:ValueOf>
    <process:theParameter rdf:resource="#O1" />
    <process:atProcess rdf:resource="#CP" />
  </process:ValueOf>
```

```
</process:sameValues>
</process:CompositeProcess>
```

4.2.5 Acceso al Servicio (Grounding): Una Realización Concreta

En un documento de descripción WSDL se separa la definición abstracta del servicio de la definición concreta del mismo. La primera describe la interfaz abstracta del servicio como el conjunto de funcionalidades ofrecidas, puestas en términos de los mensajes y de las operaciones que los agrupan. La segunda define la forma de interactuar con el servicio en cuanto a detalles de transporte, especificando además el punto de acceso a la implementación real del servicio.

El *perfil y el modelo* del servicio representan en OWL-S la representación abstracta, ya que no especifican los detalles de transporte de los mensajes, protocolos, ni enlazan con la implementación del servicio. El papel del *acceso* al servicio es proveer este conjunto de detalles de manera concreta.

Es importante aclarar que ni por medio de este último y de ninguna parte de la ontología de servicios es posible enlazar con la implementación propia del servicio. Es por ello que no es posible describir en su totalidad un servicio con OWL-S. Necesariamente debe utilizarse conjuntamente WSDL y OWL-S.

La utilización de WSDL y no otro lenguaje se debe a su estandarización, amplio uso y aceptación por la comunidad investigadora y el sector comercial. OWL-S trabaja en primera instancia con WSDL versión 1.1, sin embargo, es posible que en un futuro se adopten nuevas versiones o quizá otro lenguaje para trabajar en conjunto con la ontología para la descripción de servicios.

4.2.5.1 Relación entre WSDL y OWL-S

Los dos lenguajes no cubren el mismo espacio conceptual, sin embargo, se traslapan en un punto. Es allí donde los desarrolladores de servicios web por medio del *acceso al*

servicio deben realizar las respectivas conexiones entre la representación abstracta (OWL-S) y la representación concreta (WSDL) de un servicio.

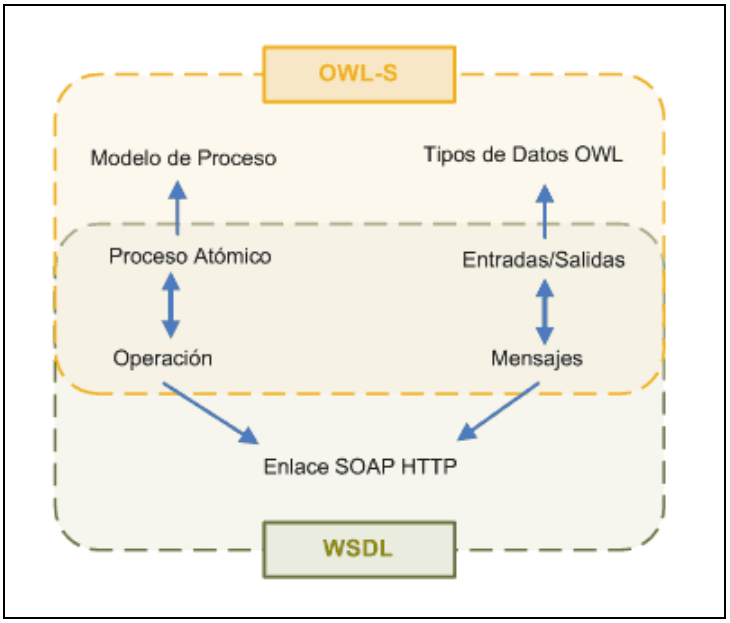

Figura 4-3 Relación entre WSDL y OWL-S

El *acceso del servicio* esta basado en las siguientes correspondencias entre OWL-S y WSDL [OWLS04].

1. Un *proceso* en OWL-S corresponde a una *operación* en WSDL versión 1.1.

   - Un *proceso atómico* con entradas y salidas constituye una *operación petición-respuesta.*
   - Un proceso atómico con entradas pero sin salidas constituye una *operación en un solo sentido.*
   - Un proceso atómico con salidas pero sin entradas constituye una *operación de notificación.*
   - Un proceso compuesto con entradas y salidas constituye una *operación solicitud respuesta.*

No es necesario que a correspondencia entre los *procesos* y las *operaciones* sea uno a uno, es decir, un *proceso* sea puede estar asociado con varias *operaciones.*

2. El conjunto de *entradas y salidas* de un proceso atómico corresponde al concepto *mensaje* en WSDL.

3. Los *tipos* de entradas y salidas de un proceso atómico corresponde a la noción de *tipos abstractos* en WSDL. Para este fin OWL-S utiliza clases OWL en cambio que WSDL usa esquemas XML. La primera opción agrega semántica y expresividad a la descripción de entradas y salidas, mientras que la segunda enlaza de manera concreta estas descripciones con su implementación.

   Una alternativa en este punto es utilizar el lenguaje de transformaciones XSLT (*XSLT: XSL Transformations*)[8] que permite transformar un documento XML en otro. Por medio de éste es posible manejar las relaciones que existen entre los tipos de datos definidos en OWL-S y los correspondientes tipos de datos definidos en WSDL.

4.2.5.2 Creación de Acceso al Servicio desde el documento OWL-S

Esta parte de la ontología de servicios se define mediante una instancia de la clase *Grounding*, la cual incluye toda la información necesaria para enlazar correctamente la descripción OWL-S con la descripción WSDL. Es lógico no tener la necesidad de modificar el documento WSDL. Sin embargo, puede ser de gran utilidad agregar al documento WSDL, información que permita la interconexión con el documento OWL-S. Debe recordarse que WSDL es extensible puesto que en sí es un documento XML.

El mecanismo establecido por OWL-S para referenciar la información necesaria del documento WSDL es la clase *WsdlGrounding*, que a su vez es una subclase de *Grounding*.

---

[8] XSLT trabaja en conjunto con el Lenguaje Extensible de Hojas de Estilo XSL (*XSL: Extensible Stylesheet Language*), el cual define reglas de transformación y presentación de documentos XML. XSTL es un complemento a XSL. URL: http://www.w3.org/TR/1999/REC-xslt-19991116

WsdlGrounding contiene una lista de instancias de *WsdlAtomicProcessGrounding,* que permiten hacer referencia a elementos específicos WSDL mediante las siguientes propiedades [OWLS04].

- *wsdlVersion:* un URI que indica la versión de WSDL que se esta usando.

- *wsdlDocument:* un URI del documento WSDL al que se hace referencia desde el documento OWL-S por medio del *acceso* al servicio.

- *wsdlOperation:* el URI que identifica la operación WSDL que se corresponde con un proceso atómico.

- *wsdlService (opcional):* el URI del servicio WSDL que ofrece la operación referenciada mediante la propiedad anterior.

- *wsdlInputMessage:* un objeto que contiene el URI del mensaje WSDL que porta las entradas de un proceso atómico dado.

- *wsdlInput:* permite asociar una entrada de un proceso atómico con una entrada de una operación WSDL. Este proceso debe estar acompañado de una instancia de la clase *wsdlInputMessagePart*.

  Por medio de las propiedades *owlsParameter* y *wsdlMessagePart* se hace referencia a la entrada del proceso atómico OWL-S y a la entrada de la operación WSDL respectivamente.

  En el caso más sencillo se pueden asociar directamente las entradas de las dos descripciones y además el tipo de dato es compatible y puede ser utilizado directamente por la especificación WSDL; basta con utilizar el URI de la entrada del proceso OWL-S. Sin embargo, si esto no es posible deben utilizarse transformaciones XSLT que hagan compatibles los tipos de datos utilizados en ambos lenguajes.

- *wsdlOutputMessage:* de manera similar a *wsdlInputMessage*, pero para salidas.

- *wsdlOutput:* de manera similar a *wsdlInput*, pero para salidas. Este proceso debe estar acompañado de una instancia de la clase *wsdlOutputMessagePart* y utiliza de la misma forma las propiedades *owlsParameter* y *wsdlMessagePart*, pero referenciado salidas y no entradas.

Para ilustrar por medio de un ejemplo el manejo de estas propiedades y la correcta interconexión entre los documentos OWL-S y WSDL, se parte del hecho que se está describiendo un servicio web que recibe como entrada un nombre de un libro y como salida devuelve el precio del mismo.

Las partes en color negro resaltado representan el puente entre el archivo OWL-S y el WSDL. Las partes en color marrón resaltado son propiedades necesarias para realizar la comunicación entre las dos descripciones de los servicios.

Es de anotar que mediante las propiedades grounding:owlsProcess y grounding:owlsParameter se hace referencia a un proceso atómico y a las entradas o salidas de dicho proceso que deben estar definidos en el modelo del servicio. Por tal razón estas definiciones no se muestran en el siguiente ejemplo, y sí se muestra la parte de acceso al servicio del documento OWL-S, el resto de partes se omite.

Documento OWL-S (http://www.ejemplo.com/preciolibro.owl)

```
<grounding:WsdlGrounding rdf:ID="PrecioLibroGrounding">
  <grounding:hasAtomicProcessGrounding rdf:resource="#ProcesoPrecioLibroGrounding" />
</grounding:WsdlGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="ProcesoPrecioLibroGrounding">
  <grounding:owlsProcess rdf:resource="#PrecioLibroOWL-S">
  <grounding:wsdlOperation>
    <grounding:WsdlOperationRef>
      <grounding:portType>
        <xsd:uriReference rdf:value="http://ejemplo.com//preciolibro.wsdl#BuscarPrecioLibroPortType" />
      </grounding:portType>
      <grounding:operation>
        <xsd:uriReference rdf:value="http://ejemplo.com//preciolibro.wsdl#BuscarPrecioLibro" />
```

```
      </grounding:operation>
    </grounding:WsdlOperationRef>
  </grounding:wsdlOperation>

<grounding:wsdlInputMessage rdf:resource="http://ejemplo.com/preciolibro.wsdl#Entrada" />
  <grounding:wsdlInput>
    <grounding:wsdlInputMessageMap>
      <grounding:owlsParameter rdf:resource="#NombreLibroOWL-S">
      <grounding:wsdlMessagePart>
        <xsd:uriReference rdf:value="http://ejemplo.com/preciolibro.wsdl#NombreLibro">
      </grounding:wsdlMessagePart>
    </grounding:wsdlInputMessageMap>
  </grounding:wsdlInput>

<grounding:wsdlOutputMessage rdf:resource="http://ejemplo.com/preciolibro.wsdl#Salida"/>
  <grounding:wsdlOutput>
    <grounding:wsdlOutputMessageMap>
      <grounding:owlsParameter rdf:resource="#PrecioOWL-S">
      <grounding:wsdlMessagePart>
        <xsd:uriReference rdf:value="http://ejemplo.com/preciolibro.wsdl#Precio">
      </grounding:wsdlMessagePart>
    </grounding:wsdlOutputMessageMap>
</grounding:wsdlOutput>

<grounding:wsdlVersion rdf:resource="http://www.w3.org/TR/2001/NOTE-wsdl-20010315">

<grounding:wsdlDocument>http://ejemplo.com/preciolibro.wsdl</grounding:wsdlDocument>
</grounding:WsdlAtomicProcessGrounding>
```

El documento WSDL no requiere ninguna modificación siempre y cuando se incluyan las propiedades correspondientes en el documento OWL-S que los enlaza.

Documento WSDL (http://www.ejemplo.com/preciolibro.wsdl)

```
<?xml version="1.0"?>
<definitions name="PrecioLibro"
  targetNamespace="http://ejemplo.com/preciolibro.wsdl"
  xmlns:tns="http://ejemplo.com/preciolibro.wsdl"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="Entrada">
    <part name="NombreLibro" type="xsd:string" />
  </message>
```

```
  <message name="Salida">
    <part name="Precio" type="xsd:float" />
  </message>

  <portType name=" BuscarPrecioLibroPortType">
    <operation name="BuscarPrecioLibro">
      <input message="tns:Entrada" />
      <output message="tns:Salida" />
    </operation>
  </portType>

  <binding name="PrecioLibroSoapBinding" type="tns:PrecioLibroPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
     <operation name="BuscarPrecioLibro">
      <soap:operation soapAction=" " />
      <input>
        <soap:body parts="NombreLibro " use="encoded" namespace="http://ejemplo.com/"
      </input>
      <output>
        <soap:body parts="Precio" use="encoded" namespace="http://ejemplo.com/"
      </output>
     </operation>
  </binding>

