UNIVERSIDAD NACIONAL DE COLOMBIA

Herramienta prototipo para generación automática de Servicios Web Semánticos a través del desarrollo de software dirigido por modelos

Wilman José Vega Castilla

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas e Industrial
Bogotá, Colombia
2013

Herramienta prototipo para generación automática de Servicios Web Semánticos a través del desarrollo de software dirigido por modelos

Wilman José Vega Castilla

Trabajo de investigación presentado como requisito parcial para optar al título de:
Magister en Ingeniería de Sistemas y Computación

Director:
Msc. Henry Roberto Umaña Acosta

Línea de Investigación:
Ingeniería de Software

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas y Computación
Bogotá, Colombia
2013

A Dios por la fuerza, sabiduría y paciencia.
A mis padres y hermanos, mi inspiración para salir adelante.

Agradecimientos

Agradezco a la Universidad Nacional de Colombia, en especial a la Facultad de Ingeniería por brindarme la oportunidad de formarme en este programa de postgrado.

Al profesor Henry Roberto Umaña Acosta, director de la presente tesis, por su oportuna tutoría, y por sabiduría, dedicación y paciencia que tuvo en este proyecto.

A Danier y Deivis, por sus consejos y ánimos para el desarrollo del trabajo.

Al Ingeniero John Cruz, por sus aportes en los aspectos técnicos del presente trabajo.

Resumen

Los Servicios Web Semánticos ofrecen beneficios que coadyuvan a la evolución de la Web.

Beneficios como el descubrimiento, invocación y composición dinámica y automática de recursos, habilitan efectivamente la interoperabilidad entre sistemas, permitiendo una amplia gama de nuevos servicios y oportunidades de negocios en la Internet. La estructura necesaria para proveer estos beneficios por parte de los Servicios Web Semánticos hace que su desarrollo sea un proceso complejo. Es necesario establecer formas más fáciles y dinámicas de desarrollar este tipo de software, que garanticen reutilización, calidad y rapidez en el proceso de desarrollo.

El desarrollo dirigido por modelos realiza una contribución eficiente en estos aspectos, dado que trabaja de manera intrínseca conceptos relacionados con estos aspectos: Separación de conceptos, reusabilidad e interoperabilidad entre componentes. En el presente trabajo se presenta un enfoque para desarrollo de software dirigido por modelos, orientado al desarrollo de los servicios web semánticos, generando las especificaciones más utilizadas para este tipo de aplicaciones.

Palabras Claves: **Servicios web semánticos, Desarrollo dirigido por modelos, ontologías web.**

Abstract

Semantic Web Services offers benefits that contribute to Web evolution. Benefits such as automatic discovery and invocation, and dynamic composition, effectively enables systems interoperability, allowing a wide range of services and Internet businesses. The necessary structure to provide those benefits by Semantic Web Services makes its development a complex process. It is necessary to establish more easy and dynamic ways to develop this kind of software, in order to assure reuse, quality and speediness in the development process. The model-driven software development makes an efficient contribution in those aspects because it works intrinsically concepts related such separation of concerns, reusability and components interoperability. In this document we present an approach to model-driven development software applied to Semantic Web Services, generating the most used specifications for this type of applications.

Keywords: **Semantic Web Services, Model-Driven Development, WebOntology**.

Contenido

Lista de figuras

Lista de tablas

Lista de Símbolos y abreviaturas

Abreviaturas

Abreviatura	Término
CIM	Computer Independent Model
DSL	Domain Specific Language
HTTP	Hypertext Transfer Protocol
IOPE	Inputs Outputs Preconditions and Effects
MDA	Model-Driven Architecture
MDD	Model-Driven Development
MOF	Meta Object Facility
OMG	Object Management Group
OWL	Ontology Web Language
OWL-S	Ontology Web Language for Semantic Web Services
PIM	Platform Independent Model
PSM	Platform Specific Model
SOAP	Simple Object Access Protocol
UML	Unified Modeling Language
WebML	Web Modeling Language
WSML	Web Services Modeling Language
WSMO	Web Service Modeling Ontology
XML	Extensible Markup Language

1. Introducción

Los Servicios Web se han caracterizado por ofrecer una solución a manera de arquitectura para módulos de software empresarial, con el objetivo de facilitar la integración entre diferentes sistemas. Sin embargo, el descubrimiento y la integración de nuevos servicios no son automáticos y requiere intervención humana [**1**]

La automatización de la invocación, la composición dinámica, el descubrimiento automático, entre otros, son capacidades que requieren los Servicios Web para mejorar la interoperabilidad entre sistemas que éstos presentan. Es aquí donde los servicios web semánticos entran en juego para proporcionar estas ventajas.

Los Servicios Web Semánticos son una tecnología clave que tiene mucha incidencia actualmente y será mucho más relevante en el futuro [**25**]. Por esto es necesario revisar cuidadosamente, los procesos de desarrollo y las herramientas que se implementan para los servicios web semánticos, de qué manera y con qué tecnologías se pueden beneficiar estos procesos. Dentro de estos procesos de desarrollo, el Desarrollo Dirigido por Modelos (Model-Driven Development – MDD) tiene una importante aceptación gracias a las ventajas que ofrece para este tipo de productos. [**19**] [**2**]

El Desarrollo Dirigido por Modelos es una forma de desarrollo en donde la abstracción, a través de modelos se establecen de forma central en todo el proceso, en contraste con otras formas o métodos de desarrollo donde los modelos son usados de manera intencional y no formal, para cumplir un papel en una parte del proceso de desarrollo, pero no a través de todo el proceso. Esto facilita, por ejemplo, la creación de descripciones semánticas de los servicios web, mientras se ocultan los detalles sintácticos asociados a sus especificaciones.

Existen estrategias, métodos, y herramientas que apoyan el proceso de desarrollo de servicios web semánticos. Se han descrito los principales enfoques utilizados en la industria del software, pero estos enfoques se basan mayormente en aproximaciones al estándar de la Arquitectura Dirigida por Modelos –*Model-Driven Architecture*, MDA, con las limitaciones que presentan. [**5**] [**24**]

Dentro de la literatura revisada, se muestran modelos y enfoques para el desarrollo de servicios web semánticos, pero muy pocas presentan herramientas que evalúen dichos enfoques. También se puede observar que algunas herramientas que se desarrollan, son para soportar una parte del proceso de construcción de estos enfoques [**1**] [**12**], que usan herramientas de transformación UML para los diagramas y perfiles UML que usan en sus propuestas. Sólo algunos [**5**]proveen herramientas para generar de manera automática aplicaciones de servicios web semánticos, aplicando enfoques de desarrollo dirigido por modelos diferente a lo que presenta MDA.

Lo anterior nos lleva a realizar el siguiente interrogante, el cuál es el que se pretende solucionar: ¿Cómo se puede desarrollar una herramienta para el desarrollo de servicios web semánticos aplicando estrategias de desarrollo de software dirigido por modelos?

El trabajo que se propone en este documento, presenta una novedad con respecto a las propuestas existentes basadas en metodologías de software: El diseño de una herramienta prototipo para realizar transformaciones automáticas de especificaciones de Servicios Web Semánticos OWL-S y WSMO. Muchas de los enfoques aquí presentados muestran herramientas que se perfilan por una de estas especificaciones y desarrollan un nivel de especialización sobre éstas que resultan ventajoso. Sin embargo, es importante destacar que la implementación de las dos especificaciones provee un mayor rango de aplicaciones y componentes que pueden interconectarse y enlazarse de manera más fácil si comparten este tipo de estructuras [**20**].

También es importante resaltar las diferentes aplicaciones que se pueden direccionar a través del desarrollo de esta propuesta. Inicialmente, se puede observar el planteamiento de una comparación de los modelos (no sólo la clasificación). También, la actualización de métodos y enfoques de desarrollo a través del trabajo planteado, con las herramientas y metodologías actualizadas que aquí se utilizan.

1.1 OBJETIVOS

1.1.1 Objetivo general

Desarrollar una herramienta para generación automática de especificaciones OWL-S y WSMO para Servicios Web Semánticos.

1.1.2 Objetivos específicos

El desarrollo de este trabajo, requiere el alcance de los siguientes objetivos.

1. Analizar los enfoques MDD para el desarrollo de Servicios Web Semánticos que apliquen dentro de sus modelos las especificaciones OWL-S y WSMO.
2. Diseñar los componentes de la herramienta propuesta mediante enfoque MDD y utilizando la herramienta XText y un Lenguaje de dominio específico - DSL.
3. Evaluar la aplicación de la herramienta propuesta a través de una prueba de concepto.

1.2 ESQUEMA DEL DOCUMENTO

Para afrontar el trabajo propuesto, se han establecido tres fases, que derivan de los objetivos definidos en este trabajo. Estas fases se establecen en los capítulos de este documento y se describen a continuación:

En el capítulo 2: **Desarrollo dirigido por modelos para los Servicios Web Semánticos**, se realiza una revisión exhaustiva de la literatura, con el fin de identificar de manera precisa, los elementos introductorios para el desarrollo de la herramienta propuesta. Aquí se proporciona los elementos claves del dominio del tema en la propuesta de investigación. Esta fase proporciona como producto principal, la implementación de referencia [**31**] para dichas especificaciones.

En el capítulo 3: **Diseño y desarrollo de la herramienta prototipo de desarrollo de Servicios Web Semánticos** se plantea el esquema MDD a aplicar, detallando los elementos y herramientas que se van definiendo en esta etapa. Esto es, mediante la implementación de referencia obtenida en la etapa anterior, se realiza un análisis para identificar, dentro de éste los elementos de código que serán modelados y cuáles pasarán de manera directa, es decir, identificar los elementos de código esquemático repetitivo, individual y genérico. Una vez identificados los elementos de código a modelar, con el código esquemático repetitivo se define el meta modelo del dominio de la aplicación, que establece las entidades del dominio, los atributos y las relaciones entre estas entidades. En este punto, se diseña el DSL para la aplicación del servicio web semántico.

Utilizando la herramienta Xtext, se realiza la implementación del DSL y luego se realiza la validación respectiva. Finalmente se crean las plantillas para la generación de código para las dos especificaciones seleccionadas.

Por último, en el capítulo 4: **Evaluación de la herramienta prototipo de desarrollo de servicio web semántico,** luego de realizar el diseño de la herramienta, es preciso realizar una demostración de ésta. Para esto se toma un caso práctico, preferiblemente, un desarrollo que sea requerido. Luego de implementada la herramienta se realiza una evaluación de su aplicación. Para esta evaluación es necesario definir previamente, los criterios de evaluación y los factores determinantes de éxito con respecto a los enfoques presentados.

2. Desarrollo dirigido por modelos para servicios web semánticos

Actualmente, el desarrollo de los servicios web se fundamenta en el uso de un lenguaje de descripción de servicios web (WSDL), que especifica el acceso a los servicios [**7**]. Este lenguaje de descripción no cuenta con las capacidades necesarias para habilitar las ventajas que ofrecen los servicios web semánticos (búsqueda, descubrimiento, selección, composición e integración, automáticas). Para alcanzar estas ventajas, es preciso asociar al lenguaje un mecanismo de descripción semántica. [**28**]

Hay diferentes maneras de asociar descripción semántica a la descripción de servicios web, tales como, WSDL-S, OWL-S y WSMO. Entre estas, las dos últimas emergen como las principales especificaciones para definir servicios web semánticos [**19**] [**29**] [**16**] [**20**].

Pero esta extensibilidad de los servicios web a servicios web semánticos tiene su desventaja. La complejidad estas especificaciones, especialmente en su gramática, hacen que el desarrollo para servicios web semánticos se dificulte. Metodologías de desarrollo se han adaptado para poder sobrellevar esta desventaja [**1**]. El desarrollo dirigido por modelos facilita la creación de estos modelos complejos, a su vez que permite a los desarrolladores de software comunicarse de una manera más eficiente para realizar el proceso de validación, diseño y desarrollo de servicios web semánticos [**22**].

En este capítulo se presentan los elementos introductorios del marco teórico del proyecto. Dividido en tres secciones, en la primera se exponen los conceptos de servicio web semántico y los elementos que los componen, se muestran detalladamente las especificaciones que se usaran como referencia del proyecto: OWL-S y WSMO. En la segunda sección se presentan los conceptos del desarrollo dirigido por modelos, y por último se enseñan los principales enfoques de desarrollo de servicios web semánticos con MDD.

2.1 Servicios Web Semánticos

En [**23**] se define un servicio web de la siguiente manera:
“Un servicio web es un componente de software modular bien definido que expone una interface sobre la red. El uso de los servicios web se hace a través del envió y recibo de mensajes XML a través del protocolo HTTP”.

Este sistema modular bien definido proporciona las bases para el intercambio de información entre organizaciones de manera uniforme y modular, permitiendo realizar procesos de interoperabilidad entre aplicaciones de forma fácil [**25**].

El objetivo del servicio web es el de soportar una infraestructura libre para servicios web "inteligentes". Así como los servicios web deben tener una descripción específica que determine su definición, para qué está hecho el servicio web y cuáles servicios se puedan invocar.

Los servicios web semánticos se definen como una extensión a los servicios web. Éstos, proporcionan ventajas como el descubrimiento automático de servicios, la invocación automática del servicio, y la composición e interoperabilidad del servicio web [**29**].

La infraestructura definida para los servicios web semánticos debe estar definida en un mecanismo similar a la infraestructura de servicios web [**29**]. Para esto, los servicios web semánticos definen sus IOPE’s (*Input, Output, Preconditions and Effects*). Las entradas (*Inputs*) definen que tipos de datos requieren los servicios, la salida (*Output*) describe lo que será entregado por el servicio web. Las precondiciones (*Preconditions*) establecen registros del estado que debe mantener para poder ofrecer el servicio. En los efectos (*Effects*) se consideran los elementos que cambian de estado (verdadero) si el servicio es ejecutado satisfactoriamente [**16**].

2.2 Ontologías Web

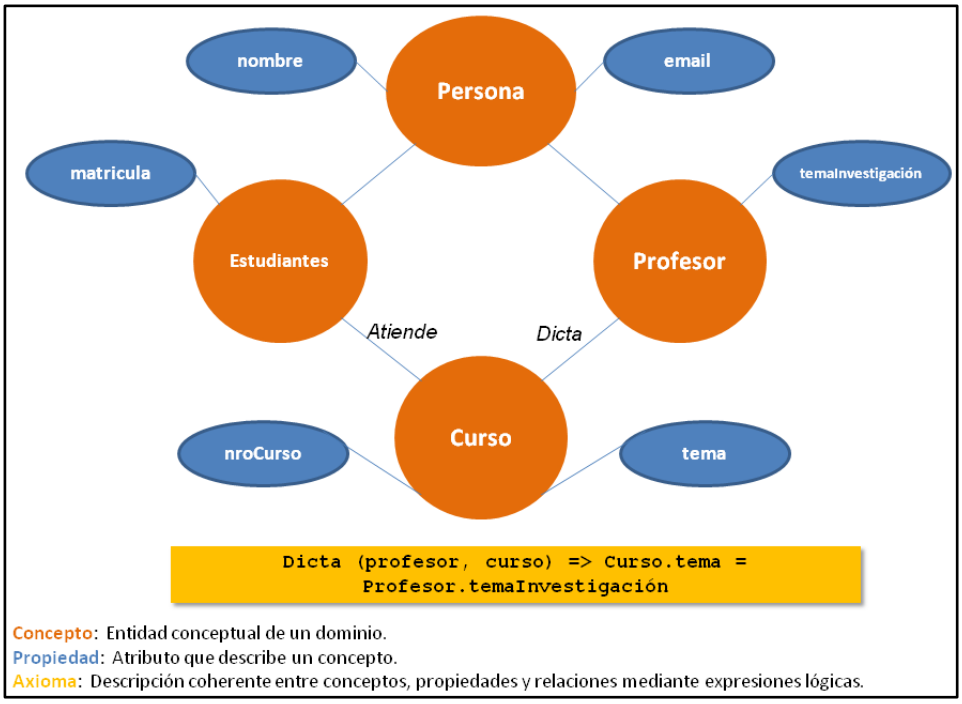
El objetivo de la web semántica es datos que sean entendidos por máquinas, y que sean compartidos en red. Para esto, la web semántica es apoyada por metadatos, que a su vez son conceptualizados y descritos por ontologías que le dan el significado para que exista tal entendimiento [**12**].

Esa conceptualización, es la base fundamental de un cuerpo formal de conocimiento, junto con los objetos y conceptos y relaciones dentro de este cuerpo formal. Una

conceptualización es un punto de vista abstracto y simplificado, con el cual se quiere representar algo para un propósito específico [**14**].

Según [**14**] una ontología es una especificación explicita de una conceptualización. Ontología, es un término tomado de la filosofía, comparte con el área de ciencias de la computación, la significación de representación de ideas y conceptos dentro de un dominio especifico, y las relaciones entre estas ideas y conceptos. En la siguiente figura, se muestra un ejemplo sencillo de ontología con sus elementos. Una de las representaciones más útiles de las ontologías, son los modelos.

Figura 2-1. Ejemplo de ontología con sus elementos



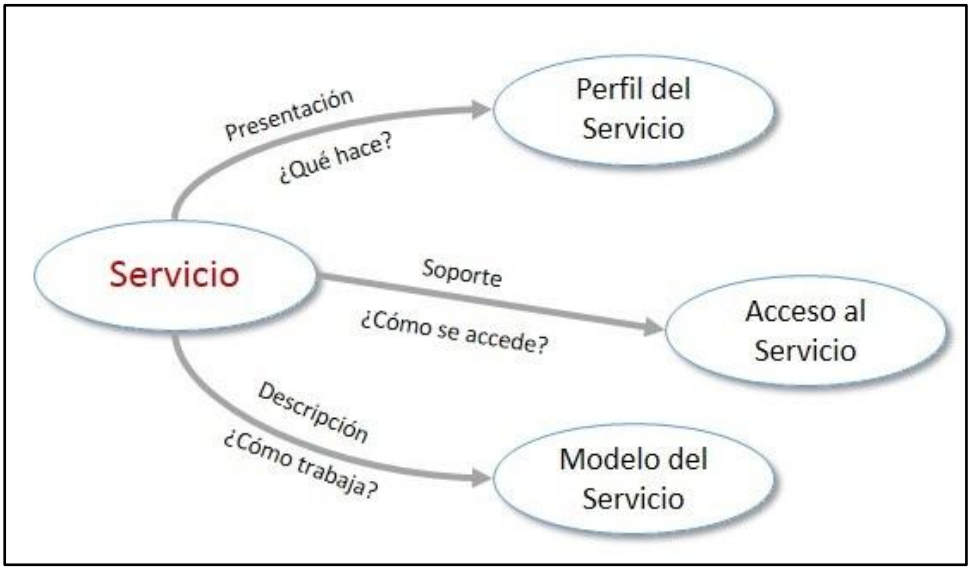
2.3 Lenguaje de Ontología Web para la Servicios Web Semánticos(OWL-S)

OWL-S se estructura bajo un meta modelo que describe los servicios, motivados por tres tipos de conocimientos acerca del servicio. Cada conocimiento es caracterizado por una pregunta, que se responde inmediatamente [**10**]. En la figura 2-2se refleja esta estructura.

La principal tecnología subyacente de OWL-S es el Lenguaje de Ontología Web (Ontology Web Language – OWL), que es un lenguaje de marcado para publicar y compartir datos en la Web, fundamentalmente ontologías, y desarrollado como extensión del Marco de Descripción de Recursos (Resource Description Framework – RDF).

A continuación se detallan los elementos principales de la especificación OWL-S.

Figura 2-2. Ontología de Servicio en OWL-S

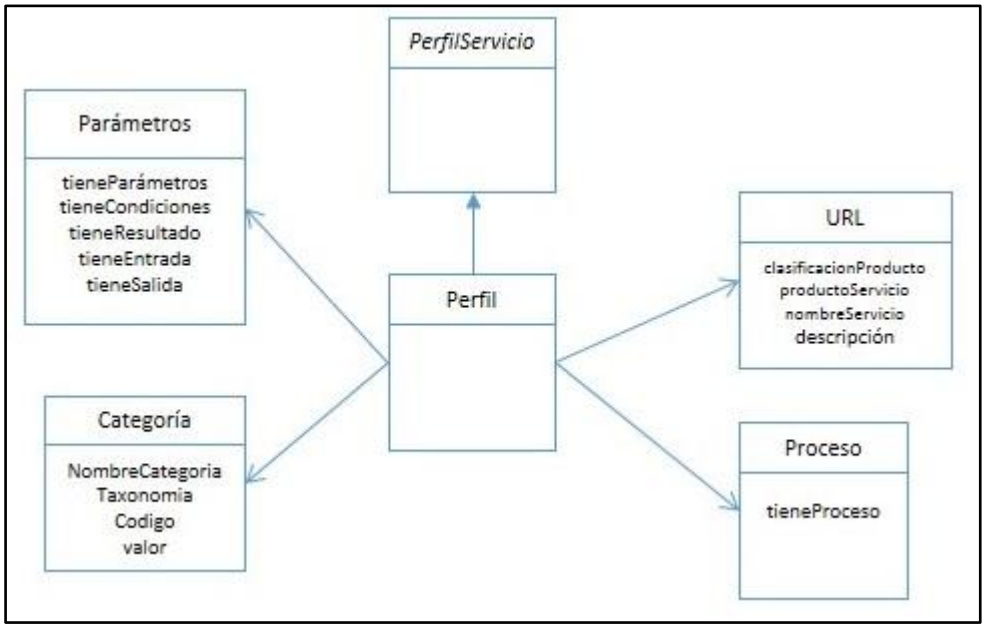


- Perfiles de Servicio.

Una transacción en un servicio web involucra tres partes: El que solicita el servicio, el proveedor del servicio, y la infraestructura subyacente. Un perfil de servicios en OWL-S, describe un servicio que es ofrecido a los solicitantes, por los proveedores de servicio, tomando como base esta infraestructura subyacente. [**6**]

Un perfil de servicio en OWL-S describe un servicio en base a tres partes básicas de información: la información del proveedor, la función que cumple el servicio y un conjunto de atributos que especifican las características del servicio. Utilizando la notación de clases y subclases dentro de OWL-S, se puede esquematizar un perfil de servicio como se muestra en la siguiente figura:

Figura 2-3. Esquematización de un perfil de servicios OWL-S



- Modelo de servicio

El modelo de servicios en OWL-S en comparado con un modelo de proceso. De manera específica, en OWL-S se establecen las subclases *ServiceModel* y *Process* para realizar un modelo de servicios. El modelo de servicio representa la manera o las maneras cómo el solicitante de un servicio puede acceder a éste, esta representación tiene más que ver con la forma de operación, que con el acceso como tal, que es definido por otra característica de los servicios [**6**].

En OWL-S hay dos tipos de procesos, procesos atómicos, que describen un servicio que espera un mensaje y retorna otro mensaje en respuesta. Y un proceso compuesto, que describe un servicio que mantiene un estado, cada vez que el solicitante envía un mensaje, se avanza a través del proceso.