  <service name="ServicioPrecioLibro">
    <documentation>Devuelve el Precio de un Libro</documentation>
    <port name="PuertoComprarLibro" binding="tns:PrecioLibroSoapBinding">
      <soap:address location="http://ejemplo.com/congo/" />
    </port>
  </service>
</definitions>
```

4.2.5.3 Creación de Acceso al Servicio desde el documento WSDL

Una forma alternativa de interconectar las dos descripciones de servicios en lo que respecta a los tipos de datos, es referenciado desde el documento WSDL los tipos de datos OWL utilizados. Esto es, asociar directamente las entradas o salidas de las operaciones WSDL con los tipos de datos utilizadas en las entradas o salidas de los procesos atómicos OWL-S.

Es posible realizar este procedimiento mediante una clase OWL en la que se definen los tipos de datos utilizados en los procesos OWL-S. Dicha clase puede ser definida dentro

de la sección tipos de WSDL o en un documento aparte y posteriormente hacer referencia a ella mediante la propiedad *owl-s-parameter* [OWLS-WSLD-04].

A continuación se detallan las extensiones OWL-S que pueden utilizarse dentro de un documento WSDL:

- *owl-s-parameter:* indica el nombre de una entrada o salida de un proceso atómico OWL-S. Los tipos de datos de dichas entradas o salidas se obtienen a partir del documento OWL-S.

- *encodingStyle:* por medio de un URI se especifica la forma como serán serializadas las entradas o salidas. Si los tipos de datos son OWL el URI que se debe utilizar es http://www.w3.org/2002/07/owl/. Esta extensión OWL-S debe colocarse dentro del elemento *binding* en el documento WSDL.

- *owl-s-process:* permite indicar el nombre del proceso atómico que debe ser asociado con una operación WSDL.

Partiendo de las mismas suposiciones que en el caso anterior, a continuación se muestra un ejemplo que interconecta las dos descripciones del servicio a partir del documento WSDL. El documento OWL-S sólo debe incluir en la parte del acceso al servicio la propiedad grounding:wsdlDocument, para indicar la ubicación del documento WSDL.

Documento OWL-S (http://www.ejemplo.com/preciolibro.owl)

```
<grounding:WsdlGrounding rdf:ID="PrecioLibroGrounding">
  <grounding:hasAtomicProcessGrounding rdf:resource="#ProcesoPrecioLibroGrounding"/>
</grounding:WsdlGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="ProcesoPrecioLibroGrounding">

<grounding:wsdlVersion rdf:resource="http://www.w3.org/TR/2001/NOTE-wsdl-20010315">

<grounding:wsdlDocument>http://ejemplo.com/preciolibro.wsdl</grounding:wsdlDocument>
</grounding:WsdlAtomicProcessGrounding>
```

Por el contrario, el documento WSDL debe indicar la correspondencia entre sus entradas, salidas y operaciones con las entradas, salidas y procesos definidos en la descripción OWL-S. Para ello debe utilizarse las extensiones (resaltadas en negrilla) antes descritas, las cuales representan el puente entre los dos documentos.

Documento WSDL (http://www.ejemplo.com/preciolibro.wsdl)

```
<?xml version="1.0"?>
<definitions name="CongoBuy"

targetNamespace="http://ejemplo.com/preciolibro.wsdl"
  xmlns:tns="http://ejemplo.com/preciolibro.wsdl"
  xmlns:preciolibro="http://ejemplo.com//preciolibro.owl#"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:owl-s-wsdl="http://www.daml.org/services/owl-s/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <message name="Entrada">
    <part name="NombreLibro" owl-s-wsdl:owl-s-parameter="preciolibro:NombreLibroOWL-S" />
  </message>

  <message name="Salida">
    <part name="Precio" owl-s-wsdl:owl-s-parameter=" preciolibro: PrecioOWL-S " />
  </message>

  <portType name="BuscarPrecioLibroPortType">
    <operation name="BuscarPrecioLibro" owl-s-wsdl:owl-s-process="preciolibro: PrecioLibroOWL-S ">
      <input message="tns:Entrada"/>
      <output message="tns:Salida"/>
    </operation>
  </portType>

  <binding name="PrecioLibroSoapBinding" type="tns:BuscarPrecioLibroPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="BuscarPrecioLibro">
        <soap:operation soapAction="http://ejemplo.com/preciolibro.wsdl#BuscarPrecioLibro"/>
        <input>
          <soap:body parts="NombreLibro" use="encoded"
                namespace="http://ejemplo.com/"
                encodingStyle="http://www.w3.org/2002/07/owl/" />
        </input>
        <output>
          <soap:body parts="Precio" use="encoded"
                namespace="http://ejemplo.com/"
                encodingStyle="http://www.w3.org/2002/07/owl/" />
```