El modelo de servicios en OWL-S especifica de manera detallada la infraestructura IOPE, de los modelos de servicios web semánticos, pero con algunos detalles dentro de su composición. [**6**]

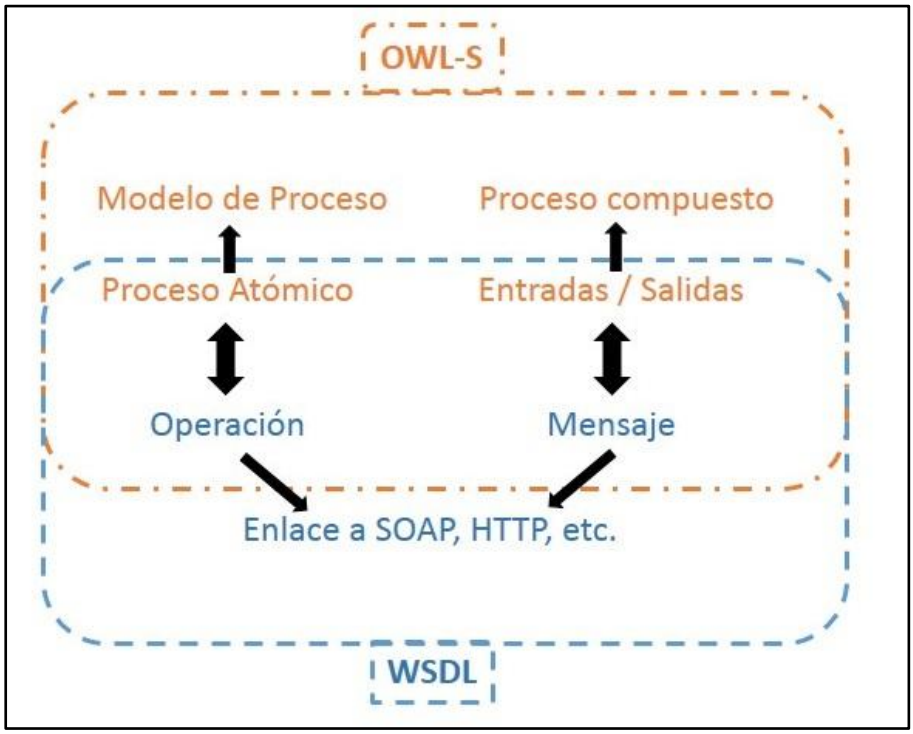
- Acceso a servicios

El acceso a los servicios dentro de OWL-S, establece en detalle la forma cómo se acceden a éstos, desde el punto de vista del protocolo, los formatos de los mensajes, serialización, transporte y direccionamiento. Teniendo en cuenta esto, el acceso a servicios es representado de manera concreta y no abstracta, como el perfil de servicios

y el modelo de servicios, ya que maneja elementos concretos de acceso relacionados con plataformas y otros elementos de un nivel de abstracción más bajo [**6**].

El acceso a servicios dentro de OWL-S, puede ser descrito a través de WSDL, permitiendo la complementariedad entre los dos lenguajes. Dicha complementariedad se la siguiente figura:

Figura 2-4. Relación entre WSDL y OWL-S mediante el acceso a servicios



Para mayor detalle sobre la especificación (en modelos de clases) de las características de servicio web semántico, según OWL-S véase el anexo A, tomado de [**6**]

2.4 Ontología de Modelado para Servicios Web (WSMO)

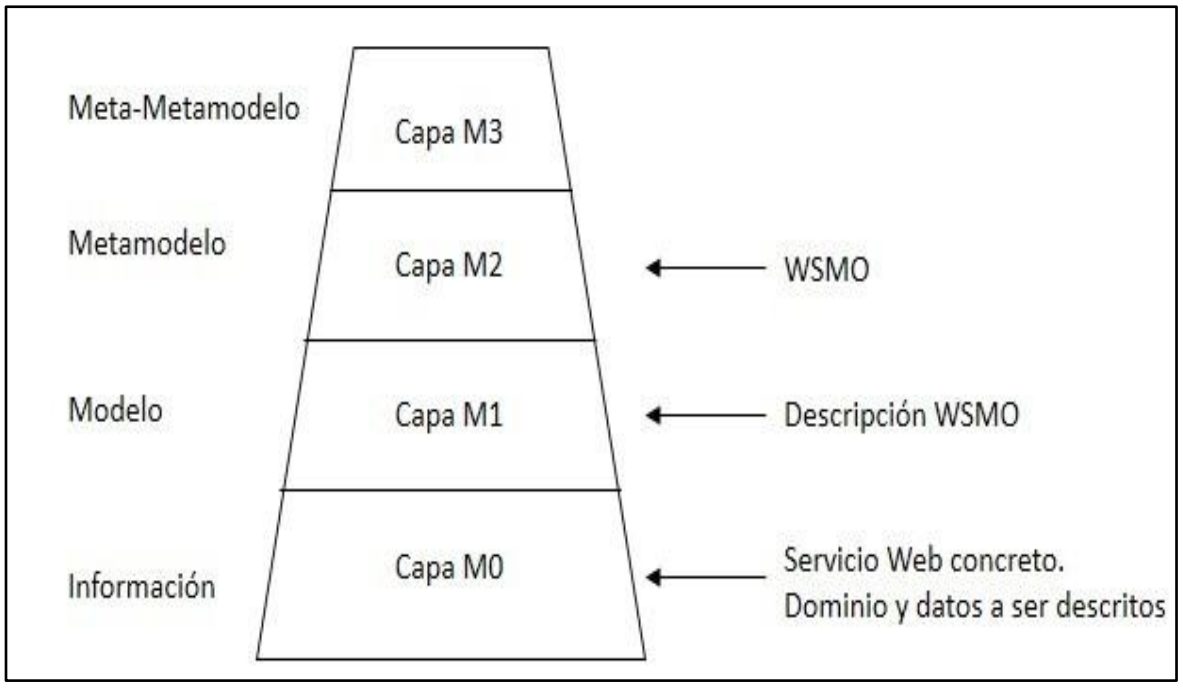
WSMO se basa en cuatro elementos básicos de infraestructura, que se conocen como elementos de modelado del lenguaje. Estos elementos son las *ontologías*, que proporcionan la semántica para la descripción de los elementos en WSMO, las *metas*, establecen los medios para especificar los objetivos del solicitante del servicio, describiendo en un nivel superior, una tarea concreta a ser ejecutada. Los servicios web, proveen la semántica para la descripción de servicios web, incluyendo las propiedades funcionales y no funcionales de estos servicios, entre otros aspectos, y por último, los

mediadores, especifican las conexiones que resuelven problemas de heterogeneidad para habilitar operaciones entre partes heterogéneas. [**8**]

Además de estos elementos, WSMO establece un conjunto de propiedades no funcionales que pueden ser usados por todos los elementos principales de WSMO, entre estas propiedades están el conjunto de elemento de metadatos *Dublín Core*, por ejemplo.

WSMO, al igual que OWL-S puede ser modelado mediante un esquema de clases y subclases, de hecho, hay una relación cercana entre una representación de modelos y meta modelos (específicamente, *Model Object Facility*, MOF) y WSMO [**32**], lo que permite abarcar de manera más fácil el desarrollo de servicios web semánticos basados en WSMO a través de MDD. En la siguiente figura se muestra esa relación:

Figura 2-5. Relación entre MOF y WSMO



En la siguiente tabla se describen los elementos que componen la especificación WSMO:

Tabla 2-1. Descripción de los elementos de WSMO

Elemento	Descripción	Subelemento
Ontología	Se describe en meta-nivel. Presenta todos los aspectos de la ontología que proporciona la terminología a los otros elementos de WSMO.	Propiedades no funcionales, Ontologías importadas
Metas	Definen los objetivos que un cliente puede tener cuando consulta un servicio.	Propiedades no funcionales, Ontologías importadas, mediadores usados, pos condiciones y efectos
Servicios Web	Establece diferentes aspectos de los servicios web dentro de WSMO	Propiedades no funcionales, mediadores *ooMediators,* capacidades e Interfaces
Mediadores	WSMO introduce el concepto de mediadores para resolver problemas de heterogeneidad. Los mediadores son elementos especiales que enlazan componentes heterogéneos involucrados en un modelo WSMO. Hay cuatro tipos de mediadores, dependiendo de los elementos que enlazan.	• *ggMediators*: Enlazan dos metas. • ooMediators: Enlaza ontologías (ontologías importadas) • *wgMediators*: Enlaza un servicio Web a una meta. • *wgMediators*: enlaza dos servicios web.

En el anexo B. se encuentran en detalle, los elementos pertenecientes a WSMO en un diagrama de clases UML [**8**].

2.5 Comparación entre OWL-S y WSMO.

Teniendo las especificaciones de servicios web semánticos definidas para el desarrollo dirigido por modelos, es preciso realizar una comparación entre estás, a modo de establecer elementos comunes que servirán para el enfoque de desarrollo MDD [**31**] que se plantea en este trabajo y que se describirá en el siguiente capítulo.

Dado que ambas especificaciones representan la manera de describir servicios web semánticos, y son las especificaciones de facto para esta descripción [**20**] [**11**], es normal que presenten semejanzas y diferencias en la forma como realizan dicha descripción.

Inicialmente, sus elementos pueden ser representados por modelos, lo cual facilita el desarrollo bajo los enfoques de MDD. También se puede destacar que ambas especificaciones permiten la misma infraestructura subyacente acerca de la forma de acceso y presentación de los servicios.

En la siguiente tabla, se presentan los principales aspectos que son comunes entre las dos especificaciones. Considerando como punto de partida los elementos de OWL-S [**20**].

Tabla 2-2. Aspectos comunes entre OWL-S y WSMO.

Elementos OWL-S	Correspondencia en WSMO	Observaciones
Perfil de servicios	Metas	Un Perfil de servicios trata sobre la presentación (quien solicita, quien provee, infraestructura subyacente) de un servicio, en WSMO esta responsabilidad recae sobre las metas y la capacidad del servicio web.
	Capacidad de Servicio Web	
Modelo de servicios	Capacidad de Servicio Web	El modelo de servicios muestra la manera cómo opera el servicio, en WSMO, esto representa la funcionalidad, la cual es definida por la capacidad del servicio web, y los modos de acceso, que son definidos por la coreografía WSMO
	Servicio Web, coreografía WSMO	
Acceso a Servicios	Coreografía WSMO	El acceso a servicios proporciona detalles sobre cómo se acceden a los servicios en OWL-S. Este aspecto en WSMO aún no está definido.[1]

Cabe destacar que la comparación entre estas especificaciones para realizar un componente que pueda generar elementos comunes entre ellos tiene limitaciones [**26**]. Entre estas, los elementos generados se limitan a la descripción de elementos no funcionales para los servicios web. Esto, debido precisamente a la limitación que tienen estas especificaciones para describir elementos funcionales. En estudio, se encuentran alternativas a las especificaciones, y variantes de estas para poder superar estas limitaciones [**26**].

2.6 Desarrollo dirigido por modelos

El modelado ha estado presente, de una u otra forma, en el desarrollo de software desde hace un buen tiempo. El Desarrollo Dirigido por Modelos (Model-Driven Development – MDD) es una forma de desarrollo en donde la abstracción a través de modelos se establecen de forma central en todo el proceso de desarrollo, en contraste con otras formas o métodos de desarrollo donde los modelos son usados de manera intencional y no formal, para cumplir un papel en una parte del proceso de desarrollo, pero no a través de todo el proceso. Es por esto que el nombre de la disciplina (“Model-Driven”) se diferencia de otras formas de uso de modelado en el software (“Model-Based”).

[1] Si bien no está definido el aspecto de acceso a servicios en WSMO, probablemente hará parte de la definición de coreografía de WSMO[13]

Un modelo es un “conjunto coherente de elementos que describen algo (un sistema, un banco, un tren, un teléfono, etc.)” [**22**]. Dentro del desarrollo dirigido por modelo, dicho modelo debe ser expresado en un lenguaje definido en un nivel de abstracción determinado. El desarrollo dirigido por modelos tiene como objetivo determinar estas abstracciones de dominio específico y hacerlas accesibles a través de un modelado formal. Este procedimiento establece un gran avance hacia la automatización de la producción de software.

Es precisamente a través de esta importante característica que es la automatización de producción de software, que el desarrollo dirigido por modelos promete grandes ventajas sobre el desarrollo de software convencional. Algunas de estas ventajas (y metas a cumplir del desarrollo dirigido por modelos) son:

- El desarrollo dirigido por modelos permite incrementar la velocidad de desarrollo.
- El uso de transformaciones automáticas y modelos definidos formalmente permite mejorar la calidad del software desarrollado.
- A través de la abstracción se puede optimizar la capacidad de gestión y complejidad del software.
- El desarrollo dirigido por modelos ofrece un entorno productivo en todas las fases de desarrollo de software, mediante la proliferación de herramientas que permiten implementar las mejoras prácticas en el desarrollo.

El desarrollo dirigido por modelos se usa ampliamente y su popularidad va en aumento. Se afirma que puede ser la continuación natural de la programación como se conoce actualmente. [**31**]

2.6.1 Arquitectura dirigida por modelos (Model-Driven Architecture – MDA)

La MDA es una tecnología estándar especificada por la Object Management Group, oficialmente adoptada en el 2001. Las metas fundamentales del MDA son la interoperabilidad, la separación de conceptos, la portabilidad y la reusabilidad. Cabe resaltar que el MDA es un subconjunto del desarrollo de software dirigido por modelos, que está limitado en técnicas de modelado basadas principalmente en el Lenguaje de Modelado Unificado - *Unified Modeling Language*, UML.

La aproximación a MDA ofrece abiertamente un conjunto de conceptos y herramientas que realizan lo siguiente:

Especifican un sistema independiente de la plataforma que la soporta. Luego, se establecen plataformas, se selecciona una en particular para el sistema y se realizan transformaciones de la especificación del sistema de otra plataforma. [**22**]

- *Meta Object Facility* (MOF)

A continuación se expresa la definición oficial del estándar MOF, establecida por el OMG: El MOF, provee un repositorio que puede ser usado para especificar y manipular modelos, y de esta manera, fomentar la consistencia en la gestión de modelos a través de todas las fases de uso del MDA [**23**].

- Modelo Independiente de La Computación (*Computer Independent Model*, CIM)

El Modelo CIM, es un modelo de un sistema que no presenta detalles de plataformas específicas ni elementos computacionales. Generalmente los modelos CIM son denominados modelos de dominio o modelos de negocio. [**23**]

Los modelos CIM ayudan a obtener un mejor panorama del problema que se está afrontando. Comúnmente el vocabulario para la generación de conceptos con otros modelos se genera de aquí.

- Modelo Independiente de La Plataforma (*Platform Independent Model*, PIM)

Un modelo PIM, puede modelar el sistema, incluir elementos que pertenecen al dominio empresarial y el dominio computacional, pero aún es independiente de la plataforma de implementación.

De manera más precisa, un modelo PIM describe el sistema con un alto nivel de abstracción sobre una plataforma tecnología [**18**].

- Modelo Dependiente de La Plataforma (*Platform Specific Model*, PSM)

El modelo de la plataforma, o modelo especifico de la plataforma, es el espacio final de implementación del sistema. Es donde se ve el sistema en ejecución. Los modelos de implementación tienen una estrecha relación la tecnología de implementación, es por esto que pueden ser generados fácilmente de manera automática.

2.7 Enfoques de desarrollo dirigido por modelos aplicado a los servicios web semánticos

En [**17**] se sintetizan los enfoques de MDD aplicados al desarrollo de servicios web semánticos bajo tres categorías: los basados en principios y metodologías de Ingeniería de Software, que proveen soluciones metodológicas y aplican buenas prácticas durante

la aproximación al MDA para la generación de especificaciones semánticas y la composición los servicios web; los basados en la aplicación formal de UML, que establecen los diagramas UML para la descripción de los elementos semánticos; y los que se basan en meta modelos, que disponen de modelos y meta modelos independientes de las especificación semánticas.

Cabe destacar que dentro de esta clasificación, los enfoques están basados principalmente en MDA. Esto tiene su asidero en que, como se apuntó anteriormente, el MDA es un estándar en la industria del desarrollo, establecido por la OMG. En la figura 2-6, se puede apreciar las categorías descritas anteriormente:

Figura 2-6. Clasificación de los enfoques MDA para desarrollo servicios web semánticos

Una forma más específica de esta caracterización, comparando las propiedades escogidas por [**17**], se muestra en la tabla2-3, tabla 2-4 y tabla 2-5.

En la tabla 2-4, se clasifican los enfoques teniendo en cuenta los lenguajes utilizados para la parte conceptual de las propuestas, en términos del modelo propiamente dicho. Dentro de la clasificación realizada por [**17**], los enfoques presentados por [**16**], [**19**], [**30**], utilizan lenguajes y elementos propios del UML, que es base principal del estándar MDA, y por eso son categorizados como enfoques basados en UML Formal. También utilizan lenguajes UML, los trabajos presentados por [**1**], [**13**] [**21**], [**3**]. Los trabajos presentados por [**2**] y [**5**] se enfocan más en procesos de negocio, por lo cual utilizan lenguajes para modelar sus componentes que no tiene base en el UML, como lo son BPMN para [**2**] y el lenguaje definido por [**5**]: WebML.

Tabla 2-3. Comparación de enfoques de desarrollo dirigido por modelos para servicios web semánticos: Categoría lenguajes de modelado

ENFOQUES	LENGUAJES DE MODELADO								
	UML						NO – UML		
	CD	AD	SQD	STD	UP	Condición	BPEL4W	BPMN	WebML
BASADOS EN UML FORMAL									
Yang & Chung. [**16**]	X			X		GUI			
Timm & Gannod. [**29**]	X	X			X	OCL			
Kim & Lee. [**19**]	X	X	X		X	Constraint			
BASADOS EN METODOLOGÍAS DE SOFTWARE									
Torres et al. [**30**]	X	X		X	X	OWL-S			
MIDAS-S. [**1**]	X				X	OCL			
Brambila et al. [**5**]								X	X
BASADOS EN META MODELOS									
Gronmo et al. [**13**]	X	X			X	OCL			
Lautebacher & Bauer. [**21**]	X	X			X	Constraint			
Bensaber & Malki. [**3**]	X	X			X	OCL			
Belouadha et al. [**2**]	X				X			X	

CD = Diagrama de clases, AD = Diagrama de Actividad, SQD = Diagrama de secuencia, STD = Diagrama de estado, UP = Perfil UML

Otra característica de clasificación utilizada por [**17**] son los lenguajes de transformación entre modelos que utilizan los enfoques. En esta característica, se destacan las propuestas de [**2**], [**30**] y [**21**], que utilizan lenguajes que están más relacionados con el desarrollo dirigido por modelos (MDD), como lo son lenguajes de transformación modelo a texto (M2T) y la herramienta Xpand. El resto de propuestas utiliza propuestas derivadas de perfiles UML y conversiones de perfiles UML a texto, como lo son Java o XSLT, por ejemplo. Esta clasificación se puede observar en la tabla 2-4.

Finalmente, en la tabla 2-5, se clasifican las propuestas teniendo en cuenta los estándares de servicios web semánticos que generan. Se destacan las propuestas de [**13**] y [**21**], que generan código para múltiples estándares de servicio web.

Tabla 2-4. Comparación de enfoques de desarrollo dirigido por modelos para servicios web semánticos: Categoría lenguajes de transformación

ENFOQUES	LENGUAJES DE TRANSFORMACIÓN				
	XPand	Java	XSLT	M2T	ATL
BASADOS EN UML FORMAL					
Yang & Chung. [**16**]			X		
Timm & Gannod. [**29**]			X		
Kim & Lee. [**19**]			X		

ENFOQUES	LENGUAJES DE TRANSFORMACIÓN				
	XPand	Java	XSLT	M2T	ATL
BASADOS EN METODOLOGÍAS DE SOFTWARE					
Torres et al. [**30**]				X	
MIDAS-S. [**1**]					
Brambila et al. [**5**]			X		
BASADOS EN META MODELOS					
Gronmo et al. [**13**]		X	X		
Lautebacher & Bauer. [**21**]	X				
Bensaber & Malki. [**3**]			X		
Belouadha et al. [**2**]				X	
XSLT = Extensible Style sheet Language Transformation, M2T = Model to Text, ATL = Atlas Transformation Language					

Tabla 2-5. Comparación de enfoques de desarrollo dirigido por modelos para servicios web semánticos: Categoría estándares de servicios web semánticos

ENFOQUES	ESTÁNDARES DE SERVICIOS WEB SEMÁNTICOS				
	OWL-S	WSMO	SWSO	WSDL-S	SAWSDL
BASADOS EN UML FORMAL					
Yang & Chung. [**16**]	X				
Timm & Gannod. [**29**]	X				
Kim & Lee. [**19**]	X				
BASADOS EN METODOLOGÍAS DE SOFTWARE					
Torres et al. [**30**]	X				
MIDAS-S. [**1**]		X			
Brambila et al. [**5**]		X			
BASADOS EN META MODELOS					
Gronmo et al. [**13**]	X	X			
Lautebacher & Bauer. [**21**]	X	X	X	X	
Bensaber & Malki. [**3**]	X				
Belouadha et al. [**2**]					X
XSTL = Extensible Style sheet Language Transformation, M2T = Model to Text, ATL = Atlas Transformation Language.					

Dentro de los enfoques clasificados, aquellos que se basan en metodologías de software buscan abordar fases más allá de la construcción de los servicios web semánticos o sus componentes. Estos enfoques proponen su implementación incluyendo las fases del ciclo de vida del desarrollo de software. MIDAS-S [**1**], por ejemplo [**21**], en una de las dimensiones que estructuran su propuesta, establece de manera ortogonal el desarrollo de las fases del ciclo de vida del software, comprendiendo desde el análisis (Modelado conceptual), hasta el modelado físico y la implementación. [**5**], con y [**30**], hacen lo propio

con sus propuestas, comprendiendo las etapas de desarrollo, desde el diseño hasta la implementación.

Estos enfoques proporcionan una ventaja, al permitir el desarrollo de proyectos dentro de un marco mucho más completo, involucrando las fases de desarrollo de software. En el campo de los servicios web semánticos, la reusabilidad de Servicios y de proyectos de desarrollo de servicios web semánticos, se destaca como otra de las ventajas. [**5**]

Es pertinente destacar los enfoques que generan código para varias especificaciones de servicios web semánticos, ya que coinciden con la característica principal que presenta la propuesta de este documento. Entre los enfoques MDD para el desarrollo de varias especificaciones al tiempo, se pueden destacar a [**21**] y [**13**].

2.7.1.Innovator Suite[2]

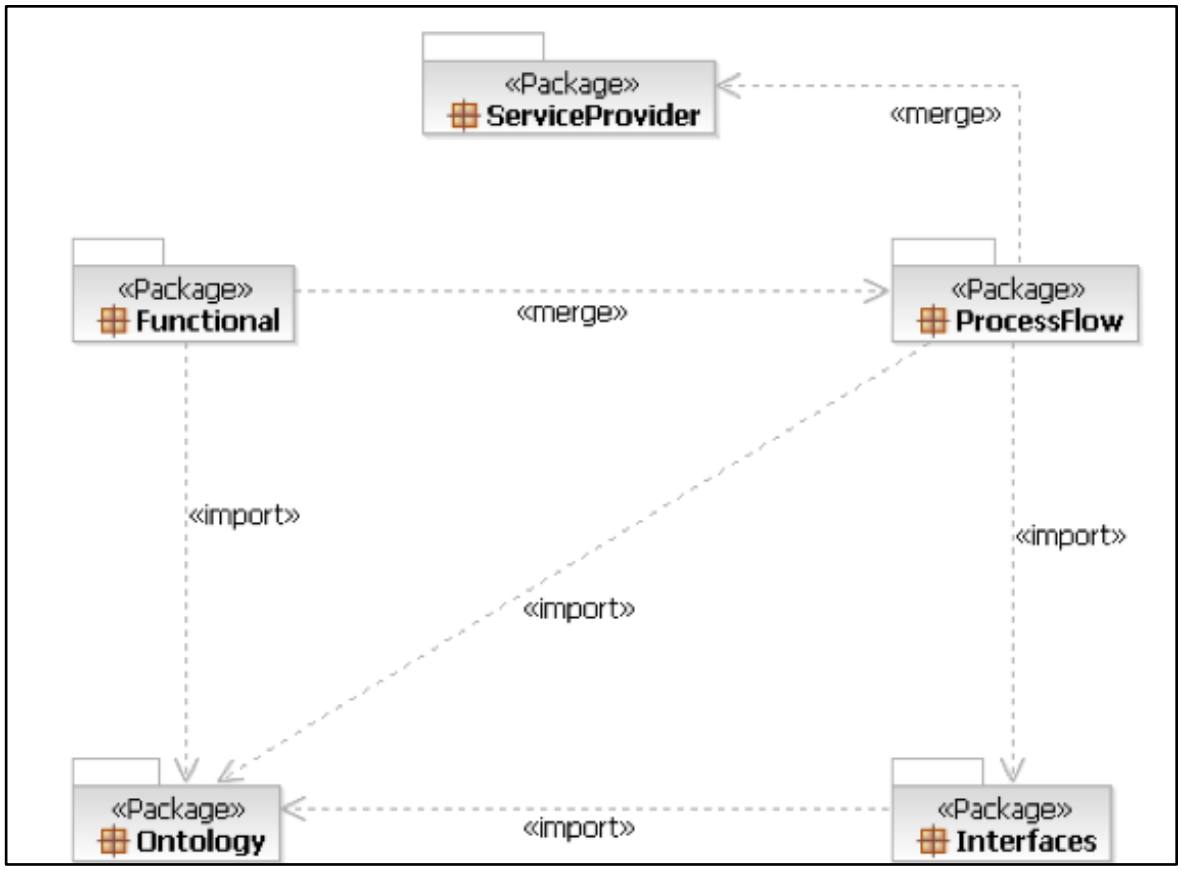
En la propuesta de [**21**] se genera código para las especificaciones de servicios web semánticos OWL-S, WSMO, SWSO y WSDL-S. Sin duda, una propuesta bastante completa para la descripción de servicios web semánticos. A través de un meta modelo para poder determinar los modelos de los estándares de descripción de servicios web. Para el diseño del meta modelo utiliza perfiles UML, y para su desarrollo e implementación utiliza la herramienta Xpand para las transformaciones (principales mente Modelo a texto – M2T). El enfoque derivó en una herramienta denominada innovatorAOX, distribuida de manera comercial como Innovator Suite por la empresa MID GMBH.