```
        </output>
      </operation>
  </binding>

  <service name="ServicioPrecioLibro">
    <documentation> Devuelve el Precio de un Libro </documentation>
    <port name="PuertoPrecioLibro" binding="tns:PrecioLibroSoapBinding">
      <soap:address location="http://ejemplo.com/" />
    </port>
  </service>
</definitions>
```

Finalmente, en el Anexo B – Ejemplo de Composición de Servicios web con OWL-S, se puede encontrar información adicional de cómo describir y ejecutar un servicio compuesto con OWL-S.

5. ONTOLOGÍA PARA MODELAR SERVICIOS WEB

La Ontología para Modelar Servicios Web (*WSMO: Web Services Modeling Ontology*) es una iniciativa del Grupo de Trabajo de Servicios Web Semánticos[9], que a su vez hace parte del Grupo SDK[10], el cual incluye más de 50 socios académicos e industriales.

El objetivo de esta iniciativa es resolver el problema de integración mediante una tecnología coherente que habilite el uso e implementación de los servicios web semánticos.

WSMO únicamente define el **modelo conceptual** para describir este nuevo tipo de servicios web, sin embargo, se apoya en otras dos iniciativas para su representación sintáctica y su ejecución [WSMO04].

El **Lenguaje para Modelar Servicios Web** (*WSML: Web Services Modeling Language*) se encarga de la descripción sintáctica de los servicios siguiendo el modelo establecido por WSMO.

Por otro lado el **Entorno de Ejecución para Modelado de Servicios Web** (WSMX: Web Services Modeling Execution Enviroment) tiene la responsabilidad de soportar la **ejecución** de los servicios web semánticos descritos con WSML y basados en el paradigma definido por WSMO.

5.1 WSMO

WSMO toma como punto de partida la Infraestructura para Modelar Servicios Web[11] (*WSMF: Web Services Modeling Framework*), para la definición del modelo conceptual.

---

[9] Más información en http://www.wsmo.org

[10] Más información en http://www.sdk-cluster.org

[11] Más información en http://informatik.uibk.ac.at/~c70385/wese/wsmf.paper.pdf

Los principios en los que se basa WSMO son:

- Modularización y Mediación: dividir funcionalidades en componentes con tareas especificas es una de las premisas fundamentales, al igual que facilitar la manipulación de recursos provenientes de diferentes fuentes.

- Interfaz vs Implementación: cuando se describen interacciones entre entidades, es importante reconocer las diferencias que existen entre una interfaz que describe una interacción y la implementación interna de la entidad. Es decir, su comportamiento externo e interno, WSMO se enfoca únicamente en el comportamiento externo.

5.1.1 Ontologías

Representan el elemento principal de WSMO y cumplen con dos propósitos específicos, definir semántica formal y hacer entendible por los computadores los conceptos que se definen. Las partes que constituyen la descripción de una ontología son: propiedades no funcionales, ontologías importadas, uso de mediadores, axiomas, conceptos, relaciones, funciones e instancias.

- Conceptos: son las ideas básicas que se intentan formalizar, habitualmente se asocian con jerarquías que pertenecen a un dominio específico.

- Relaciones: describen interdependencias entre parámetros, cada parámetro tiene como dominio un cierto concepto.

- Funciones: son un tipo especial de relaciones que tiene un único rango.

- Instancias: se utilizan para representar objetos determinados de un concepto o una relación.

- Axiomas: son expresiones lógicas que se definen sobre los conceptos o las relaciones.

5.1.2 Metas

Específica los objetivos que un cliente tiene cuando consulta un servicio web, es decir, las funcionalidades que el servicio provee desde la perspectiva del cliente. La descripción de una meta está compuesta por un conjunto de propiedades no funcionales, poscondiciones y efectos. Es posible que existan diferencias entre las ontologías utilizadas por los usuarios para definir metas y las ontologías utilizadas para describir servicios. En este caso se deben utilizar mediadores, concepto que se explica más adelante.

5.1.3 Servicios Web

La descripción de un servicio web en WSMO consiste de 3 elementos básicos que son: Propiedades no Funcionales, Capacidades e Interfaces. En cualquiera de estos tres elementos se puede hacer uso de mediadores o de otras ontologías.

5.1.3.1 Propiedades no Funcionales

Como su nombre lo dice proveen información adicional para el uso concreto de servicios web, sin afectar su uso funcional. Además de las propiedades generales, WSMO permite describir propiedades no funcionales que permitan tener en cuenta aspectos como: calidad del servicio, aspectos de rendimiento, seguridad, robustez e incluso información financiera de los costos de su uso.

5.1.3.2 Capacidades

Define la funcionalidad de un servicio desde el punto de vista del proveedor en términos de las precondiciones, poscondiciones, suposiciones y efectos.

5.1.3.3 Interfaces

La interfaz de un servicio web provee información adicional acerca de cómo se logra obtener la funcionalidad de un servicio.

- Interfaz de Coreografía: define los patrones de comunicación que es necesario establecer para poder consumir el servicio [C-WSMO04].

- Interfaz de Orquestación: descripción de cómo se logra alcanzar la funcionalidad general de un servicio compuesto por medio de la cooperación entre diferentes servicios [O-WSMO04].

Es importante anotar que estas interfaces no se encuentran implementadas y además no se han definido en su totalidad, aspecto que se convierte en una limitación importante para la composición servicios.

5.1.4 Mediadores

Son uno de los aspectos novedosos de WSMO y buscan solucionar los problemas de heterogeneidad que pueden existir en sistemas distribuidos donde los datos provienen de múltiples fuentes.

- Mediadores de Ontologías: Encargado de resolver posibles incompatibilidades entre ontologías como por ejemplo las diferencias entre ontologías hechas en diferentes lenguajes o incluso en conceptualizaciones de un mismo dominio.

- Mediadores de Metas: Por medio de esta clase mediadores es posible realizar especializaciones, refinamientos y restricciones entre dos metas.

- Mediadores Metas-Servicios: Su tarea consiste en enlazar las metas de los usuarios con los servicios web que más se ajusten a los requerimientos del cliente.

- Mediadores de Servicios: Su misión es resolver los problemas de incompatibilidad e incongruencias en la interacción de dos servicios web.

5.2 WSMX

WSMX es un ambiente de ejecución para localización dinámica, mediación e invocación de servicios web semánticos. Esta basado en el modelo conceptual que define WSMO.

En términos generales el funcionamiento de este ambiente de ejecución se puede resumir como sigue, primero se deben registrar los servicios, cuya descripción debe estar basada en WSML y seguir el modelo conceptual que establece WSMO.

Posteriormente, un cliente puede solicitar una meta por medio de unas interfaces predefinidas para este fin. WSMX interpreta la meta del cliente y se encarga de enlazarla con la capacidad del servicio registrado que más se adapte a ésta. Si es necesario se solucionan las inconsistencias que puede haber entre las ontologías que utiliza el cliente y el servicio, finalmente se invoca el servicio web seleccionado.

Hay dos aspectos operacionales que la arquitectura de WSMX maneja de forma independiente, la *compilación* y la *ejecución*. Por *compilación* se entiende el mecanismo de dejar listos los diferentes componentes asociados con los servicios web semánticos, para el uso de otros componentes de WSMX. El concepto de *ejecución* involucra el proceso de localizar e invocar el servicio web que mejor se adapte a los requerimientos del cliente [WSMX04].

Es importante tener cuenta que para poner en funcionamiento este ambiente de ejecución de servicios web semánticos, se requiere de un **servidor de aplicaciones**, lo cual demanda un alto procesamiento para la máquina en la que se pretenda alojar.

5.2.1 Arquitectura WSMX

La arquitectura de WSMX está organizada por capas y sigue el patrón de invocación descendente, es decir, los componentes de capas superiores pueden invocar componentes de capas inferiores pero el caso contrario no es posible. Esta restricción

tiene el objetivo de evitar invocaciones circulares, sin embargo, los componentes de un mismo nivel si pueden invocarse entre sí.

En la figura 5-1 se muestra cada una de las capas de WSMX y sus respectivos componentes.

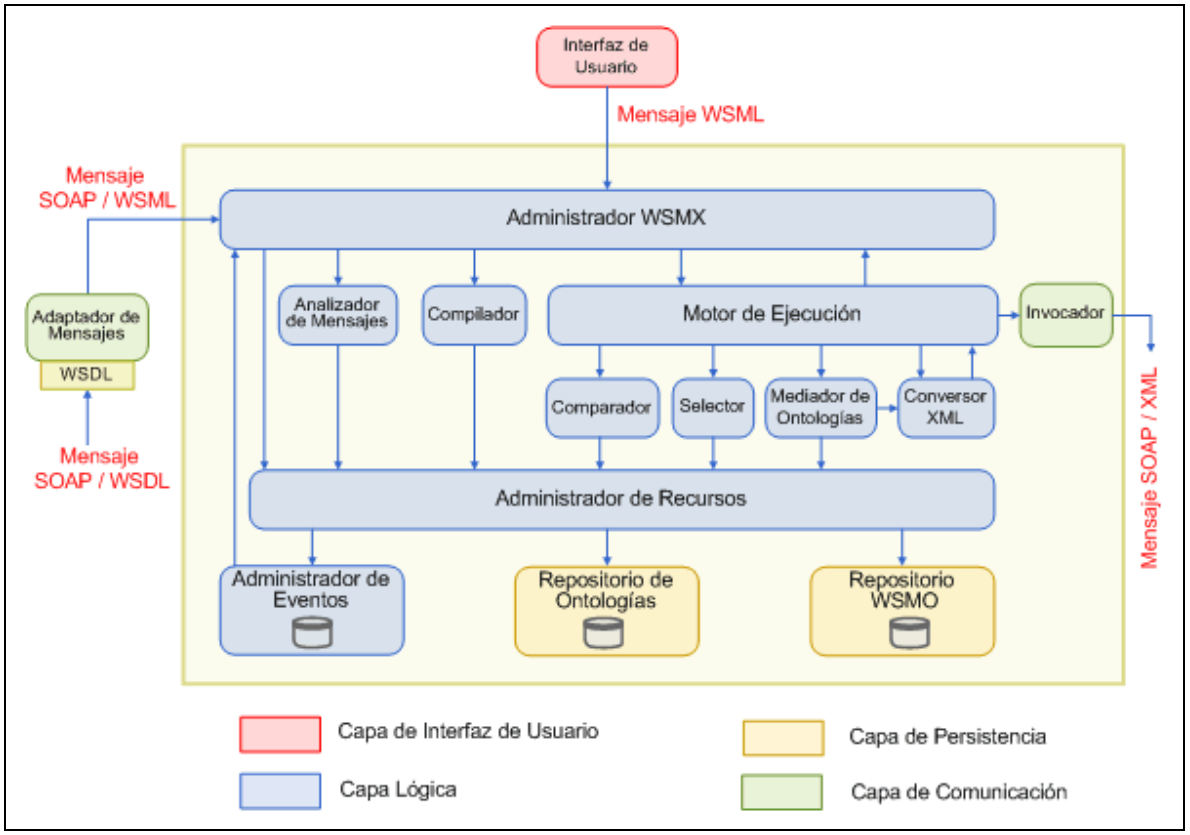

Figura 5-1 Arquitectura WSMX [WSMX04]

5.2.1.1 Capa de Interfaz de Usuario

Actualmente el único componente de este nivel es el *Editor de WSMO* que permite a los usuarios crear descripciones WSML de servicios web, ontologías, mediadores y metas. Posteriormente se envían estas descripciones a WSMX para que sean compiladas.

5.2.1.2 Capa Lógica

- Administrador WSMX: maneja la lógica interna del sistema, es decir, se encarga de gestionar la interacción entre los diferentes componentes de WSMX mediante eventos. Dichos eventos se identifican al interior del sistema mediante un tipo y un estado.

  Así por ejemplo, una vez se haya pasado una descripción WSML al *Compilador*, el *Administrador WSMX* se encarga de devolver un mensaje de error o de éxito a la interfaz de usuario indicando el resultado de la operación.

- Administrador de Eventos: este componente almacena los eventos y sus respectivos estados cuando suceden dentro del sistema.

- Motor de Ejecución: controla los procesos de comparación, selección, mediación e invocación de servicios web semánticos.

- Comparador: es el responsable de enlazar las metas de los usuarios con las capacidades de los servicios.

- Selector: es invocado en el caso de que múltiples servicios sean encontrados para satisfacer una meta específica, su responsabilidad consiste en escoger el servicio que mejor se adapte a los requerimientos del cliente.

- Mediador: transforma los datos basados en conceptos de una ontología en datos basados en otra ontología diferente, el mapeo esta basado en reglas definidas entre conceptos. En la actualidad, la arquitectura de WSMX sólo implementa el *Mediador de Ontologías.*

- Conversor XML: puede ser invocado para traducir los resultados del *Mediador* en formato XML.

- Compilador: se encarga de analizar la validez sintáctica de los mensajes WSML recibidos desde el editor WSMO, después de realizar este análisis el mensaje se almacena en el repositorio WSMX. Una vez que los elementos han sido compilados, estos quedan disponibles para su ejecución a través de metas.

- Analizador de Mensajes: su misión es similar al del componente anterior, con la diferencia que analiza los mensajes WSML en tiempo de ejecución.

5.2.1.3 Capa de Comunicación

- Adaptador de Mensajes: permite a otras aplicaciones comunicarse con WSMX sin utilizar las interfaces que define este último para la comunicación.

- Invocador: es el responsable de realizar las invocaciones a los servicios web.

5.2.1.4 Capa de Persistencia

- Repositorios: almacenan definiciones de metas, servicios web, ontologías y mediadores. Dentro de estos se encuentran un *Repositorio de Ontologías* (almacenamiento de Ontologías) y un *Repositorio WSMO* (almacenamiento de descripciones WSML basadas en WSMO).

- Administrador de Recursos: gestiona el almacenamiento persistente del sistema.

5.3 WSML

Es una familia de lenguajes que permiten la descripción de servicios web semánticos y la definición de ontologías. WSML cuenta con diferentes variantes u alternativas para los diferentes propósitos que se deben cumplir en un contexto semántico.

Dichas alternativas se diferencian en el nivel de expresividad que ofrecen y también en la capacidad de razonamiento que agregan a las máquinas.

No es posible pensar en localización, composición y ejecución automática de servicios web sin la utilización de mecanismo de inferencia, que habilite a las máquinas para **razonar** e **interpretar** los datos y la lógica que maneja un servicio tal y como lo hace una persona. Esto es precisamente lo que busca habilitar WSML.