En el trabajo de [**21**] el meta modelo definido establece los lineamientos para la construcción de servicios web semánticos en cinco estructuras o paquetes. Un paquete denominado ontología, que define los elementos de la ontología para la parte semántica del servicio. Esta parte se fundamenta en un meta modelo de definición de ontologías, desarrollado por la OMG, para que pueda ser interoperable en las especificaciones generadas. Un paquete de interfaces, que contiene los elementos para la descripción del servicio web (WSDL, entre otros). Un paquete de proveedor de servicios, que proporciona los elementos no funcionales del servicio web semánticos. Los últimos dos paquetes, flujo de proceso y el proveedor de servicios extienden a los paquetes de funcionalidad y el paquete de ontología, respectivamente, para realizar la compatibilidad entre las especificaciones a generar.

[2]http://www.mid.de/en/products/innovator-for-business-analysts.html

En la figura 2-7 se puede apreciar el diseño del meta modelo con sus paquetes y relaciones:

Figura 2-7, Vista de paquete del meta modelo propuesto por [21**]**



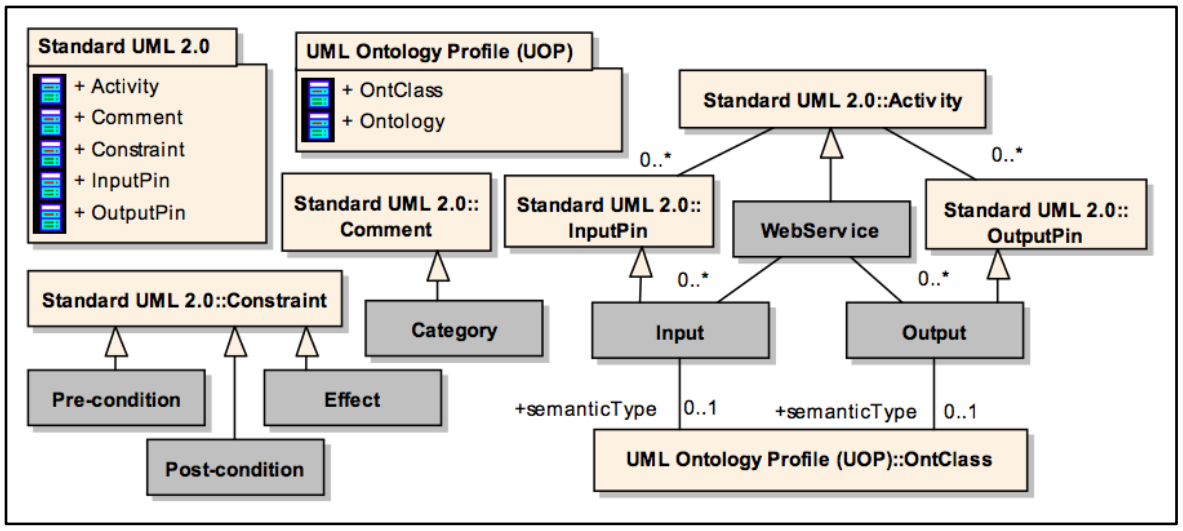
En el anexo C, se aprecian con más detalle la estructura interna de cada uno de los paquetes propuestos por [**21**].

2.7.2. Herramienta de transformación entre OWL-S y WSMO

La herramienta de transformación entre las dos especificaciones motivo de estudio de este trabajo, los realiza también [**13**]con su propuesta basado en perfiles UML para modelar los aspectos semánticos de los servicios web, y una combinación de lenguajes, entre ellos Java y XSLT, para la construcción del algoritmo que se apoya en el perfil UML para realizar las transformaciones.

El perfil UML modela los principales componentes que describen el servicio web, los elementos que lo componen o modelan (Entradas, salidas, pre y pos condiciones, categoría, etc.,) y otras extensiones de UML que definen el componente ontológico. En la figura 2-8 se muestra el perfil UML para la propuesta de [**13**].

Figura 2-8. Perfil UML para la descripción de servicios web semánticos propuesta por [13]



En el anexo D. Se puede revisar en detalle cómo se hace la relación entre la especificación OWL-S y el perfil UML propuesto por [**13**].

3. Diseño y desarrollo de la herramienta prototipo para el desarrollo de servicios web semánticos

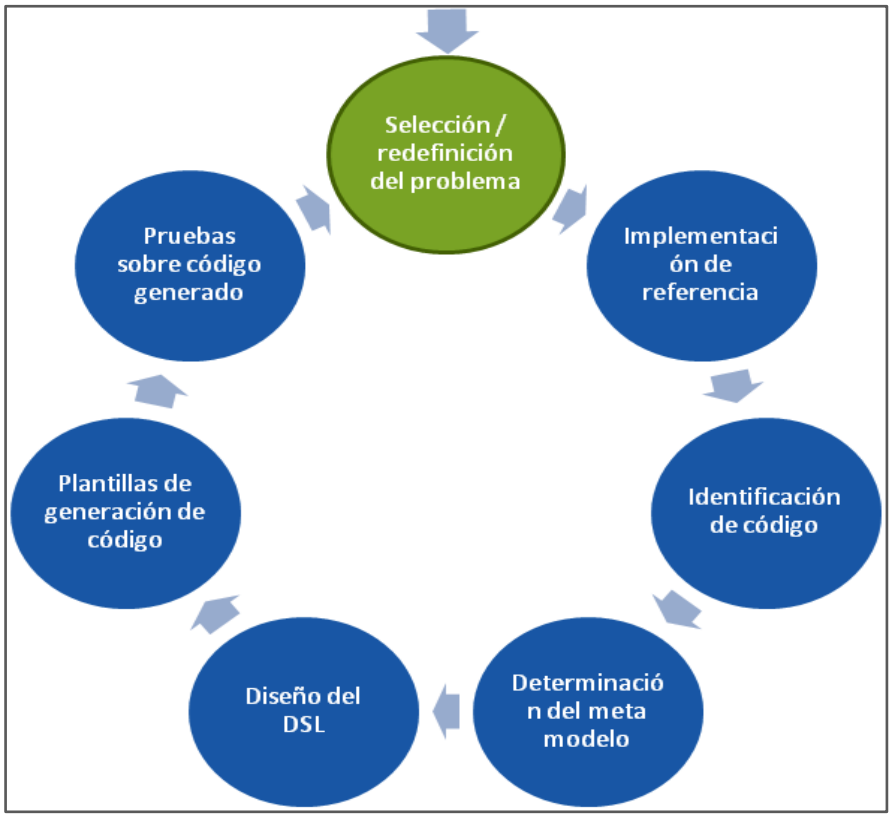
Este capítulo se centra en el diseño de la herramienta prototipo para la generación de servicios web semánticos. Para esto, es importante determinar la estrategia a seguir, en términos de metodología propiamente, a fin de poder asegurar que el producto final sea el esperado.

Una metodología relacionada al desarrollo dirigido por modelos es afrontar el desarrollo desde el dominio del problema de forma manual e individualizada, utilizando el modelado de software para el desarrollo de procesos o como documentación, y luego se trata de contrarrestar los problemas de sincronización que se presenten. Otra metodología es el desarrollo de 'ida y vuelta' o ingeniería inversa, utilizado por muchas herramientas de UML, que permite la generación de código a partir de diagramas UML. [**31**]

La metodología a implementar se enfoca en el uso de modelos dentro del dominio que se observa [**31**]. Realmente, el enfoque que se hace del dominio en el que se trabaja, es común dentro de las metodologías que se manejan, pero el uso de modelos garantiza la reusabilidad de los componentes que se van desarrollando. Los modelos dentro del desarrollo dirigido por modelos representan una abstracción en el sentido del software que se desea desarrollar, comparable a su código fuente. Luego, los modelos no serían usados simplemente como documentación, sino como parte del software como tal, lo que constituye una ventaja en términos de asegurar las metas del desarrollo dirigido por modelos.

En la siguiente gráfica se esquematiza la metodología que se utilizara para el diseño y desarrollo de la herramienta.

Figura 3-1. Metodología del desarrollo dirigido por modelos



3.1 Selección del problema

En esta sección se determinan los elementos básicos del dominio del problema que queremos afrontar, en términos estrictos de la metodología. El dominio elegido para esta fase del trabajo es el de la especificación de servicios web, con los aspectos semánticos proporcionados por los estándares en el área: OWL-S y WSMO. A continuación se detallan las tecnologías que se utilizan en el dominio del problema y las herramientas utilizadas en la aplicación de la metodología de desarrollo dirigido por modelos que se va a implementar.

3.2 Tecnologías y Herramientas

Como se indico anteriormente, el dominio del problema radica en producir aplicaciones de servicios web con componentes semánticos proporcionados por las especificaciones OWL-S y WSMO. Sin embargo, es necesario presentar, además de estos estándares, las tecnologías y herramientas que están involucradas, tanto para la representación del dominio del problema (identificado con mayor claridad en la implementación de referencia), como las herramientas usadas en la aplicación de la metodología.

3.2.1 Tecnologías y Herramientas para la especificación del dominio del problema

- Resource Description Framework (RDF)

"*Es un modelo estándar para el intercambio de información en la Web. RDF tiene características que permiten la unión de información entre aplicaciones de diferentes plataformas y tecnologías. RDF extiende la estructura de la Web mediante el uso de URIs para denominar los objetos y sus relaciones (usualmente denominado como 'triple')".*[3]Basado en una estructura similar al XML, lo que le da el carácter de universalidad y legibilidad por parte de las computadoras, RDF es el estándar de facto para la web semántica.

El formato RDF se utiliza para genera los resultados y modelar las especificaciones de los servicios web semánticos OWL-S. Aunque ésta especificación puede generar resultados y ser modelada en un formato distinto, se utiliza RDF por ser el más utilizado en la comunidad.

- Web Service Modeling Language (WSML)[4]

WSML es un lenguaje para la especificación de los diferentes elementos de un servicio web, principalmente las ontologías. WSML es la principal tecnología subyacente para especificar un servicio web semántico para el estándar WSMO.

- Apache AXIS

Apache Axis es una implementación del estándar SOAP (*Simple Object Access Protocol*) del W3C (*World Wide Web Consortium*).

"*SOAP es un protocolo ligero para el intercambio de información en un entorno descentralizado y distribuido. Basado en XML el protocolo consiste de tres partes: Una envoltura que define una plataforma para describir qué es un mensaje y cómo procesarlo, un conjunto de reglas de codificación para expresar instancias de tipos de datos definidos por una aplicación, y una convención para representar llamadas a procedimientos remotos y sus respuestas*."[5]

[3]http://www.w3.org/RDF/
[4]http://www.wsmo.org/wsml/wsml-syntax
[5]http://www.w3.org/TR/2007/REC-soap12-part0-20070427/

Mediante este protocolo se realiza una implementación de un servicio web. Apache Axis proporciona una implementación en C++ y Java del estándar SOAP, además de utilidades y librerías para desarrollar y desplegar servicios web.

- Protégé

"*Protégé es una plataforma de código abierto que provee a una creciente comunidad de usuarios, un conjunto de herramientas para construir modelos de dominio y aplicaciones basadas en conocimiento mediante ontologías*".[6]

Con Protégé se puede desarrollar diferentes aplicaciones relacionadas con aspectos semánticos, en especial, con el estándar OWL. Provee herramientas para la creación de ontologías, aplicaciones web, edición de aplicaciones de dominio específico, soporte para la mayoría de formatos comunes en la Web Semántica (RDF/XML/OWL/OBO, entre otros), además de asistentes, editores y validadores de código que ayudan a generar de manera más rápida y con mayor calidad las aplicaciones desarrolladas.

En este proyecto se utiliza la funcionalidad de validador de código OWL de protégé para el código generado a través de la herramienta.

- Eclipse Web Tools Platform

Eclipse Web Tools Platform (Eclipse WTP) es un conjunto de herramientas y librerías dentro del entorno de desarrollo Eclipse, que permiten el desarrollo de proyectos Web. En este trabajo lo utilizamos para enlazar y probar las aplicaciones generadas.

3.2.1 Tecnologías y Herramientas para la aplicación de la metodología MDSD

- Eclipse Modeling Framework (EMF)

Eclipse Web Tools Platform (Eclipse WTP) es una suite de desarrollo dentro del entorno Eclipse que facilita el trabajo de la Ingeniería orientada por modelos (MDE). Utiliza un vasto conjunto de herramientas para trabajar los enfoques más conocidos y los estándares más utilizados en la MDE. En este trabajo se utiliza el entorno EMF durante todo el proceso de la metodología, porque contiene la mayoría de herramientas descritas en este apartado.

[6]http://protegewiki.stanford.edu/wiki/Main_Page

- Xtext[7]

Xtext es la herramienta utilizada para desarrollar el lenguaje específico de dominio DSL. Esta herramienta proporciona un conjunto de artefactos y librerías que permiten desarrollar cualquier tipo de lenguaje, ya sea un lenguaje de dominio específico o un lenguaje de propósito general. Estos artefactos y librerías ayudan a describir todos los aspectos de nuestro lenguaje, y generan una completa implementación del lenguaje generado en la máquina virtual de java, de esta forma, el lenguaje desarrollado se puede trabajar en cualquier entorno java que estemos utilizando.

- Xtend[8]

Esta herramienta se une a Xtext para generar las plantillas de código automático que derivan del DSL. Xtext es un lenguaje de verificación estática de tipos similar a java, pero que ofrece mayor flexibilidad y características que permiten realizar plantillas de código para automatizar aspectos de una aplicación.

3.3 La implementación de referencia

En la fase de implementación de referencia se desarrolla la aplicación, o se toma una aplicación ya desarrollada, que servirá como referencia para las fases futuras. De esta implementación se toman las características para el modelado y el resultado del proceso debe quedar de tal forma que se pueda reproducir tal implementación en un nivel de avance significativo.

Para la implementación de referencia, tomamos un ejemplo de una implementación en la web de una especificación OWL-S. *Zipcodefinder*, es una aplicación que busca los códigos postales de Estados Unidos mediante el código de área, ciudad o estado. Esta es una de las primeras aplicaciones de ejemplo que realizan una especificación completa de OWL-S.

Dentro de la aplicación de ejemplo se encuentran los elementos descritos de la especificación: El perfil del servicio (*profiles*), el acceso a los servicios (*groundings*) y el modelo de servicios (*process*). También se identifican las ontologías utilizadas en esta aplicación. Una aplicación que implementa un servicio web basado en OWL-S por lo general tiene su archivo principal con extensión owl.[9]

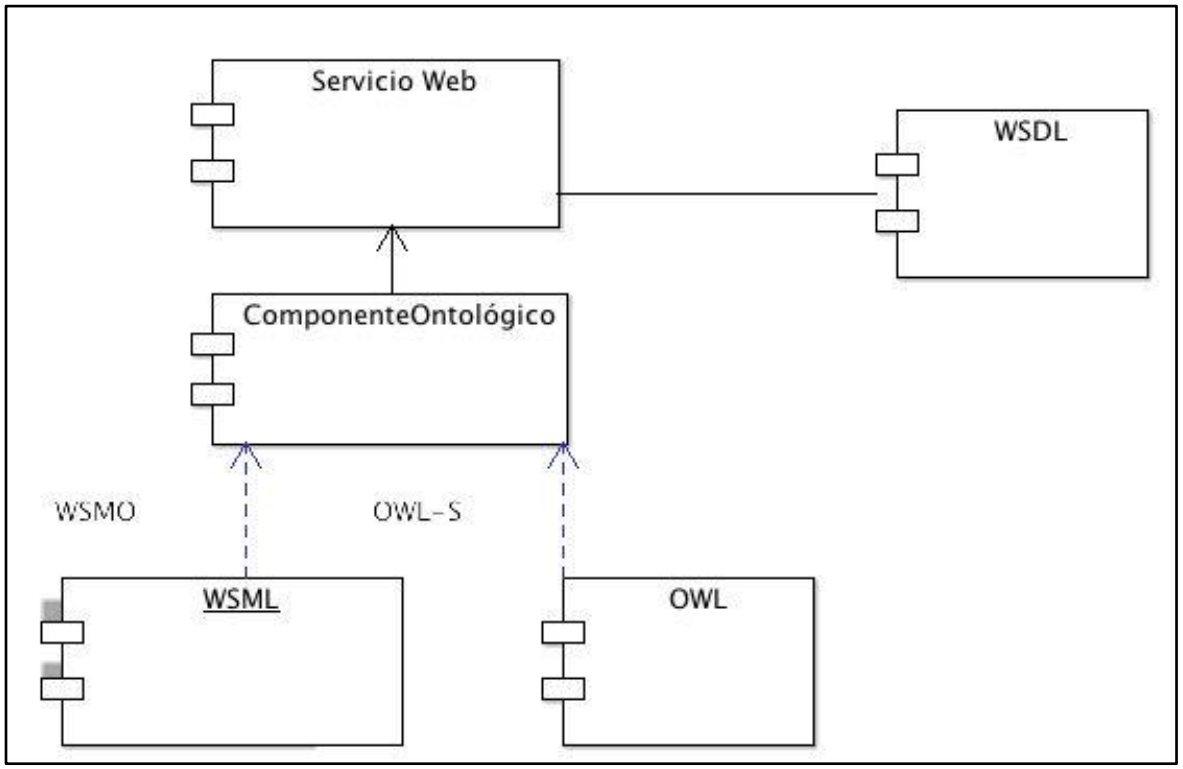
---

[7] http://www.eclipse.org/Xtext/
[8] http://www.eclipse.org/xtend/documentation.html
[9] La especificación de ejemplo está alojada en http://on.cs.unibas.ch/owl-s/1.2/ZipCodeFinder.owl.

Aunque la base de la implementación es el servicio web, es útil que se que se desarrolle un cliente que pueda consumir el servicio y mostrar las capacidades de éste. Es de anotar que el cliente que consume el servicio no está dentro de la generación de código que provee la herramienta.

En la siguiente imagen se puede apreciar la arquitectura de la implementación de referencia.

Figura 3-2. Esquema general de la implementación de referencia



3.3.1 Análisis de código de la implementación de referencia

En esta fase se hace un análisis del código de la implementación de referencia. De este análisis se deben obtener tres tipos de código: Un código genérico, que es idéntico para todas las implementaciones futuras de la aplicación; un código repetitivo esquemático, que no es idéntico para cada aplicación, pero tiene las mismas características sistemáticas a través de las aplicaciones (ejemplo: se repite utilizando un patrón); y un código individual, o específico de la aplicación, que no puede ser generalizado a otras implementaciones.

El código genérico identificado en la implementación de referencia, está representado por el código de encabezado de la especificación OWL-S, que consiste en la identificación del documento.

En la figura 3-2 se muestran los elementos de los que se identifican los tipos de código genérico.

Figura 3-3. Código genérico Identificado en el análisis de la implementación de referencia

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.2/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.2/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.2/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.2/Grounding.owl">
  <!ENTITY mind "http://on.cs.unibas.ch/owl-s/1.2/MindswapProfileHierarchy.owl">
  <!ENTITY zip "http://www.daml.org/2001/10/html/zipcode-ont">
]>
<rdf:RDF
  xmlns:rdf="&rdf;#"
  xmlns:rdfs="&rdfs;#"
  xmlns:owl="&owl;#"
  xmlns:xsd="&xsd;#"
  xmlns:service="&service;#"
  xmlns:profile="&profile;#"
  xmlns:process="&process;#"
  xmlns:grounding="&grounding;#"
  xmlns:mind="&mind;#"
  xml:base="http://on.cs.unibas.ch/owl-s/1.2/ZipCodeFinder.owl">
```

La especificación WSMO a través del lenguaje WSML muestra cómo código genérico de la misma forma que OWL-S su encabezado. Este encabezado se compone de los elementos, agrupados en un componente denominado namespaces, se utilizan a lo largo de la estructura del documento para definir las propiedades y valores de los demás elementos del documento. Generalmente, estos elementos de la cabecera no cambian.

En la siguiente figura se puede observar un esquema con estos elementos.

Figura 3-4. Ejemplo de encabezado del formato WSML

```
namespace { _"http://www.wsmo.org/ontologies/sfs/ZipCodeFinder#",
    dc _"http://purl.org/dc/elements/1.1#",
    dt _"http://www.wsmo.org/ontologies/dateTime#",
    bo _"http://www.wsmo.org/ontologies/sfs/zipcodes#",
    ho _"http://www.wsmo.org/ontologies/sfs/zip#",
    loc _"http://www.wsmo.org/ontologies/location#",
    xsd _"http://www.w3.org/2001/XMLSchema#"
}
```

El código esquemático repetitivo se identifica rápidamente en la especificación. Corresponde a cada uno de los elementos que hacen parte de la especificación (*profiles, process, groundings*). Estos elementos se repiten dentro de la especificación, con algunas diferencias que son definidas por las propiedades de cada uno de ellos.

En la siguiente figura, se observa que los elementos del formato: servicio (service), perfiles (mind, profiles) y procesos (process) se repiten indefinidamente en él. Esto indica que los elementos deben estar representados en el modelo que definamos. Cada uno de estos elementos puede representar una clase dentro del modelo de nuestra propuesta. Se puede observar que también estos elementos contienen propiedades, que pueden representar atributos de estas clases, por ejemplo, para el caso de los servicios, tiene propiedades como *serviceName, profiles, hasInput, hasOutput*, entre otros.

Figura 3-5. Código repetitivo en la implementación de referencia

```
<!-- Service description -->
<service:Service rdf:ID="ZipCodeFinderService">
    <service:presents rdf:resource="#ZipCodeFinderProfile"/>

    <service:describedBy rdf:resource="#ZipCodeFinderProcess"/>

    <service:supports rdf:resource="#ZipCodeFinderGrounding"/>
</service:Service>

<!-- Profile description -->
<mind:MapService rdf:ID="ZipCodeFinderProfile">
    <service:presentedBy rdf:resource="#ZipCodeFinderService"/>

    <profile:serviceName xml:lang="en">Find ZipCode</profile:serviceName>
    <profile:textDescription xml:lang="en">Retorna el codigo zip para la
        ciudad seleccionada.</profile:textDescription>

    <profile:hasInput rdf:resource="#City"/>
    <profile:hasInput rdf:resource="#State"/>

    <profile:hasOutput rdf:resource="#ZipCode"/>
</mind:MapService>

<!-- Process description -->
<process:AtomicProcess rdf:ID="ZipCodeFinderProcess">
    <service:describes rdf:resource="#ZipCodeFinderService"/>

    <process:hasInput rdf:resource="#City"/>
    <process:hasInput rdf:resource="#State"/>

    <process:hasOutput rdf:resource="#ZipCode"/>
</process:AtomicProcess>

<process:Input rdf:ID="City">
    <process:parameterType rdf:datatype="&xsd;#anyURI">&xsd;#string</process:parameterType>
    <rdfs:label>0:City</rdfs:label>
</process:Input>
```

Por el lado del código esquemático repetitivo para la especificación WSMO, se observa que los elementos (*ontology, webservice, capabiity, goals, mediator*, entre otros) tienen la misma estructura: inician con la sentencia *nonFunctionalProperties* y terminan con la sentencia *endNonFunctionalProperties*, determinando un bloque de elemento dentro de la estructura. Luego, dentro de cada bloque, se incluyen sentencias similares para cada uno de los elementos, por ejemplo, *name, description*, y las variables que incluyan estos elementos, y si tienen valores, se les agrega la etiqueta *hasValue*. En la siguiente figura se puede observar uno de los elementos, el elemento webservice esquematizado.

Figura 3-6. Elemento webService del formato WSML