Los componentes de esta familia de lenguajes son: WSML-Básico (*WSML-Core*), WSML-Intermedio (*WSML-Flight*), WSML-Regla (*WSML-Rule*), WSML-Lógica de Descripción (*WSML-Description Logics*) y WSML-Completo (*WSML-Full*).

5.3.1 WSML-Básico

WSML Básico está basado en las Descripciones Lógicas de OWL[12] (*OWL-DL: OWL Description Logics*).

De todos los componentes de esta familia de lenguajes es el que menos expresividad permite definir. Facilita la realización de las siguientes tareas: modelado de ontologías, representación de expresiones lógicas y la definición de tipos de datos.

WSML Básico tiene muchas limitaciones respecto del modelo conceptual que define WSMO, así por ejemplo, no es posible especificar restricciones sobre los atributos como el rango, el dominio o la cardinalidad. También presenta serias limitaciones en la sintaxis de las expresiones lógicas [WSML04].

5.3.2 WSML-Intermedio

WSML Intermedio es sintáctica y semánticamente más completo que WSML Básico, su estructura esta basada en OWL-Flight[13] y en F-Logic[14].

[12] Más información en http://www.w3.org/TR/owl-ref/

[13] Más información en http://www.w3.org/TR/owl-ref/

[14] Más información en http://www.wsmo.org/2004/d16/d16.2/v0.1/20040324/

Las principales características que hacen más robusto WSML-Intermedio con respecto a WSML-Básico, son:

- Permite representar restricciones de cardinalidad entre atributos.
- Definición de atributos para dominios abstractos.
- Habilita la representación de expresiones lógicas negativas.
- Se cuenta con un vocabulario más amplio para la definición de ontologías y descripciones semánticas de servicios web.

5.3.3 WSML-Regla

Este lenguaje todavía no esta especificado, pero se cree que será una extensión de WSML-Intermedio, enfocado hacia la definición de reglas lógicas que permitan realizar inferencias.

5.3.4 WSML-Lógica de Descripción

Será una extensión de WSML Básico que habilitará completamente las descripciones lógicas con una expresividad similar a la de OWL-DL. Actualmente su definición se encuentra en etapa desarrollo.

5.3.5 WSML-Completo

De igual forma que en los casos anteriores, este componente de WSML todavía no esta desarrollado. Teóricamente WSML-Completo combinará las principales características de WSML-DL y WSML-Regla.

6. ANÁLISIS DE LAS ALTERNATIVAS TECNOLÓGICAS PARA LA COMPOSICIÓN DE SERVICIOS WEB

En el presente capítulo se expone un análisis cualitativo de las diferentes alternativas que existen para componer servicios web. Dicho análisis está enfocado únicamente hacia el estudio de las características de cada tecnología en cuanto a composición, es decir, no se tienen en cuenta sus bondades en campos como localización y monitoreo de servicios.

El estudio incluye la comparación de las diferentes alternativas, las ventajas y desventajas de cada una de ellas, las herramientas disponibles y una serie de criterios para la selección.

El objetivo de este análisis es seleccionar una de las alternativas estudiadas en los capítulos anteriores y con ésta realizar un **prototipo** que **valide** la **composición de servicios web** como solución para el Sistema de Gestión de Cuentas de la Red de Datos de la Universidad del Cauca.

Existen muchas alternativas para el desarrollo de una aplicación web que sirva como solución para este sistema de gestión, que pueden o no involucrar servicios web. La utilización de composición de servicios web, puede no ser la alternativa más robusta, ni la que más ventajas brinde a la solución del problema, debido precisamente a su inmadurez como tecnología. Sin embargo, representa una opción invaluable incursionar en este campo por medio de un caso práctico que se adapta a la perfección al contexto académico e investigativo de la Universidad del Cauca.

Es importante tener cuenta que el prototipo representa el **medio** que permite validar la composición de servicios en un entorno real, por lo tanto, lo importante no es el desarrollo de una lógica complicada para cada servicio, sino la interacción de los diferentes servicios mediante la utilización de una de las tecnologías disponibles para ello.

Los detalles del prototipo en cuanto a su arquitectura, componentes y funcionalidades se pueden encontrar en el siguiente capítulo.

6.1 COMPARACIÓN ENTRE LAS ALTERNATIVAS

6.1.1 Comparación entre OWL-S y BPEL4WS

Los modelos conceptuales de estos dos lenguajes pueden considerarse superpuestos, ya que ambos permiten representar procesos de negocio complejos mediante la composición de servicios web. La diferencia entre BPEL4WS y OWL-S puede identificarse en los objetivos con que fueron diseñados y no en las tareas que permiten realizar.

OWL-S esta enfocado en servicios web semánticos (interpretables por computadores), los cuales deben estar descritos con suficiente información (semántica explicita) para lograr objetivos específicos como la localización, invocación, monitoreo y composición automática de servicios web.

Por otro lado el enfoque de BPEL4WS se centra en proveer un lenguaje robusto para describir interacciones de procesos, flujos de control, flujos de datos, excepciones y condiciones de error, que permiten componer servicios web a partir de sus documentos de descripción WSDL.

Se puede asociar un proceso BPEL4WS con lo que representa el *modelo de proceso* en OWL-S*,* ya que cumplen prácticamente la misma función; es en este punto donde se traslapan los modelos conceptuales. Sin embargo, existe una diferencia explícita ya que en el segundo no se especifica nada acerca de la ejecución del servicio, mientras que en el primero esto sí es posible. Lo anterior debido a que la ejecución y el punto de acceso al servicio se describen por medio de la ontología *acceso al servicio* de OWL-S.

Las últimas tendencias apuntan a utilizar BPEL4WS para describir los *modelos de proceso* definidos en OWL-S, lo que aprovecharía la potencialidad de ambos lenguajes y separaría sus modelos conceptuales.

6.1.2 Comparación entre WSMO y OWL-S

Tanto WSMO como OWL-S consideran a las ontologías un elemento fundamental para hacer realidad la localización, invocación, monitoreo y composición automática de servicios web. Sin embargo, a pesar de coincidir en esto, difieren en la forma que establecen para lograr la consecución de estas tareas.

Mientras OWL-S define un conjunto de ontologías que permiten la manipulación de servicios web respecto de su descripción, composición y acceso al servicio como tal, WSMO define un modelo conceptual dentro del cual se deben crear las ontologías que permitan administrar los servicios web. Dicho modelo está basado en seis elementos que son: ontologías, metas, servicios web, capacidades, interfaces y mediadores.

Una de las diferencias más importantes entre estas dos alternativas para realizar servicios web semánticos es el concepto de *mediadores,* definido en WSMO para resolver los posibles problemas de heterogeneidad e interoperabilidad que se pueden presentar en ambientes distribuidos, en lo que respecta a las ontologías y servicios utilizados en las composiciones. Los mediadores son elementos especiales cuyo objetivo es enlazar componer heterogéneos mediante el mapeo de sus componentes, transformaciones necesarias e incluso asociaciones inteligentes o inferencias.

En OWL-S, por medio del *perfil del servicio* se especifica lo que éste hace, tanto desde el punto de vista del cliente como del proveedor. En WSMO este concepto se divide en *metas* y *capacidades*, el primero especifica los objetivos que el cliente tiene cuando consulta un servicio, mientras que el segundo representa lo que hace el servicio desde el punto de vista del proveedor (lo que ofrece).

El *modelo del servicio* en OWL-S describe la funcionalidad del servicio y la manera de interactuar con el mismo. Estas dos tareas se llevan a cabo en WSMO por medio de las *capacidades del servicio* en lo respecta a la descripción funcional y las *interfaces de coreografía y orquestación,* quienes suministran información acerca de la manera de interactuar con el servicio.

La interfaz de coreografía proporciona la información necesaria para que un usuario pueda comunicarse e interactuar con el servicio desde la perspectiva del cliente. La interfaz de orquestación describe cómo trabaja el servicio desde el punto de vista del proveedor (especifica cómo se hace uso de otros servicios para alcanzar sus capacidades).

Es importante anotar que estas interfaces que define WSMO aún no están implementadas, por tanto no hay certeza acerca de los elementos (lenguajes u ontologías) que se van a utilizar para este fin, ni de la forma en que van a combinar. Sin embargo, dentro de las alternativas que se están evaluando están precisamente el *modelo de proceso* de OWL-S y BPEL4WS.

OWL-S especifica los detalles de cómo acceder al servicio por medio de un documento WSDL al que se hace referencia desde una parte de la ontología llamada *acceso al servicio*. En la actualidad WSMO no especifica nada acerca de este punto, pero se presume que por medio de las interfaces de *orquestación* y *coreografía* se enlace la descripción abstracta del servicio con las descripción concreta del mismo y por tanto con su implementación.

Una gran ventaja de WSMO respecto de OWL-S es el nivel de organización de los lenguajes que lo componen y la especificación de la función que cada uno de ellos puede desempeñar. Esta situación no es totalmente clara en la ontología de servicios, puesto que no se especifica cuales lenguajes pueden utilizarse de manera conjunta con la ontología y tampoco la forma como éstos deben ser usados, como es el caso de los lenguajes para representar expresiones lógicas.

Finalmente, puede considerarse que WSMO presenta un modelo conceptual más robusto, mientras que OWL-S es una tecnología mucho más madura y a diferencia de la primera cuenta en la actualidad con herramientas que permiten describir y **ejecuta**r servicios web semánticos.

6.1.3 Comparación entre WSMO y BPEL4WS

La comparación de estas dos alternativas se dificulta debido a que el espacio conceptual que cubre BPEL4WS aún no ha sido implementado en WSMO. Como ya se mencionó, las interfaces de *orquestación* y *coreografía* que representan la funcionalidad que ofrece BPEL4WS aún no han sido detalladas en WSMO.

Se conoce que dichas interfaces realizarán tareas como la interacción del cliente, con el servicio y el uso de otros servicios para cumplir metas, pero aún está por definir quién se encargará de tareas como el manejo de flujos de control, flujos de datos, errores, excepciones e interacciones entre procesos.

Como se mencionó anteriormente BPEL4WS no agrega semántica a la descripción de los servicios como si lo hacen OWL-S y WSMO, pero en cambio es un lenguaje mucho mas robusto para realizar descripciones de cómo interactúan diferentes procesos.

6.2 HERRAMIENTAS PARA LA COMPOSICIÓN DE SERVICIOS WEB

A continuación se exponen brevemente las diferentes herramientas que existen para componer servicios, organizadas de acuerdo con el lenguaje u ontología que permiten manejar.

6.2.1 Herramientas para BPEL4WS

6.2.1.1 Ambiente de Ejecución Java para BPEL4WS[15]

Es un entorno de ejecución de procesos BPEL4WS desarrollado por IBM y basado en Java. Por medio de una aplicación web es posible gestionar dichos procesos.

[15] Más información en http://www.alphaworks.ibm.com/tech/bpws4j

Para cada proceso, el entorno de ejecución requiere de un documento BPEL4WS (que describe el proceso) y los documentos WSDL de cada uno de sus socios. Al publicar el proceso este queda listo para ser invocado por medio de una interfaz SOAP, como cualquier otro servicio web.

6.2.1.2 Gestor de Procesos BPEL4WS de Oracle[16]