```
webService _ "http://www.wsmo.org/webservices/sfs/ZipCodeFinder"

nonFunctionalProperties
  dc#title hasValue "Online Zip Code Find Web Service"
  dc#creator hasValue "Wilvec"
  dc#subject hasValue {"ZipCode", "Zip Code finder", "Code", "State"}
  dc#description hasValue "Online Zip Code Find Web Service"
  dc#publisher hasValue "Wilvec"
  dc#contributor hasValue {"Wilman Vega,"Henry Umaña"}
  dc#date hasValue "2013-11-21"
  dc#type hasValue _"http://www.wsmo.org/2004/d2#webservice"
  dc#format hasValue "text/html"
  dc#identifier hasValue _"http://www.wsmo.org/webservices/sfs/ZipCodeFinder"
  dc#source hasValue _"http://www.tilisoft.com/ws/LocInfo/City"
  dc#language hasValue "en-US"
endNonFunctionalProperties
```

3.4 Determinación del meta modelo de la aplicación

Una vez identificados los tipos de código de la implementación de referencia, es necesario establecer el meta modelo de la aplicación, que servirán de base para el diseño del lenguaje específico de dominio que comprende la siguiente fase.

Para el desarrollo del meta modelo no se requirieron herramientas especiales. La especificación del meta modelo puede hacerse a por medio de UML. En este trabajo se utilizó la herramienta *Ecore* de eclipse que se integra con las otras herramientas que se utilizaron en la aplicación del enfoque.

Dentro del meta modelo de la aplicación se tienen las siguientes entidades: Modelo, representa el modelo de la especificación en su conjunto; ontología, que representa la ontología (o las ontologías) a utilizar; Los servicios, y en un nivel de modelo están: Los perfiles, los procesos y los Accesos (*groundings*).

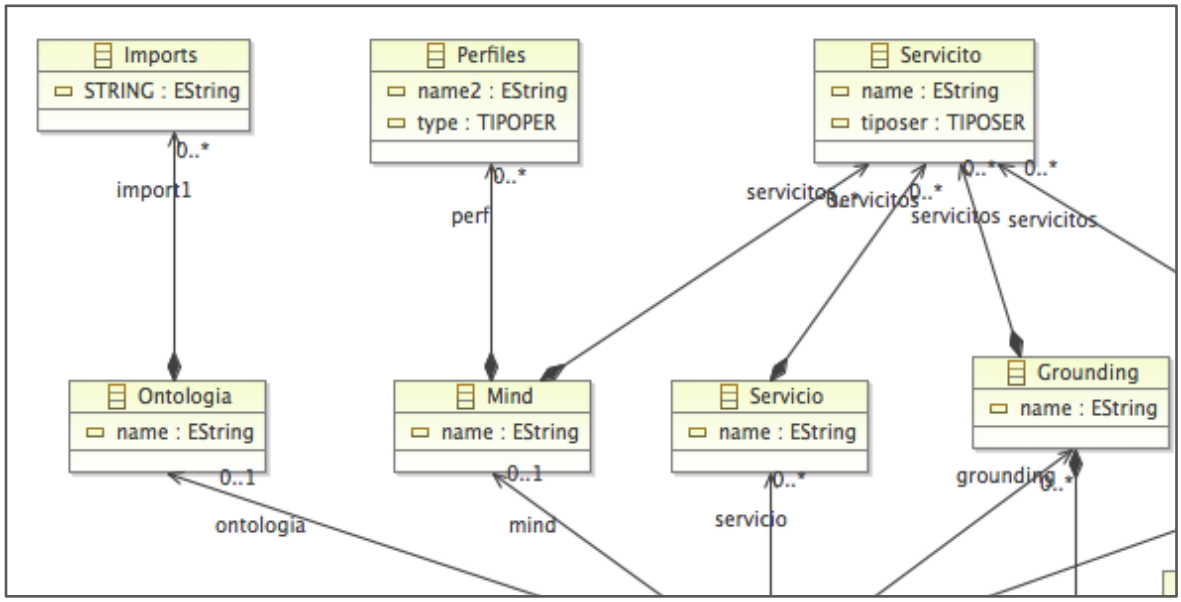
Una vez identificados los tipos de código de la implementación de referencia, es necesario establecer el meta modelo de la aplicación, que servirá de base para el diseño del lenguaje específico de dominio que comprende la siguiente fase.

Para el desarrollo del meta modelo no se requirieron herramientas especiales. La especificación del meta modelo puede hacerse por medio de UML. En este trabajo se utilizó la herramienta Ecore de eclipse que se integra con las otras herramientas que se utilizaron en la aplicación del enfoque.

Dentro del meta modelo de la aplicación se tienen las siguientes entidades: Modelo, representa el modelo de la especificación en su conjunto; ontología, que representa la ontología (o las ontologías) a utilizar; Los servicios, y en un nivel de modelo están: Los perfiles, los procesos y los Accesos (*groundings*).

La siguiente figura muestra los elementos que forman parte del meta modelo.

Figura 3-7. Meta modelo de la aplicación.



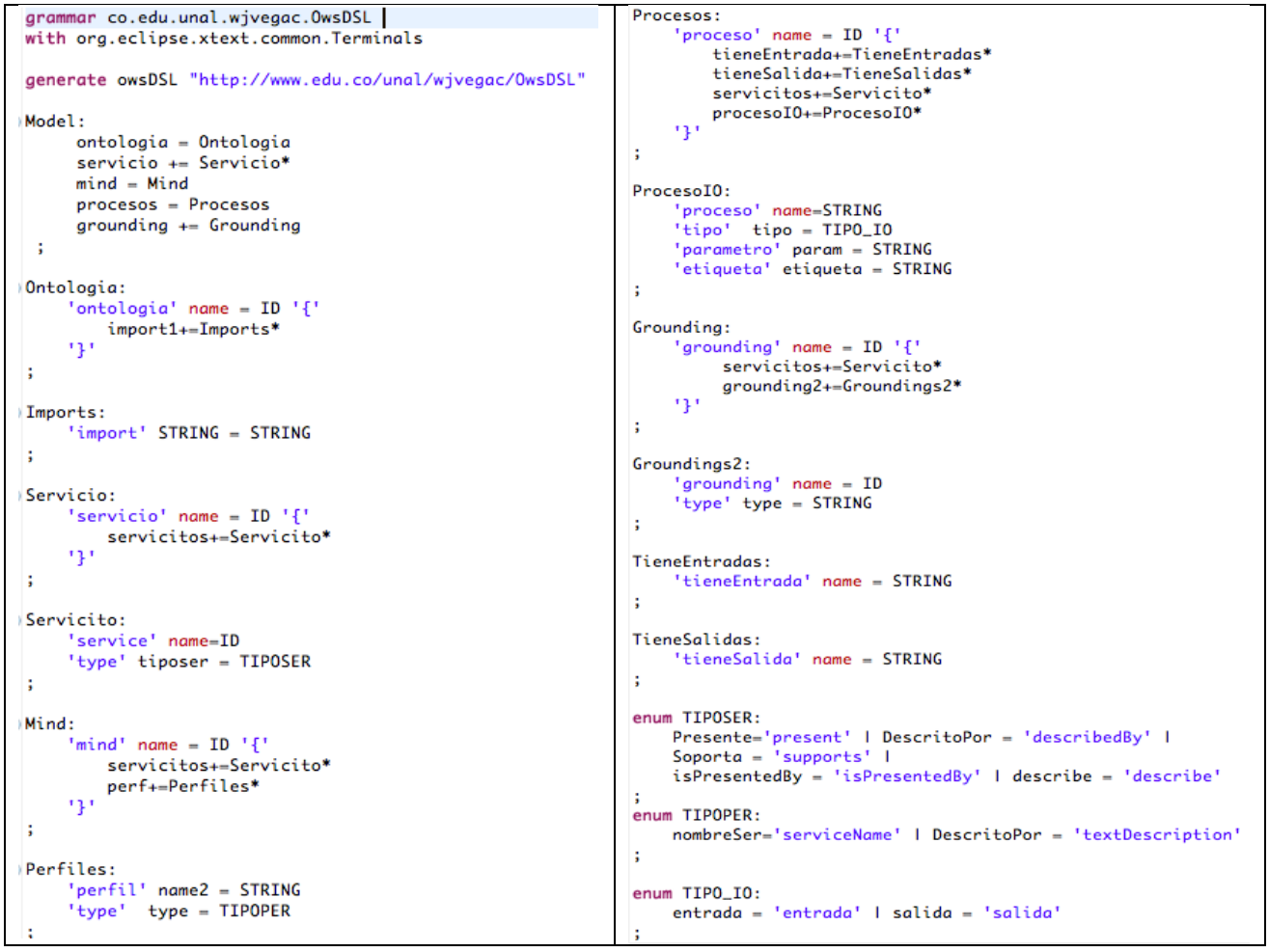
3.5 Diseño del lenguaje de dominio específico

Esta fase consiste en el diseño del Lenguaje de Dominio Específico (DSL, por sus siglas en inglés), basado en el meta modelo obtenido en la fase anterior. En esta fase se establece una línea con el código generado y el dominio de la aplicación. En este punto

se estableció a través de la herramienta Xtext el DSL de la especificación de referencia. En la siguiente imagen se puede ver una porción de código del DSL construido.

El DSL ha sido denominado owlDSL (referente a *Ontology Web Service Domain Specific Language*). En la figura 3-5 se pueden apreciar cada uno de los elementos que componen las especificaciones que se generaran a partir de la herramienta.
En la primera parte se define los lineamientos generales de la especificación en la regla *Model.* Se define dentro del DSL, que se incluirán ontologías, perfiles de servicio procesos y acceso a servicios (groundings) y luego se desarrolla cada elemento dentro del DSL. para resumir, se establecen las propiedades de cada elemento, siendo las más comunes nombre y tipo, y seguidamente se establece si contiene más elementos (se indica por las llaves '{' y '}')

Figura 3-8. Diseño del Lenguaje específico de Dominio.



```
grammar co.edu.unal.wjvegac.OwsDSL
with org.eclipse.xtext.common.Terminals

generate owsDSL "http://www.edu.co/unal/wjvegac/OwsDSL"

Model:
     ontologia = Ontologia
     servicio += Servicio*
     mind = Mind
     procesos = Procesos
     grounding += Grounding
 ;

Ontologia:
    'ontologia' name = ID '{'
        import1+=Imports*
    '}'
;

Imports:
    'import' STRING = STRING
;

Servicio:
    'servicio' name = ID '{'
        servicitos+=Servicito*
    '}'
;

Servicito:
    'service' name=ID
    'type' tiposer = TIPOSER
;

Mind:
    'mind' name = ID '{'
        servicitos+=Servicito*
        perf+=Perfiles*
    '}'
;

Perfiles:
    'perfil' name2 = STRING
    'type'  type = TIPOPER
;
```

```
Procesos:
    'proceso' name = ID '{'
        tieneEntrada+=TieneEntradas*
        tieneSalida+=TieneSalidas*
        servicitos+=Servicito*
        procesoIO+=ProcesoIO*
    '}'
;

ProcesoIO:
    'proceso' name=STRING
    'tipo'  tipo = TIPO_IO
    'parametro' param = STRING
    'etiqueta' etiqueta = STRING
;

Grounding:
    'grounding' name = ID '{'
         servicitos+=Servicito*
         grounding2+=Groundings2*
    '}'
;

Groundings2:
    'grounding' name = ID
    'type' type = STRING
;

TieneEntradas:
    'tieneEntrada' name = STRING
;

TieneSalidas:
    'tieneSalida' name = STRING
;

enum TIPOSER:
    Presente='present' | DescritoPor = 'describedBy' |
    Soporta = 'supports' |
    isPresentedBy = 'isPresentedBy' | describe = 'describe'
;
enum TIPOPER:
    nombreSer='serviceName' | DescritoPor = 'textDescription'
;

enum TIPO_IO:
    entrada = 'entrada' | salida = 'salida'
;
```

3.6 Plantillas de generación de código automático

Una vez se obtenido el DSL diseñado, este se proporcionó como insumo a la herramienta Xtend, para que hacer la interpretación del DSL y realizar la generación de código automático. Con la herramienta Xtend y el DSL como insumo, insertamos los tipos de código identificados en la fase de análisis. El código genérico se introduce tal y como se encuentra en la implementación de referencia, y se implementan los patrones de código necesarios para poder adaptar el código esquemático repetitivo. En este trabajó se utilizó el patrón de diseño denominado *Delegate*. En la siguiente figura se puede apreciar el código generado con la herramienta Xtend.

Figura 3-9. Plantilla de generación de código automático.