El entorno de desarrollo de Oracle llamado *JDeveloper*[17] proporciona un entorno gráfico que permite la creación, despliegue y monitoreo de procesos BPEL4WS. Esta misma funcionalidad puede ser incluida en el IDE Eclipse[18] como un plugin.

Para la ejecución de dichos procesos, se cuenta con un motor de ejecución basado en Java que puede ser usado en cualquier servidor de aplicaciones J2EE. Junto a este motor se incluye una aplicación web que permite gestionar y monitorear procesos.

6.2.2 Herramientas para OWL-S

Existe una gran variedad de herramientas, proyectos de investigación, aplicaciones y esfuerzos colaborativos entorno a OWL-S que se han desarrollado desde su lanzamiento en el 2.001. A continuación se presentan únicamente las herramientas más importantes que permiten la manipulación de la ontología como tal y en consecuencia la descripción semántica de los servicios.

6.2.2.1 API OWL-S [19]

Es una API para la plataforma Java que permite leer, **ejecutar** y escribir descripciones de servicios web hechas con la versión 1.0 de OWL-S.

---

[16] Más información en http://www.oracle.com/technology/products/ias/bpel/index.html

[17] Más información en http://www.oracle.com/technology/products/jdev/index.html

[18] Más información en http://www.eclipse.org

[19] Más información en http://www.mindswap.org/2004/owl-s/api/doc/

Un aspecto de gran importancia de esta herramienta con respecto a las otras es que permite la **ejecución** de las descripciones hechas, característica no ofrecida por ninguna otra herramienta, ya que se centran únicamente en la descripción.

6.2.2.2 Plugin para *Protégé*[20]

Protégé[21] es una herramienta muy poderosa que permite la manipulación de cualquier clase de ontologías mediante la instalación de plugins.

El plugin OWL-S habilita la manipulación de esta ontología por medio de un entorno gráfico que facilita el proceso de descripción de un servicio web. Como resultado final de la interacción con el entorno gráfico se obtiene un documento OWL que incluye la descripción del servicio con base a la versión 1.1 de OWL-S.

Actualmente la única herramienta que permite ejecutar servicios cuyas descripciones están basadas en OWL-S es la API descrita con antelación. Sin embargo, ésta sólo permite ejecutar servicios cuyas descripciones estén basadas en la versión 1.0 de OWL-S, por esta razón no es posible ejecutar un servicio descrito con la versión 1.1 de OWL-S.

6.2.2.3 CODE: Ambiente de Descripción OWL-S[22]

Es un entorno de desarrollo para OWL-S realizado por la Universidad *Carnegie Mellon*, que permite construir descripciones de servicios web semánticos en un entorno amigable. Al igual que en el caso anterior, debido a que las descripciones se realizan con base en la versión 1.1 de OWL-S no es posible **ejecuta**r el servicio descrito con la ontología de servicios.

---

[20] Más información en http://owlseditor.semwebcentral.org/

[21] Más información en http://protege.stanford.edu/

[22] Más información en http://www.daml.ri.cmu.edu/code/code.pdf

6.2.3 Herramientas para WSMO

6.2.3.1 Estudio SWWS[23]

Editor gráfico para servicios web semánticos basado en el modelo conceptual que define WSMO. Una característica importante de esta herramienta es que permite **modelar** la composición de procesos de manera totalmente gráfica

6.2.3.2 IRS III: Internet Reasoning Service[24]

Define un entorno de desarrollo para servicios web semánticos basado en WSMO, incluye repositorios de ontologías y el manejo de lenguajes para representar expresiones lógicas.

6.2.3.3 API para WSMO[25]

API para la plataforma Java que facilita la construcción de servicios web semánticos basados en el modelo conceptual que define WSMO.

6.2.3.4 Verificador de WSML[26]

Es una herramienta en línea que permite validar la sintaxis de un documento WSML, mediante el URL donde se encuentra dicho documento.

---

[23] Más información en http://stronghold.sirma.bg/swws/

[24] Más información en http://kmi.open.ac.uk/projects/irs/

[25] Más información en http://wsmo4j.sourceforge.net

[26] Más información en http://dev1.deri.at:8080/wsml/

6.3 CRITERIOS DE SELECCIÓN

A continuación se expone una serie de criterios, con base en los cuales se determina cuál de las alternativas tecnológicas expuestas anteriormente es la mejor para componer servicios web, de acuerdo con el entorno donde se va aplicar dicha composición.

El **por qué** de estos criterios obedece a que permiten seleccionar una opción que cumpla en su totalidad con los **objetivos** planteados para el trabajo de grado y además se tienen en cuenta las condiciones impuestas por el entorno donde se va a implantar el prototipo.

Otro elemento importante para la selección de estos criterios, es que permitan realizar un **aporte** significativo a la **línea de investigación de servicios web** de la Universidad del Cauca. Dicho aporte, busca derivar en futuros trabajos de investigación que facilitarán tanto a estudiantes como profesores estar a la vanguardia en este campo del conocimiento.

6.3.1 Madurez de las Tecnologías

El concepto de composición de servicios web surgió hace poco tiempo, por lo tanto la mayoría de las opciones tecnológicas se encuentran en etapa de desarrollo. Por esta razón es importante evaluar mediante este criterio si el estado actual de la tecnología define de forma concreta como **describir** y **ejecutar** un proceso compuesto. Es decir, se busca determinar si la parte de la tecnología relacionada con composición ya esta definida e implementada y además es posible utilizarla.

6.3.2 Descripción Semántica

De acuerdo con los objetivos planteados en el trabajo de grado es un requerimiento que la descripción de los servicios utilizados en la composición incluya semántica. Lo anterior implica mejorar las descripciones WSDL de los servicios mediante el uso de ontologías.

6.3.3 Disponibilidad y Funcionalidad de Herramientas

Tanto las tecnologías como las herramientas para composición de servicios web se encuentran en desarrollo. Por ello es importante evaluar mediante este criterio, si el estado actual del conjunto de herramientas con que se cuenta para cada tecnología permite **describir** y **ejecutar** composición de servicios web.

Es importante tener en cuenta que lo se busca es utilizar un herramienta que permita realizar lo antes descrito y no involucrarse en el proceso de desarrollo de una. No hace parte de los objetivos del trabajo de grado el desarrollo de ningún tipo de herramienta que facilite la descripción y composición semántica de servicios.

6.3.4 Aceptación en el Entorno Académico e Investigativo

El aporte que se hace al estado del arte de los servicios web en el contexto investigativo de la Universidad del Cauca, debe estar **fundamentado** en tecnologías con proyección a futuro, con gran aceptación a nivel internacional, con apoyo de la comunidad investigativa mundial y sobretodo cuya tendencia sea convertirse en estándares. Por esta razón es fundamental que la tecnología utilizada cuente con las características antes citadas.

6.3.5 Adecuación a la Arquitectura

El contexto o entorno donde se debe implementar el prototipo de gestión de cuentas de usuario, esta muy ligado con la arquitectura y equipos por medio de los cuales la Red de Datos de la Universidad del Cauca ofrece servicios a sus usuarios. Una cuenta de usuario involucra cuatro servidores diferentes, cada uno de ellos con funcionalidades específicas y limitaciones.

Estos equipos son los servidores de correo (Atenea y Afrodita), el servidor de directorio (Juno) y el servidor web (Acuario). Con excepción de Acuario, en ninguno de los anteriores es posible alojar un servidor web que soporte Java, debido principalmente a la

carga manejada por las máquinas. Tampoco se puede pensar en usar un servidor de aplicaciones en alguno de los equipos que conforman la infraestructura para la gestión de una cuenta de usuario.

Por ello la opción elegida no debe demandar, para su correcto funcionamiento, un alto procesamiento de maquina. Resumiendo, este indicador representa la **viabilidad** de aplicar la opción tecnológica de acuerdo con los requerimientos que demanda para poder funcionar.

6.4 SELECCIÓN DE UNA ALTERNATIVA TECNÓLOGICA PARA LA COMPOSICIÓN DE SERVICIOS WEB

A continuación se muestra una tabla donde se relaciona cada tecnología con los criterios definidos con antelación. Por medio de una **X** se indica si la tecnología cumple con el criterio.

	BPEL4WS	OWL-S	WSMO
Descripción Semántica		X	X
Disponibilidad y Funcionalidad de Herramientas	X	X	
Madurez de la Tecnología	X	X	
Aceptación en el Entorno Académico e Investigativo	X	X	X
Adecuación a la Arquitectura	X	X	

Tabla 6-1 Criterios de Selección de la alternativa de composición de servicios web

Como resultado final de este análisis se puede concluir lo siguiente para la viabilidad de utilización de cada opción tecnológica en el entorno demarcado.

- **BPEL4WS:** como lenguaje es muy robusto en cuanto a la interacción de procesos, su aplicabilidad se imposibilita debido al incumplimiento del criterio *Descripción Semántica*, lo cual deriva en no cumplir a cabalidad los objetivos planteados del trabajo de grado. Sin embargo desde el punto de vista de la Red de Datos esta opción es la más adecuada, debido a la posibilidad de definir flujos de errores y excepción más flexibles y completos.

- **OWL-S:** Es la única de las tres alternativas que cumple con todos los criterios establecidos, por esta razón permite cumplir todos los objetivos del trabajo de grado, hacer un aporte significativo al estado del arte de los servicios web y además realizar una aplicación flexible y robusta como solución para la gestión de cuentas de la Universidad del Cauca.

- **WSMO:** A pesar de ser conceptualmente la mejor y más completa de las opciones, no es posible hacer uso de ella debido a que como tecnología aún es inmadura y carece de ciertos elementos, especialmente en el campo de la composición, orquestación y coreografía de servicios web.

  Situación similar ocurre con las herramientas que hacen posible la utilización de WSMO. Por otro lado, para su posible implantación se requiere de un servidor de aplicaciones, lo cual no es posible de acuerdo con el criterio *Adecuación de la Arquitectura*. Estas tres limitaciones se pueden ven reflejadas en los indicadores mostrados en la Tabla 6-1.

Por todas las razones expuestas con anterioridad el desarrollo de la aplicación de Gestión de Cuentas para la Red de Datos de la Universidad del Cauca se basa en OWL-S.

6.5 SELECCIÓN DE UNA HERRAMIENTA PARA LA OPCIÓN SELECCIONADA

La selección de una herramienta para la manipulación de OWL-S se fundamenta en no sólo poder **describir** los servicios con la ontología, sino poder **ejecutarlos** a partir de sus descripciones semánticas. La **única** herramienta que nos facilita la manipulación de la ontología en estos dos aspectos es la **API OWL-S**.

Es indispensable la ejecución de los servicios para poder implementar el prototipo de forma concreta, es decir, no sólo hacer la descripción de los servicios con base en la potencialidad teórica demostrada por la tecnología, sino demostrarla en forma práctica mediante la aplicación de la composición semántica de servicios web.

Sin embargo, esta API se encuentra en etapa de desarrollo, por lo que no soporta la manipulación de OWL-S en su totalidad. Sus principales limitaciones consisten en soportar únicamente los constructores de control *Sequence*, *Unorderer* y *Split*, lo cual deriva en limitaciones a la hora de componer los servicios.

Desafortunadamente, esto representa un factor limitante de consideración, puesto que las opciones de composición disminuyen al no poder utilizar otros constructores de control que pueden maximizar la definición de procesos complejos como lo son el If-Then-Else o RepeatUntil.

Sólo es posible realizar **composiciones sencillas de servicios**, que impliquen la ejecución ordenada (Sequence), desordenada (Unordered) o en forma concurrente (Split). No es posible definir condiciones (If-Then-Else) o crear ciclos repetitivos (RepeatUntil) en la manipulación de los procesos que hacen parte de un servicio compuesto.

Es importante anotar que **no son limitaciones de la ontología OWL-S**, sino más bien de la API que la implementa, con la salvedad que pronto será posible utilizar toda la potencialidad que define OWL-S, cuando los esfuerzos por mejorar y terminar la API culminen.

Las principales limitaciones de versión 1.0 de la API OWL-S son:

- La ontología OWL-S permite definir varios *perfiles* para un servicio, la API no cuenta con esta característica, sólo permite definir un perfil por servicio.

- No soporta la definición de entrada y salidas condicionales, al igual que precondiciones y efectos.

- Sólo permite ejecutar servicios compuestos basados en los *constructores de control Sequence*, *Unordered* y *Split*.

7. VALIDACIÓN DE LA ALTERNATIVA TECNOLÓGICA SELECCIONADA PARA LA COMPOSICIÓN DE SERVICIOS WEB

7.1 ENTORNO

El término **comunidad universitaria** hace referencia al conjunto de profesores, estudiantes, funcionarios, unidades organizacionales (facultades, departamentos, grupos de investigación, eventos, etc.) de la Universidad del Cauca. Esta última, a través de la Red de Datos, ofrece los siguientes servicios a la comunidad universitaria:

1. Correo electrónico
2. Alojamiento de páginas personales
3. Disco Virtual
4. Acceso conmutado a Internet

La infraestructura de servidores que da soporte a la prestación de estos servicios se muestra en la siguiente tabla.

SERVIDOR	RESPONSABILIDAD	CARACTERÍSTICAS
Atenea	• Servidor de Correo Electrónico • Alojamiento de Sitios Personales • Disco Virtual	• Sistema Operativo GNU/Linux Debian • Servidor Web Apache 2 • Lenguaje PHP • Correo Electrónico
Afrodita	• Servidor de Correo Electrónico • Alojamiento de Sitios Personales • Disco Virtual	
Acuario	• Servidor Web Institucional • Servidor de Base de Datos	• Sistema Operativo GNU/Linux Debian • Servidor Web Apache 2 • Servidor de BD MySQL • Cliente MySQL phpMyAdmin
Armagedon	Servidor de Acceso Remoto	Soporte a 30 usuarios simultáneos por medio de una línea RDSI.
Juno	Servidor de Directorio	Servidor de Directorio basado en OpenLDAP[27]

Tabla 7-1 Infraestructura de Servidores de la Red de Datos

[27] Más información en http://www.openldap.org

Toda persona o entidad perteneciente a la comunidad universitaria tiene derecho al uso de estos servicios y para ello, cada persona o entidad debe tener una **cuenta de usuario** con un *nombre de usuario* y una *clave* únicos. Cada cuenta de usuario consta de:

- Una cuenta de correo electrónico en *Atenea* o en *Afrodita* dependiendo del tipo de usuario.

- Una carpeta para el alojamiento de archivos y otra para el alojamiento de sitios personales, también alojados en *Afrodita* o *Atenea*.

- Un registro en el servidor de directorio y un registro en la base de datos.

La persistencia de la información de las cuentas de usuario se logra a través de estos dos últimos servidores (base de datos y directorio). Un *directorio* es un tipo especial de base de datos (análogo a un directorio convencional) que permite almacenar grandes cantidades de información con una estructura robusta y que facilita procesos como actualización y consulta. El uso de un servidor de directorio está motivado por el incremento en el número de personas que usan servicios de Internet, lo cual exige un método pertinente de almacenamiento de información [LDAP01].

7.2 APLICACIÓN DE HELP DESK ACTUAL

7.2.1 Características

La Red de Datos cuenta con una aplicación de Help Desk que tiene la responsabilidad de proveer funcionalidad de gestión de cuentas de usuario a sus funcionarios, específicamente al área de Help Desk.

Ésta aplicación está alojada en el servidor *Acuario* y está desarrollada con el lenguaje PHP. La interacción de la aplicación con el servidor de base de datos y con el servidor de directorio se logra por medio de funciones nativas del lenguaje PHP.

La interacción entre la aplicación de Help Desk y los servidores *Afrodita* y *Atenea* hace uso del servidor web presente en cada uno de ellos destinado al alojamiento de sitios personales. En cada servidor existe un *script* PHP que invocan la ejecución programas hechos en Lenguaje C, cada uno de los cuales tiene funciones específicas para la creación de una cuenta de usuario (creación de carpetas y creación de cuenta de correo electrónico) y la asignación del espacio en el disco duro para almacenamiento de archivos, correos y sitio personal. Este script PHP es invocado desde la aplicación de Help Desk (desde *Acuario*) usando el método GET de HTTP, es decir pasando parámetros a través de la URL. Estos parámetros son los usados por los programas C para sus tareas. La Figura 7-1 ilustra el despliegue de la aplicación de Help Desk sobre la infraestructura de servidores ya descrita.

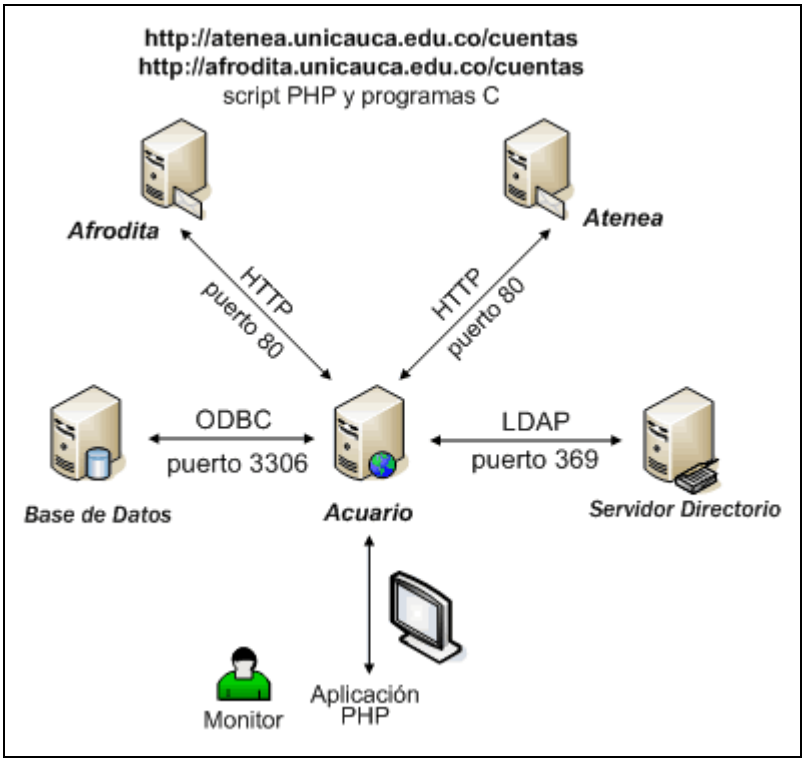

Figura 7-1 Arquitectura Funcional de la Aplicación de Help Desk

7.2.2 Problemas de la aplicación de Help Desk

La aplicación de soporte a la Gestión de Cuentas de Usuario tiene los siguientes problemas:

- El *script* PHP que reside en los servidores *Afrodita* y *Atenea* y que sirve de puente entre estos y la aplicación Help Desk carece de un nivel apropiado de seguridad, lo cual permitiría a cualquier persona tratar de ejecutarlo, anexando a la URL parámetros arbitrarios. Aunque es poco probable que alguien logre pasar variables con nombres y valores apropiados, la falla de seguridad existe.

- La lógica de la aplicación no tiene una estructura apropiada, lo cual la hace poco escalable, difícil de entender y de modificar.

- La funcionalidad general está incompleta. Actualmente se brinda soporte para la creación de cuentas, reestablecimiento de claves y actualización de permisos para el servicio de acceso remoto a Internet, pero no brinda soporte a la modificación de información de las cuentas de usuario ni a la eliminación de cuentas. Estos dos procedimientos deben ser ejecutados en forma manual y requieren de un cliente para el servidor de directorio (LDAP Browser) y un cliente para la base de datos (Cliente MySQL). Esto dificulta estos procedimientos, los hace propensos a errores y exige además que la persona que los efectúa tenga conocimiento del lenguaje SQL y de filtros LDAP.

- El procedimiento de creación de una cuenta, que involucra a tres servidores diferentes carece de flexibilidad y de robustez. Esto se evidencia porque si en el transcurso de la creación de una cuenta, alguno de los servidores no está disponible y los otros sí, la cuenta se crea parcialmente y por tanto no es funcional. Este error debe corregirse en forma manual antes de reintentar la creación de la cuenta, pues la lógica de la aplicación no contempla en ningún caso la posibilidad de ocurrencia de errores y mucho menos su tratamiento.

A manera de ejemplo, puede ocurrir que no se creen las carpetas en el servidor de correo electrónico y sí los registros en la base de datos y el directorio. En tal situación, cuando el usuario intente usar el correo, obtendrá un mensaje de error. Así, el usuario se ve obligado a acudir al área de Help Desk para buscar solución a su problema, la cual consiste en la creación manual de las mencionadas carpetas.

7.3 ANÁLISIS – ESCENARIO IDEAL

7.3.1 Descripción

A continuación, se expone un **escenario ideal** donde se puede apreciar cómo la **composición de servicios,** se puede utilizar para dar solución al Sistema de Creación de Cuentas de la Universidad del Cauca. En este escenario, se describe cómo de acuerdo con los conceptos definidos en la ontología para describir servicios OWL-S, se puede solucionar un proceso de negocio complejo que implica la interacción de varios servicios, que en principio pueden estar desarrollados en cualquier lenguaje e incluso estar distribuidos en diferentes máquinas.

Debido a la inmadurez de las herramientas que permiten implementar OWL-S, específicamente la API OWL-S, este escenario **no se puede llevar a cabo**. Como se menciono en el capitulo 6, la API en cuestión no permite ejecutar procesos compuestos cuya lógica éste basada en interacciones de procesos que impliquen condiciones (If-Then-Else) o ciclos repetitivos (RepeatUntil). Es decir, aunque se puede **describir** servicios compuestos que incluyan condiciones y ciclos repetitivos, **no es posible ejecutarlos**.

No obstante lo anterior, es importante dejar un precedente de la potencialidad que ofrece este conjunto de tecnologías, de la gran proyección a futuro con que cuentan, de lo que se podrá hacer más adelante y además, dejar claro cómo este nuevo paradigma de servicios web puede ser aplicado en un caso práctico.

El término *gestión de cuentas de usuario* abarca:

- Creación de una cuenta de usuario
- Consulta de los datos de una cuenta de usuario
- Modificación de los datos de una cuenta de usuario
- Eliminación de una cuenta de usuario
- Activación o desactivación del Servicio de Acceso Remoto

La aplicación que se podría desarrollar en este escenario, cubre las deficiencias existentes en la aplicación actual de la siguiente forma:

ASPECTO	MEJORAMIENTO
Seguridad	Tiene en cuenta criterios de seguridad que garantizan que nadie externo al personal de la red podrá usar sus funcionalidades.
Escalabilidad	Tiene en cuenta criterios de escalabilidad que faciliten la implementación de cambios.
Funcionalidad	Inclusión de las funcionalidades que la aplicación actual no presenta (ver 7.2.2)
Robustez	Incremento de la flexibilidad en la ocurrencia de fallas durante la interacción con los servidores, por medio de acciones de compensación.

Tabla 7-2 Mejoramiento logrado con la aplicación en el escenario ideal

Los servicios web utilizados se describen a continuación:

SERVICIO	RESPONSABILIDAD
Gestión LDAP	Provisión de operaciones para la interacción con el servidor de directorio para la creación, eliminación y edición de información.
Gestión Carpetas	Provisión de operaciones para la interacción con los servidores *Atenea* y *Afrodita* para la creación y eliminación de las cuentas de usuario.
Gestión BD	Provisión de operaciones para la interacción con el servidor de Base de Datos para la creación, eliminación y edición de información.
Gestión de Errores	Provisión de operaciones que son invocadas como acciones de compensación cuando ocurre algún error en alguno de los procesos o servicios asociados con los diferentes servidores. No es un servicio web independiente. Cada uno de los servicios anteriores se asocia con operaciones de gestión de errores, que pueden mostrar información acerca del error o deshacer las acciones hechas hasta el momento de la aparición de un fallo. Lo anterior debido a que, si alguno de los componentes del proceso general falla, deben deshacerse las acciones realizadas por los servicios que ya actuaron.

Tabla 7-3 Servicios Web

7.3.2 Diagrama de Orquestación

La parte más relevante de esta aplicación de Help Desk es el conjunto de procesos web descritos con OWL-S que implementan la lógica del negocio. La Figura 7-2 ilustra el **modelo ideal** del proceso de composición.

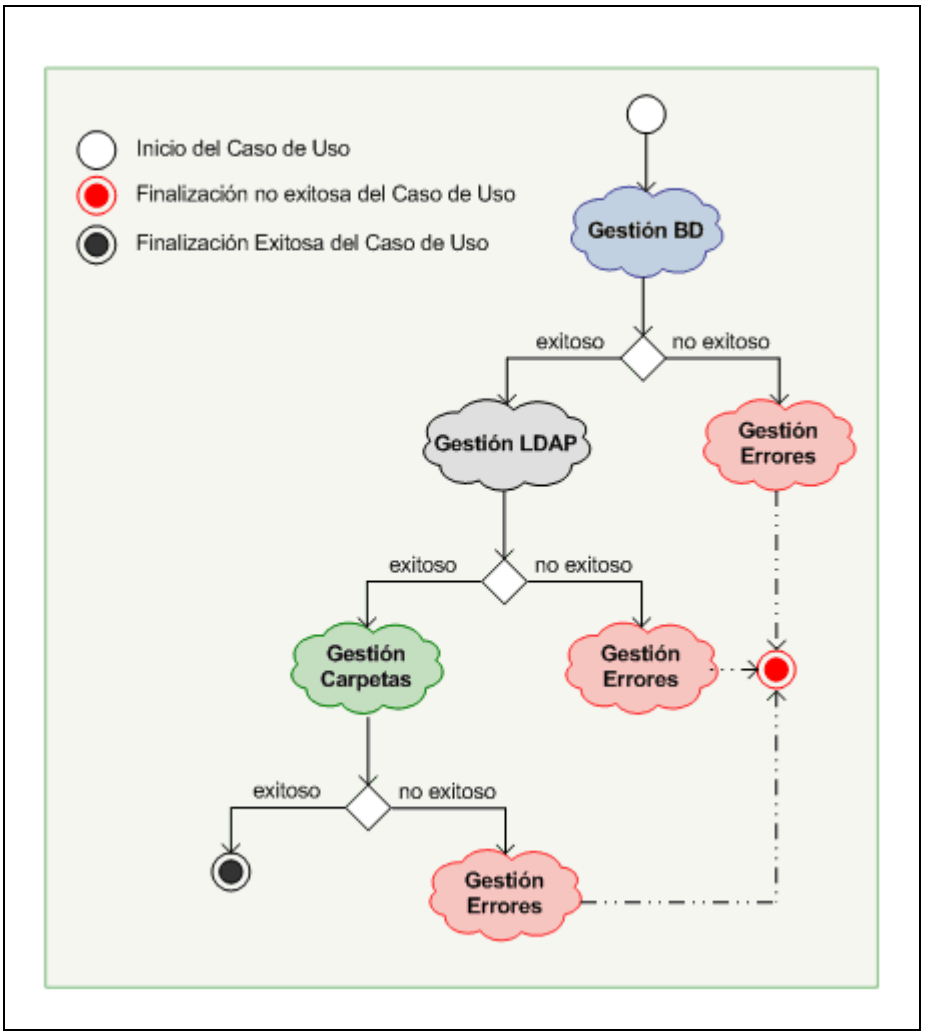

Figura 7-2 Modelo general de composición de servicios

En la anterior figura, cada nube representa un servicio y cada rombo un proceso de decisión. La secuencia de la interacción ocurre de la siguiente manera:

1. Inicio: el monitor inicializa un caso de uso y la aplicación lanza el proceso web correspondiente.

2. Invocación a Gestión BD: aquí se ejecuta la interacción con el servicio de gestión de la base de datos. Éste servicio puede insertar, consultar, modificar o eliminar un registro de la base de datos. La operación es exitosa si el comando se ejecuta en forma correcta. El mensaje de respuesta generado por el servicio indicará el resultado obtenido.

3. Decisión Gestión BD: el proceso continuará si la respuesta obtenida por el servicio anterior da cuenta del éxito en la interacción con la base de datos. En el caso contrario, el proceso invoca el servicio de gestión de errores, el cual indica al monitor la ocurrencia de una falla relacionada con la base de datos. En este caso el proceso finaliza en forma no exitosa.

4. Invocación a Gestión LDAP: aquí se ejecuta la interacción con el servicio de gestión del servidor de directorio. Éste servicio puede insertar, consultar, modificar o eliminar un registro del directorio. La operación es exitosa si el comando se ejecuta en forma correcta. El mensaje de respuesta generado por el servicio indicará el resultado obtenido.

5. Decisión Gestión LDAP: el proceso continuará si la respuesta obtenida por el servicio anterior da cuenta del éxito en la interacción con el servidor de directorio. En el caso contrario, el proceso invoca el servicio de gestión de errores, el cual es responsable de eliminar el registro creado en la base de datos. En este caso el proceso finaliza en forma no exitosa.

6. Invocación a Gestión Carpetas: aquí se ejecuta la interacción con el servidor de correo electrónico. Éste servicio puede crear o eliminar una cuenta de correo electrónico. La operación es exitosa si el comando se ejecuta en forma correcta. El mensaje de respuesta generado por el servicio indicará el resultado obtenido.

7. Decisión Gestión Carpetas: el proceso habrá concluido en forma exitosa si la respuesta obtenida por el servicio anterior da cuenta del éxito en la interacción con el servidor de correo. En el caso contrario, el proceso invoca el servicio de gestión de errores, el cual es responsable de eliminar el registro creado en la base de datos y el registro creado en el servidor de directorio. En este caso el proceso finaliza en forma no exitosa.

   Si todo el proceso se ejecuto con éxito se notifica al monitor esta situación por medio de la aplicación web a la que él accede. En caso de que ocurra un fallo, esta situación también se debe notificar al monitor por medio de la aplicación web.

7.4 ANÁLISIS – ESCENARIO REAL

7.4.1 Descripción

En este escenario se describe **lo que es posible** realizar en la actualidad, de acuerdo con el trabajo de investigación realizado. También se nombran las mejoras respecto de la aplicación con que cuenta la Red de Datos de la Universidad del Cauca.

A partir de los objetivos planteados para el presente trabajo de grado, se ha desarrollado una aplicación web prototipo de Help Desk para la gestión de las cuentas de usuario para el acceso y uso de los servicios de red ofrecidos por la Universidad del Cauca a la comunidad universitaria. Esta aplicación tiene su lógica basada en servicios web descritos y compuestos con el *Lenguaje de Ontologías Web para Servicios OWL-S*.

Por todas las razones expuestas antes, la composición se limita a una **secuencia simple**, que involucra los servicios Gestión LDAP, Gestión BD y Gestión Carpetas.

La aplicación desarrollada cubre las deficiencias existentes en la aplicación actual de la siguiente forma:

ASPECTO	MEJORAMIENTO
Seguridad	Tiene en cuenta criterios de seguridad que garantizan que nadie externo a la red podrá usar sus funcionalidades.
Escalabilidad	Fue diseñada con criterios de escalabilidad que facilitan la implementación de cambios.
Funcionalidad	Incluye las funcionalidades que la aplicación actual no presenta.

Tabla 7-4 Mejoramiento logrado con la aplicación desarrollada

Los servicios web que son compuestos en diferentes procesos se explican a continuación:

SERVICIO	RESPONSABILIDAD
Gestión LDAP	Provee operaciones para la interacción con el servidor de directorio para la creación, eliminación y edición de información.
Gestión Carpetas	Provee operaciones para la interacción con los servidores *Atenea* y *Afrodita* para la creación y eliminación de las cuentas de usuario.
Gestión BD	Provee operaciones para la interacción con el servidor de Base de Datos para la creación, eliminación y edición de información.

Tabla 7-5 Servicios Web

7.4.2 Diagrama de Orquestación

La parte más relevante de esta aplicación de Help Desk es el conjunto de procesos web descritos con OWL-S que implementan la lógica del negocio. La Figura 7-6 ilustra en el modelo general de proceso de composición utilizado. Todos los casos de uso se implementaron con base este modelo. El diagrama detallado de cada caso de uso puede encontrarse en el documento Anexo A – Modelado.

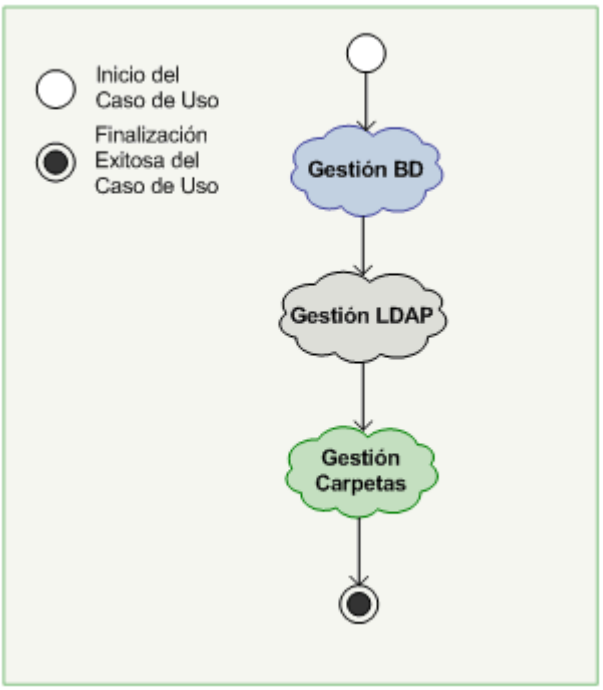

Figura 7-3 Modelo general de composición de servicios

En la anterior figura, cada nube representa un servicio. La secuencia de la interacción, descrita por el servicio compuesto y representada por el recuadro de color verde, ocurre de la siguiente manera:

1. Inicio: el monitor inicializa un caso de uso y la aplicación lanza el proceso web correspondiente.

2. Invocación a Gestión BD: aquí se ejecuta la interacción con el servicio de gestión de la base de datos. Éste servicio puede insertar, consultar, modificar o eliminar un registro de la base de datos. La operación es exitosa si el comando se ejecuta en forma correcta.

3. Invocación a Gestión LDAP: aquí se ejecuta la interacción con el servicio de gestión del servidor de directorio. Éste servicio puede insertar, consultar, modificar o eliminar un registro del directorio. La operación es exitosa si el comando se ejecuta en forma correcta.

4. Invocación a Gestión Carpetas: aquí se ejecuta la interacción con el servidor de correo electrónico. Éste servicio puede crear o eliminar una cuenta de correo electrónico. La operación es exitosa si el comando se ejecuta en forma correcta.

   Si todo el proceso se ejecuto con éxito se notificará al monitor esta situación al monitor por medio de la aplicación web a la que él accede. En caso de que ocurra un fallo, esta situación también se debe notificar al monitor por medio de la aplicación web.

7.4.3 Interacción de los Documentos de Descripción de un Servicio Compuesto

Un servicio compuesto está **descrito** por un **documento OWL-S** que indica la manera como interactúan los servicios, al igual que el **intercambio** de entradas y salidas que existen entre ellos (Flujo de Datos). Es de aclarar que un servicio compuesto **no está asociado directamente** con un **documento WSDL**.

Caso contrario ocurre con los servicios que hacen parte del servicio compuesto, quienes a partir de su propio documento OWL-S **se comunican** con el documento WSDL que permite acceder a la implementación del servicio como tal. En este tipo de servicios, también llamados procesos atómicos, se mapean las entradas y salidas del proceso con los mensajes de entrada y salida descritos en el documento de descripción WSDL.

Tal situación se puede ilustrar de manera práctica en la figura 7-4, donde el *servicio compuesto Gestión Cuentas,* ubicado en el servidor Acuario, se comunica con los servicios Gestión LDAP, Gestión Carpetas y Gestión BD, ubicados en sus respectivos servidores, a partir de los documentos OWL-S de los servicios individuales.

La manera como interactúan estos tres servicios, se describe en el servicio compuesto, para este caso es una secuencia sencilla de ejecución cuyo orden se muestra con los números 1, 2 y 3 en la siguiente figura.

Finalmente, se accede a la implementación del servicio como tal por medio del documento OWL de cada servicio individual, que a su vez se comunica con la descripción WSDL.

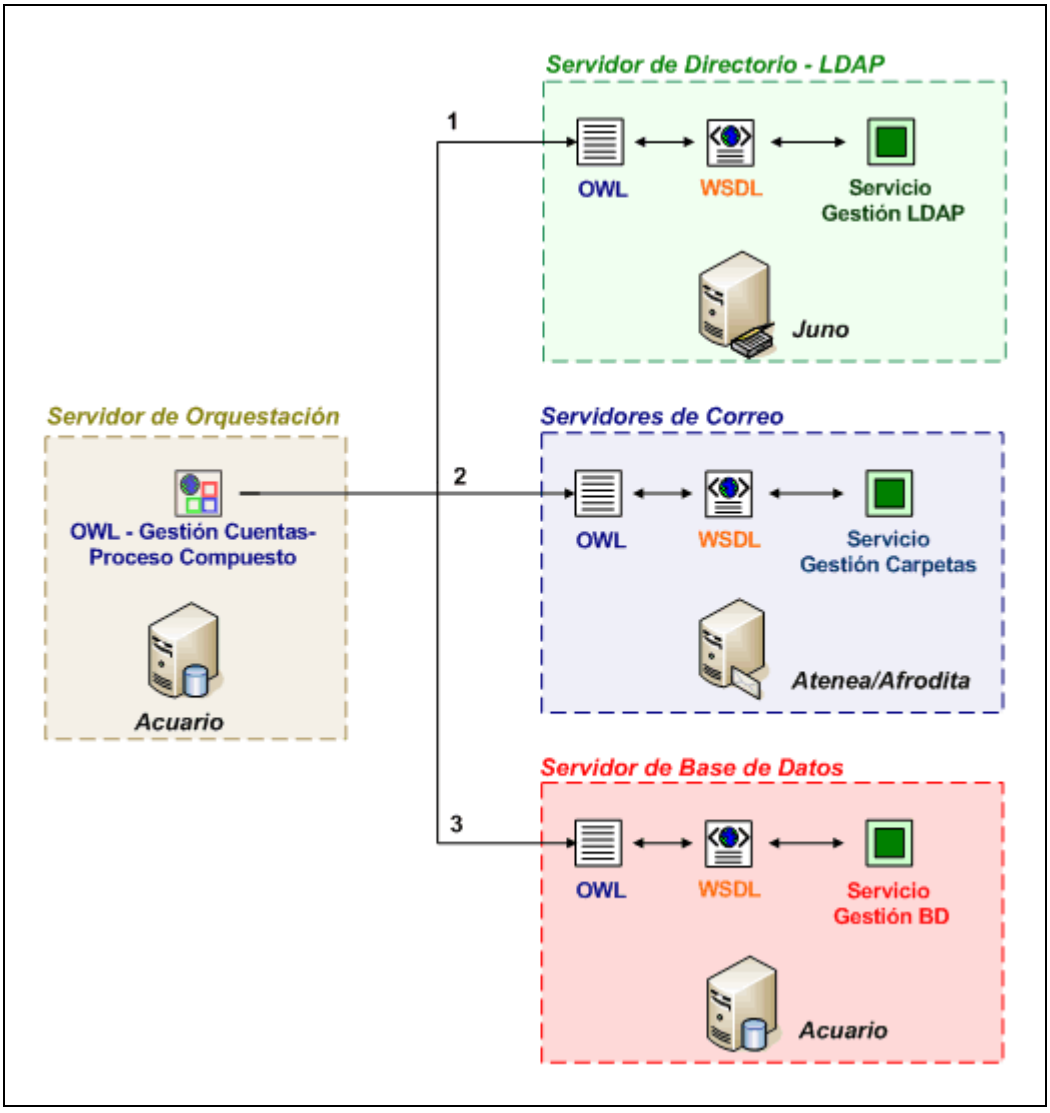

Figura 7-4 Interacción de los Documentos de Descripción de un Servicio Compuesto

En la figura anterior, cada servicio individual (Gestión LDAP, Gestión Carpetas y Gestión BD) está asociado con el servidor donde tiene su efecto. La ubicación o lugar donde residen estos servicios se discute en la siguiente sección, donde de selecciona un arquitectura para la aplicación.

7.5 ALTERNATIVAS DE ARQUITECTURA FUNCIONAL

Se contemplaron dos posibilidades para el despliegue de los servicios. La primera era publicar todos los servicios en el mismo servidor que aloja la aplicación web. La segunda era publicar cada servicio en la máquina que implementa el servicio con el que interactúa, esto es, *Gestión LDAP* en el Servidor de Directorio, *Gestión Carpetas* en *Afrodita y Atenea* y *Gestión BD* en Acuario. Debido a que el administrador de servidores de la Red de Datos sugirió que el servidor de directorio maneja una carga bastante alta y que lo mejor era no asignarle más responsabilidades, fue necesario dejar tal servicio en *Acuario*.

Las dos alternativas se explican a continuación.

7.5.1 Servicios Web Centralizados

Lo positivo de esta alternativa es que la interacción con los servidores *Atenea* y *Afrodita* se hace a través de una conexión SSH (*Secure Shell*) a través de la cuál se invocan los mismos programas C existentes. Tal conexión cuenta con autenticación basada en llaves, lo cual representa un incremento notable en la seguridad. Lo negativo es que la carga está centralizada en el mismo equipo que actúa como servidor web institucional.

Los servicios encargados de interactuar con la base de datos y el servidor de directorio, residen ambos en el servidor web de la universidad (Acuario). La figura 7-5 ilustra el despliegue de esta aplicación en la infraestructura de servidores existente.

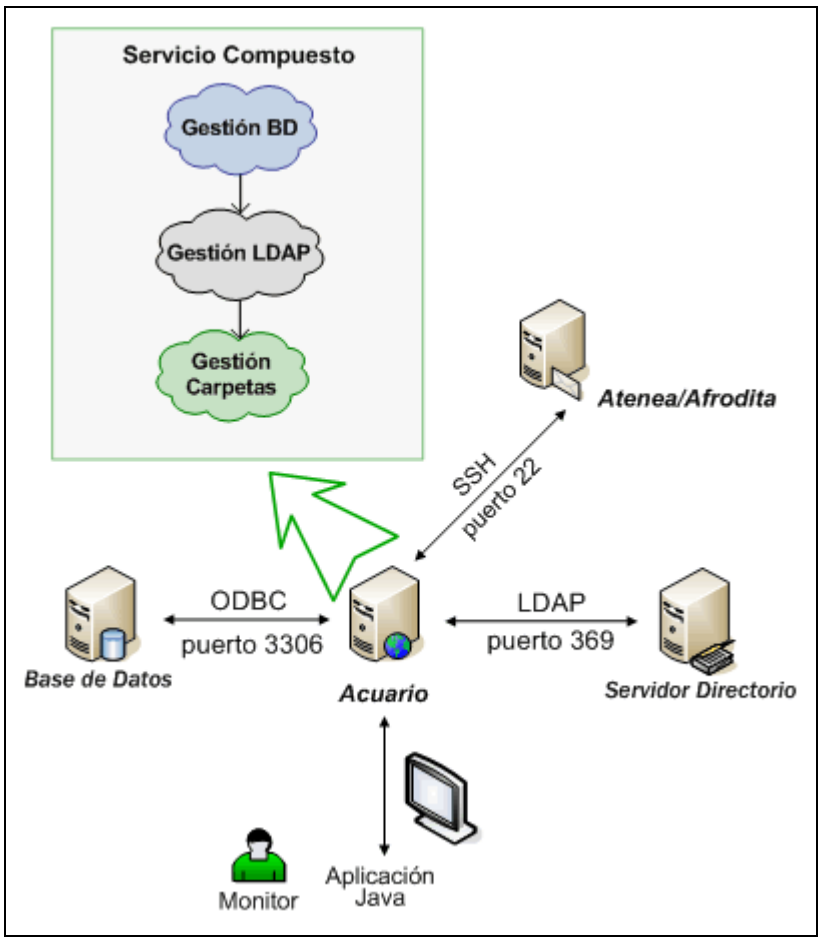

Figura 7-5 Alternativa de Servicios Web Centralizados

7.5.2 Servicios Web Distribuidos

Debido a la restricción de uso del Servidor de Directorio y a que el Servidor de Base de Datos reside en *Acuario*, esta alternativa se redujo a trasladar el servicio web de Gestión Cuentas a los servidores *Atenea* y *Afrodita* y dejar los otros dos servicios en Acuario. Por otro lado, dado que estos servidores no tienen soporte a Java y que en razón a su nivel de uso no es recomendable instalarlo, el servicio Gestión Cuenta debe ser implementado con PHP para aprovechar los servidores web disponibles.

La Figura 7-6 ilustra el despliegue de la aplicación según esta alternativa.

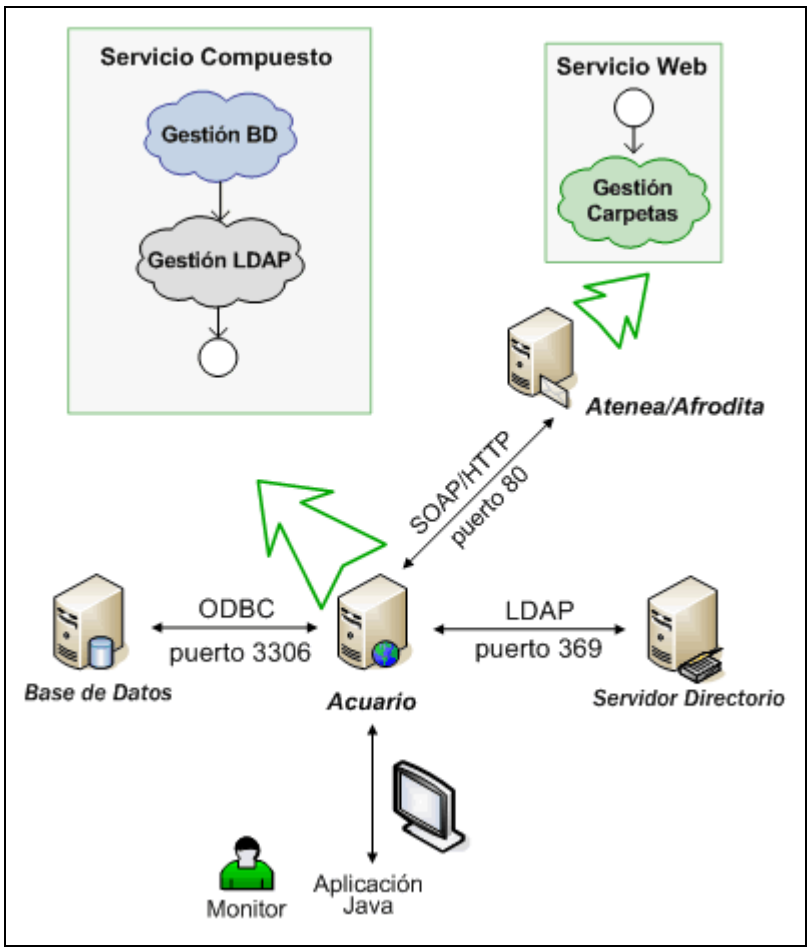

Figura 7-6 Alternativa de Servicios Web Distribuidos

Después de **analizar** las alternativas planteadas con los directores de las áreas de Desarrollo y de Servidores de la Red de Datos, se escogió la de **servicios web distribuidos**. La razón principal para la selección tiene que ver con la necesidad de no recargar el servidor *Acuario* y de aprovechar los servidores web disponibles en las máquinas Afrodita y Atenea.

La arquitectura seleccionada se **justifica** aún más desde el punto de vista académico, ya que la opción escogida se adecua mejor con la arquitectura distribuida con que cuenta la Red de Datos. Además es interesante poder tener servicios remotos hechos en PHP invocados desde un entorno Java y además **compuestos** mediante la utilización de la ontología OWL-S.

7.6 DISEÑO

7.6.1 Modelo de Casos de Uso

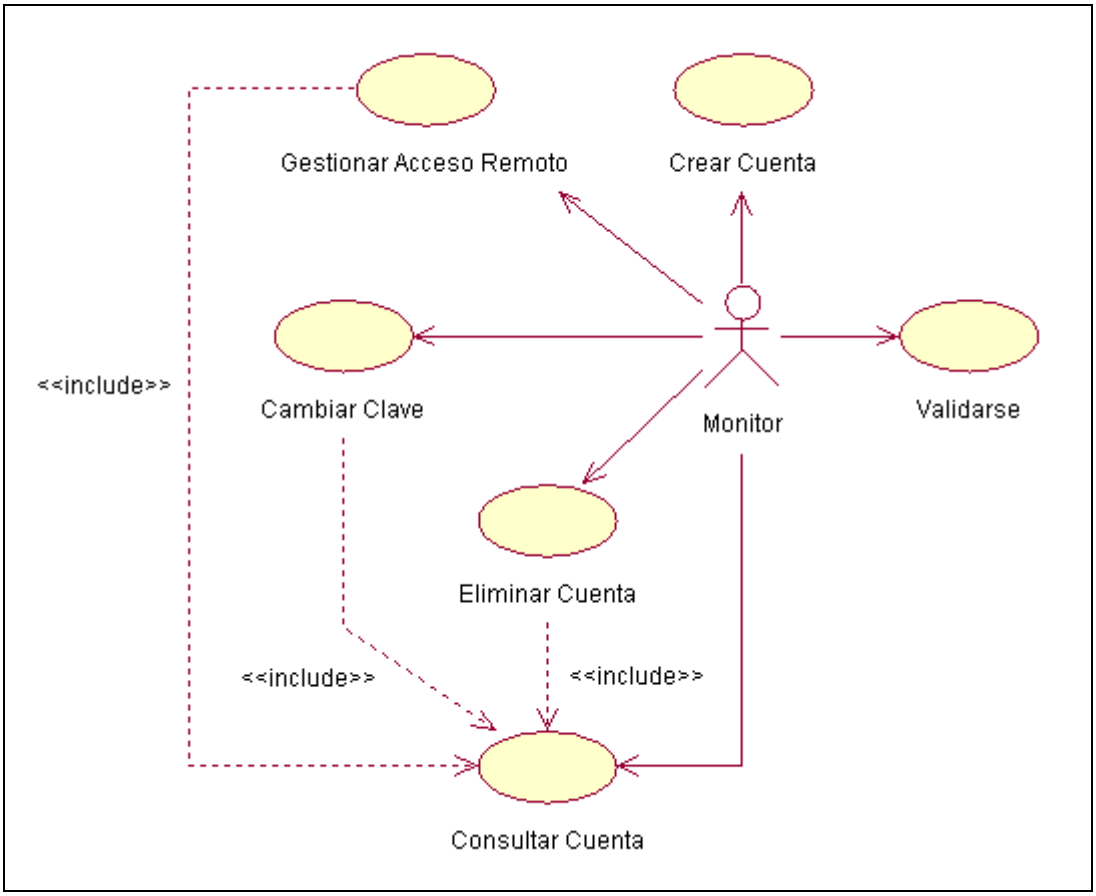

Figura 7-7 Modelo de Casos de Uso

7.6.2 Descripción de los Casos de Uso

Caso de Uso	Validarse
Iniciador	Monitor
Propósito	Restringir el derecho de uso de la aplicación a los Monitores de la Red de Datos de la Universidad del Cauca.
Resumen	En un formulario de ingreso, el monitor proporciona su nombre de usuario y su clave. Con estos datos se verifica en el servidor de directorio si la cuenta existe y en la aplicación si el monitor tiene permiso de acceso. Si el permiso se verifica se muestra la interfaz principal, de lo contrario se vuelve a mostrar el formulario

	de ingreso.

Caso de Uso	Crear Cuenta
Iniciador	Monitor
Propósito	Crear una nueva cuenta de usuario.
Resumen	En un formulario de creación de cuentas, el monitor ingresa todos los datos necesarios para crear una nueva cuenta de usuario, entre ellos un nombre de usuario que no haya sido asignado previamente. La creación de la cuenta de usuario implica lo siguiente: Creación de un registro en la tabla T_DatosUsuario de la Base de Datos. Creación de una entrada en el Servidor de Directorio. Creación de una cuenta Linux en el servidor correspondiente, dependiendo del tipo de usuario.

Caso de Uso	Consultar Cuenta
Iniciador	Monitor
Propósito	Visualizar la información relativa a una cuenta de usuario.
Resumen	En un formulario, el monitor proporciona el nombre de usuario de la cuenta que desea consultar. El sistema efectuará la búsqueda en el servidor de directorio y presentará los datos relacionados con la cuenta. Si la cuenta no existe, se mostrará un mensaje que así lo indique.

Caso de Uso	Gestionar Acceso Remoto
Iniciador	Monitor
Propósito	Modificar el estado (activo/inactivo), la fecha de vencimiento y el método de pago del Servicio de Acceso Remoto para una cuenta en particular.
Resumen	Este caso de uso depende de *Consultar Cuenta*. Cuando se están desplegando los datos de una cuenta de usuario, el monitor puede modificar las propiedades de la cuenta relacionadas con el Servicio de Acceso Remoto. Una vez efectuados los cambios, estos se almacenan.

Caso de Uso	Cambiar Clave
Iniciador	Monitor
Propósito	Cambiar la clave de una cuenta de usuario.
Resumen	Este caso de uso depende de *Consultar Cuenta*. Cuando se están desplegando los datos de una cuenta de usuario, el monitor puede permitir al usuario la creación de una nueva clave. Una vez efectuados los cambios, estos se almacenan.

Caso de Uso	Eliminar Cuenta
Iniciador	Monitor
Propósito	Eliminar una cuenta de usuario
Resumen	Este caso de uso depende de **Consultar Cuenta**. Cuando se están desplegando los datos de una cuenta de usuario, el monitor puede elegir eliminar esa cuenta en particular.

En el Anexo A - Modelado puede encontrarse la información complementaria del diseño de la aplicación.

CONCLUSIONES

1. Por medio de una revisión del origen y de la evolución de los servicios web, se comprendió el motivo que impulsa a la industria a apostar por su estandarización y masificación. Gracias a la carrera de evolución tecnológica y al afán de *integración* en todos los niveles, la aparición de esta opción de integración de aplicaciones basada en una plataforma de servicios se convierte en la alternativa que la industria esperaba para lograr la evolución y complemento de las tecnologías de computación distribuida existentes. Es acertado decir que los servicios web están en la etapa que antecede a su desarrollo pleno, lo cual hace pensar que a futuro se verán aún más ventajas y características interesantes, dignas de ser analizadas y aplicadas.

2. Para hacer posible una explotación real de los servicios web, es indispensable el desarrollo de tecnologías que enriquezcan su arquitectura. Si bien la publicación de funcionalidad es quizá el aspecto más importante, también lo es la seguridad, la calidad de servicio y la *composición*. Además, es crucial entender que la estandarización, al igual que otros campos, es un esfuerzo indispensable que facilita el logro de los objetivos planteados para cada tecnología y asegura su aceptación y expansión.

3. La iniciativa conocida como la web semántica propone nuevas características importantes para la evolución de la web. Si bien se reconoce la importancia de asociar metadatos y ontologías como nuevos elementos de la web, se debe reconocer también que el carácter abierto de Internet y la relativa dificultad encontrada para el manejo de los lenguajes, logrará que la conversión a una web de conceptos sea un proceso lento. Por otro lado, la web semántica no sólo requiere la disponibilidad de metadatos, sino también el desarrollo de aplicaciones que los usen para algo. El impulso más fuerte a la iniciativa viene desde el ámbito académico, así que la web semántica en sí misma puede convertirse en un tema de investigación dentro la Universidad del Cauca.

4. La adición de semántica explicita a los recursos de la web permitirá a las máquinas comprender e interpretar de forma similar a como lo hacen los seres humanos el significado de la información y por tanto hacer deducciones e inferencias que ayuden a la automatización de procesos.

5. Durante el proceso de investigación de las tecnologías de composición de servicios web semánticos, se encontró que una gran parte del apoyo viene de proyectos de universidades de diferentes países del mundo. Así mismo, todas las alternativas estudiadas están ejecutando o planeando el proceso de estandarización. Todo lo anterior lleva a concluir que esta área específica de los servicios web mantendrá un impulso considerable y que los avances serán continuos.

6. La posibilidad de que la web esté inundada de agentes inteligentes, con capacidad de colaborar entre sí para ejecutar tareas complejas basadas en requisitos puntuales, existe pero aún es lejana. Es posible que la iniciativa de los servicios web semánticos sea el punto de inicio que allane el camino para lograr tal nivel de automatización en la web, ya que se está abriendo la posibilidad de que estos agentes busquen y usen servicios disponibles en la web, con el mínimo de intervención humana.

7. Un aspecto critico y que aún no se ha resuelto en la composición de servicios, es el resultado que se obtiene al final de la ejecución de un proceso compuesto cuando alguno de sus componentes no está disponible. Explícitamente, lo que sucede cuando alguna de las descripciones del servicio (OWL-S o WSDL) es inalcanzable o el servidor donde se aloja la implementación del servicio referenciada desde el documento WSDL no responde. Lo anterior evidencia la necesidad de un mecanismo estándar que permita determinar la disponibilidad de un servicio web.

8. Aunque las diferentes tecnologías estudiadas en el trabajo de grado son independientes, cabe la posibilidad de maximizar su potencialidad mediante la

integración de las características más relevantes de cada una de ellas. Esto se ve reflejado en WSMO, la iniciativa más ambiciosa, la cual evalúa la posibilidad de integrar BPEL4WS o el Modelo de Proceso OWL-S como una parte funcional de su propio modelo.

9. Después de una ardua búsqueda e investigación sobre los proyectos, trabajos y documentos realizados acerca de las temáticas que abarcaban el proyecto, se concluye que a nivel nacional es prácticamente nulo el desarrollo de proyectos que involucren servicios web semánticos, orquestación, coreografía o composición de servicios. Por tal razón con este trabajo de grado se hace un aporte significativo al contexto investigativo de la Universidad del Cauca.

RECOMENDACIONES Y TRABAJOS FUTUROS

1. Seria de gran valía, orientar el presente trabajo de grado hacia la definición de una ontología formal que describa el contexto universitario y sus partes como conceptos, al igual que las relaciones que existen entre ellos.

2. Aunque de todas las tecnologías estudiadas WSMO es la más inmadura, también representa la iniciativa más ambiciosa y con mayor proyección a futuro, es por ello que es fundamental para el contexto investigativo de la Universidad del Cauca no dejar de lado esta iniciativa que muy seguramente sea la que más posibilidades tiene de convertirse en el estándar acogido por la comunidad para los servicios web semanticos.

3. Un campo de aplicación que no se abordó en el presente trabajo de grado y que puede ser motivo de estudio son los Agentes Inteligentes y los Lenguajes de Reglas Lógicas, que permitirán en un futuro realizar deducciones o inferencias acerca de conceptos y en consecuencia automatizar procesos.

GLOSARIO

BPEL4WS	Lenguaje de Ejecución de Procesos de Negocio para Servicios Web.
IOPE	Entradas, Salidas, Precondiciones y Efectos
KIF	Formato de Intercambio de Conocimiento (*Knowledge Interchange Format)*. Lenguaje usado para el intercambio de información entre sistemas de cómputo heterogéneos
LDAP	Protocolo Liviano de Acceso a Directorio (*Lightweight Directory Access Protocol*)
Ontología	Conjunto de términos, común y compartido, que describe los conceptos de un dominio, sus propiedades y las relaciones entre dichos conceptos
OWL	Lenguaje de Ontologías Web (*Ontology Web Language*)
OWL-S	Lenguaje de Ontologías de Servicios Web (*Ontology Web Language – Services*)
PDDL	Lenguaje de Definición y Planeación de Dominios (*Planning Domain Definition Language*)
RDF	Infraestructura de Descripción de Recursos (*Resource Description Framework*)
SOAP	Protocolo Simplificado de Acceso a Objetos (*Simple Object Access Protocol*). También conocido como Protocolo para Arquitecturas Orientadas al Servicio. *(Service Oriented Architecture Protocol*)
SWRL	Lenguaje de Reglas de la Web Semántica (*Semantic Web Rules Language*)
UDDI	Registro Universal de Descripción, Localización e Integración (*Universal Description, Discovery and Integration*)
W3C	World Wide Web Consortium
WSDL	Lenguaje de Descripción de Servicios Web (*Web Services Description Language*)
WSML	Lenguaje de Modelado de Servicios Web (*Web Services Modelling Language*)
WSMO	Ontología de Modelado de Servicios Web (*Web Services Modelling Ontology*)
WSMX	Entorno de Ejecución para Modelado de Servicios Web (*Web Services Execution Enviroment*).

REFERENCIAS

[WSAR04] W3C Web Services Architecture Working Group. Web Services Architecture. World Wide Web Consortium. Febrero de 2.004 http://www.w3.org/TR/ws-arch

[WSCA01] Kreger, Heather. Web Services Conceptual Architecture 1.0. IBM Software Group. 2.001 http://www-3.ibm.com/software/ solutions/webservices/pdf/wsca.pdf

[WSDL01] Web Services Description Working Group. Web Services Description Language. World Wide Web Consortium. Marzo de 2.001. http://www.w3.org/TR/wsdl

[UDDI01] Consorcio UDDI. Universal Description, Discovery and Integration. Septiembre de 2.000. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

[WWS02] Adam, Collin. Why web services. WebServices.org. http://www.mywebservices.org/index.php/article/articleview/75/1/61/

[SOMF03] XML Protocol Working Group. SOAP Version 1.2 Part 1: Messaging Framework. World Wide Web Consortium. Junio de 2.003. http://www.w3.org/TR/soap12-part1

[SOPR03] XML Protocol Working Group. SOAP Version 1.2 Part 0: SOAP Primer. World Wide Web Consortium. Junio de 2.003. http://www.w3.org/TR/soap12-part0/

[COSC02] Critical Issues of Information Systems Management. 13th Annual Survey. http://www.csc.com/survey

[WSOR03] Peltz, Chris. Web Services Orchestration: A review of emerging technologies, tools and standards. Hewlett-Packard Company. Enero de 2.003 http://devresource.hp.com/drc/technical_white_papers/WSOrch/WSOrchestration.pdf

[BPEL03] IBM Developer Works. Business Process Execution Language for Web Services Version 1.1. Mayo de 2.003. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[OWL04] Web Ontology Working Group. Ontology Web Language Guide. World Wide Web Consortium. Febrero 2004. http://www.w3.org/TR/owl-guide/

[OWLS04] DAML Services. Ontology Web Language – Service: Semantic Markup for Web Services. DAML Services. Noviembre de 2004. http://www.w3.org/TR/2004/REC-owl-guide-20040210/

[OWLS-WSDL-04] DAML Services. Describing Web Services using OWL-S and WSDL. DAML

Services. Noviembre de 2004.
http://www.daml.org/services/owl-s/1.1/owl-s-wsdl.html

[WSMO04] WSMO Working Group. WSMO Primer. WSMO Working Group. Noviembre 2004
http://www.wsmo.org/2004/d3/d3.1/v0.1/

[C-WSMO04] WSMO Working Group. Choreography in WSMO. WSMO Working Group. Noviembre 2004.
http://www.wsmo.org/2004/d14/v0.1/20041112/

[O-WSMO04] WSMO Working Group. Orchestration in WSMO. WSMO Working Group. Noviembre 2004.
http://www.wsmo.org/temp/d15/v0.1/20040529/

[WSMX04] WSMX Working Group. WSMX Architecture. WSMX Working Group. Junio 2004.
http://www.wsmo.org/2004/d13/d13.4/v0.2/20040622/

[WSML04] WSML Working Group. The WSML Family of Representation Languages. WSML Working Group. Diciembre 2004.
http://www.wsmo.org/2004/d16/d16.1/v0.2/

[LDAP01] Calzada Pradas, Rafael. Introducción al Servicio de Directorio.
http://www.rediris.es/ldap/doc/ldap-intro.pdf