```
package co.edu.unal.wjvegac.generator

import co.edu.unal.wjvegac.owsDSL.Grounding

class GeneradorOWLS {

    def generarOntologias(Ontologia onto)'''
    <?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE uridef [
      <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
      <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
      <!ENTITY owl "http://www.w3.org/2002/07/owl">
      <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
      <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
      <!ENTITY profile "http://www.daml.org/services/owl-s/1.1/Profile.owl">
      <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
      <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
      <!ENTITY mind "http://www.mindswap.org/2004/owl-s/1.1/MindswapProfileHierarchy.owl">
        ]>
    <rdf:RDF
        xmlns:rdf="&rdf;#"
        xmlns:rdfs="&rdfs;#"
        xmlns:owl="&owl;#"
        xmlns:xsd="&xsd;#"
        xmlns:service="&service;#"
        xmlns:profile="&profile;#"
        xmlns:process="&process;#"
        xmlns:grounding="&grounding;#"
        xmlns:mind="&mind;#"
        xml:base="http://www.mindswap.org/2004/owl-s/1.1/ZipCodeFinder.owl">
    <owl:Ontology rdf:about="«onto.name»">
    «FOR imports2 : onto.import1»
    <owl:imports rdf:resource="«imports2.STRING»"/>
    «ENDFOR»
    </owl:Ontology>
    '''
```

En esta imagen se puede observar el código genérico, que es el código dentro de las tres comillas seguidas, que dentro de la sintaxis de Xtend indica código que no cambia dentro de la transformación. El código esquemático repetitivo está presente a través la combinación de este código, de comodines para representación de variables, como por ejemplo (<<, >>), y de los componentes propios del lenguaje, como las sentencias de bifurcación (IF…ELSE…ENDIF) o las de iteración, como el caso de la imagen anterior (FOR…ENDFOR).

3.7 Pruebas sobre el código generado

A través de un proyecto dentro del mismo entorno que se está utilizando, se realiza una prueba del lenguaje de dominio especifico que utilizamos en la herramienta. La extensión seleccionada para los archivos que soportan este lenguaje es owl3, en consideración a una de las especificaciones que se generan.

En la siguiente figura se muestra la implementación del DSL construido para generar una aplicación completamente similar a nuestra implementación de referencia.

Figura 3-10. Implementación del DSL desarrollado

modelo.owl3

```
ontologia ontologia1 {
    import "http://www.daml.org/services/owl-s/1.1/Service.owl"
    import "http://www.daml.org/services/owl-s/1.1/Profile.owl"
    import "http://www.daml.org/services/owl-s/1.1/Process.owl"
    import "http://www.daml.org/services/owl-s/1.1/Grounding.owl"
    import "http://www.daml.org/2001/10/html/zipcode-ont"
    }

    servicio FindLatLongService {
        service FindLatLongProfile
        type present
        service FindLatLongProcess
        type describedBy
        service FindLatLongGrounding
        type supports
    }

    mind FindLatLongProfile {
        service FindLatLongService
        type present
        perfil "ZipCode"
        type serviceName
        perfil "Returns the zip code for the given city/state"
        type textDescription
    }

    proceso ZipCodeFinderProcess {
        tieneEntrada "City"
        tieneEntrada "State"
        tieneSalida "ZipCode"
        service ZipCodeFinderService
        type describe

        proceso "City"
        tipo entrada
        parametro "http://www.w3.org/2001/XMLSchema#anyURI"
        etiqueta "0:City"

        proceso "State"
        tipo entrada
        parametro "http://www.w3.org/2001/XMLSchema#anyURI"
        etiqueta "1:State"
```

Una vez que guardamos el archivo que se construyó con el DSL, se actualiza el proyecto para que tome los cambios que realiza. La ventaja es que una vez que el archivo se construye, los artefactos son generados de manera automática, como lo se puede apreciar en las figuras 2-8, 2-9, 2-10 y 2-11.

Figura 3-11. Artefactos generados en la herramienta

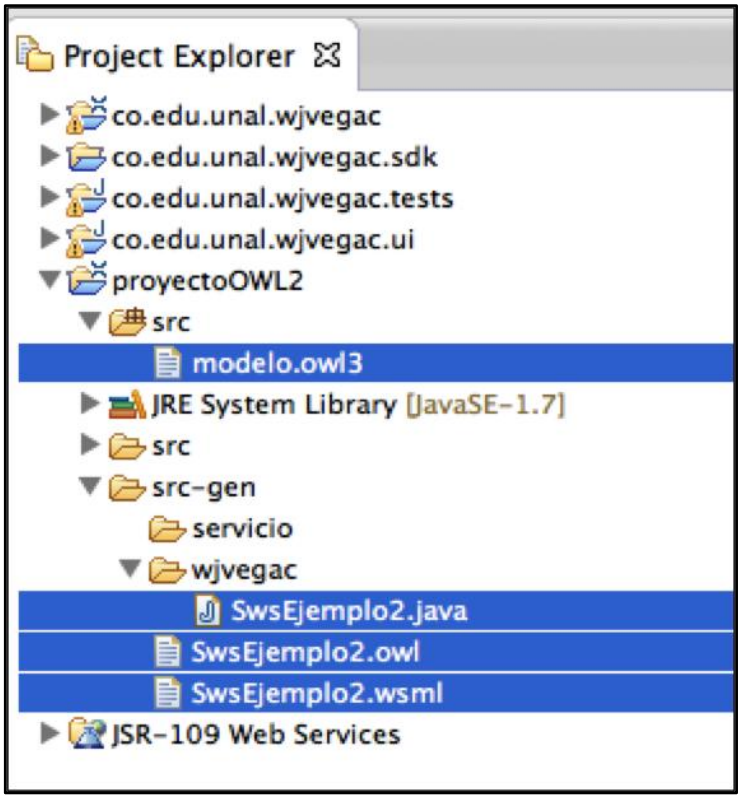


Figura 3-12. Código de ejemplo de clase para servicio web generado

```
package co.edu.unal.wjvegac;

import java.util.List;

import co.edu.unal.wilvec.db.AccesoDB;
import co.edu.unal.wilvec.db.ZipCodeVO;

public class SwsEjemplo2 {

        private AccesoDB db = null;


        public List<Object> getListZipCodeByCity(String city){
            db = new AccesoDB();
            return db.getListZipCodeByCity(city);
        }

        public Object getZipCodeByZip(String zipCode){
            db = new AccesoDB();
            return db.getZipCodeByZip(zipCode);
        }
    }

}
```

Figura 3-13. Archivo de la especificación OWL-S generado

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE uridef [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.1/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
  <!ENTITY mind "http://www.mindswap.org/2004/owl-s/1.1/MindswapProfileHierarchy.owl">
  ]>
<rdf:RDF
   xmlns:rdf="&rdf;#"
     xmlns:rdfs="&rdfs;#"
     xmlns:owl="&owl;#"
     xmlns:xsd="&xsd;#"
     xmlns:service="&service;#"
     xmlns:profile="&profile;#"
     xmlns:process="&process;#"
     xmlns:grounding="&grounding;#"
     xmlns:mind="&mind;#"
```

Figura 3-14. Archivo de la especificación WSML generado

```
namespace { _"http://www.mindswap.org/2004/owl-s/1.1/ontologia1#",
     rdf _"http://www.w3.org/1999/02/22-rdf-syntax-ns#",
     rdfs _"http://www.w3.org/2000/01/rdf-schema#",
     owl _"http://www.w3.org/2002/07/owl#",
     service _"http://www.daml.org/services/owl-s/1.1/Service.owl#",
     process _"http://www.daml.org/services/owl-s/1.1/Process.owl#",
     grounding _"http://www.daml.org/services/owl-s/1.1/Grounding.owl#",
     dc _"http://purl.org/dc/elements/1.1/",
     wsml _"http://www.wsmo.org/wsml/wsml-syntax#",
     xsd _"http://www.w3.org/2001/XMLSchema#" }

ontology _"http://www.mindswap.org/2004/owl-s/1.1/ontologia1"

     importsOntology
          { _"http://www.daml.org/services/owl-s/1.1/Service.owl",
            _"http://www.daml.org/services/owl-s/1.1/Profile.owl",
            _"http://www.daml.org/services/owl-s/1.1/Process.owl",
            _"http://www.daml.org/services/owl-s/1.1/Grounding.owl",
            _"http://www.daml.org/2001/10/html/zipcode-ont"}

instance _"http://www.mindswap.org/2004/owl-s/1.1/ZipCodeFinder.owl##State" memberOf process#Input
```

Para las pruebas sobre el código generado se debe realizar la completitud del servicio web a través del código individual, que corresponde a código de implementación de los métodos del servicio. Luego se debe desarrollar un cliente que consuma el servicio web generado y realizar una prueba sobre el servicio. En la figura 2-12 se puede apreciar la interfaz del cliente que consume el servicio generado en ambas especificaciones y los resultados arrojados. También se puede apreciar el servicio web publicado en el servidor local listo para el consumo, mediante la tecnología Axis2 (figura 2-13).

Figura 3-15. Interfaz gráfica del cliente que consume el servicio generado

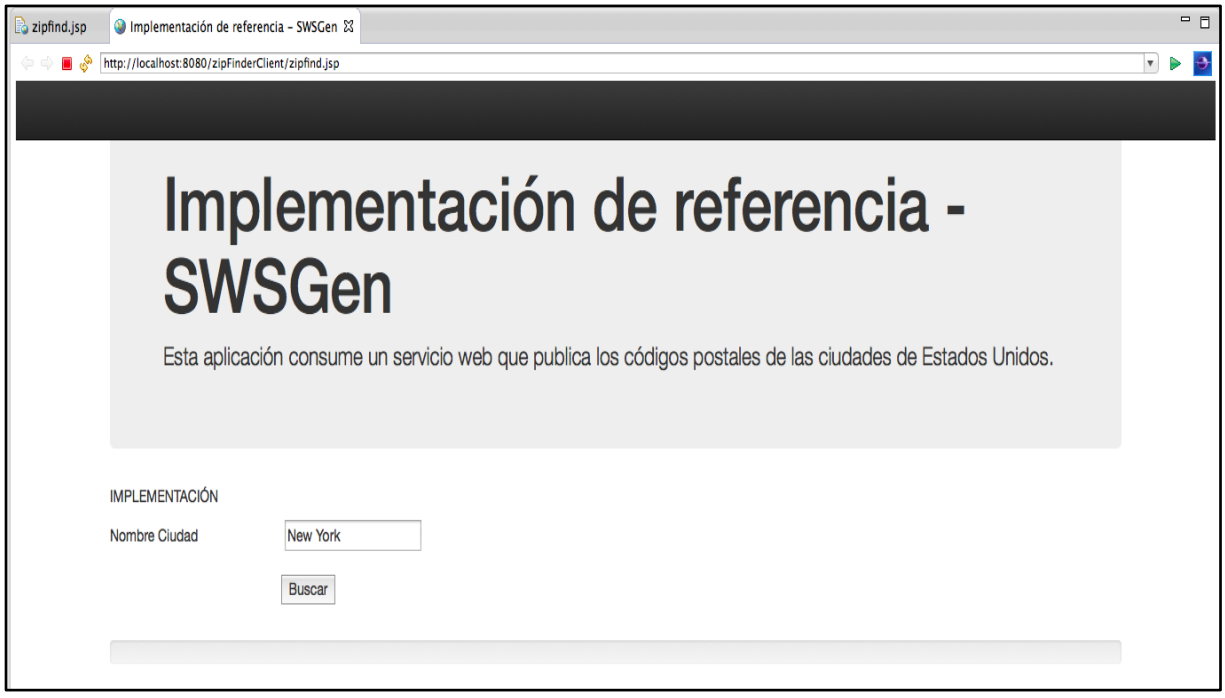
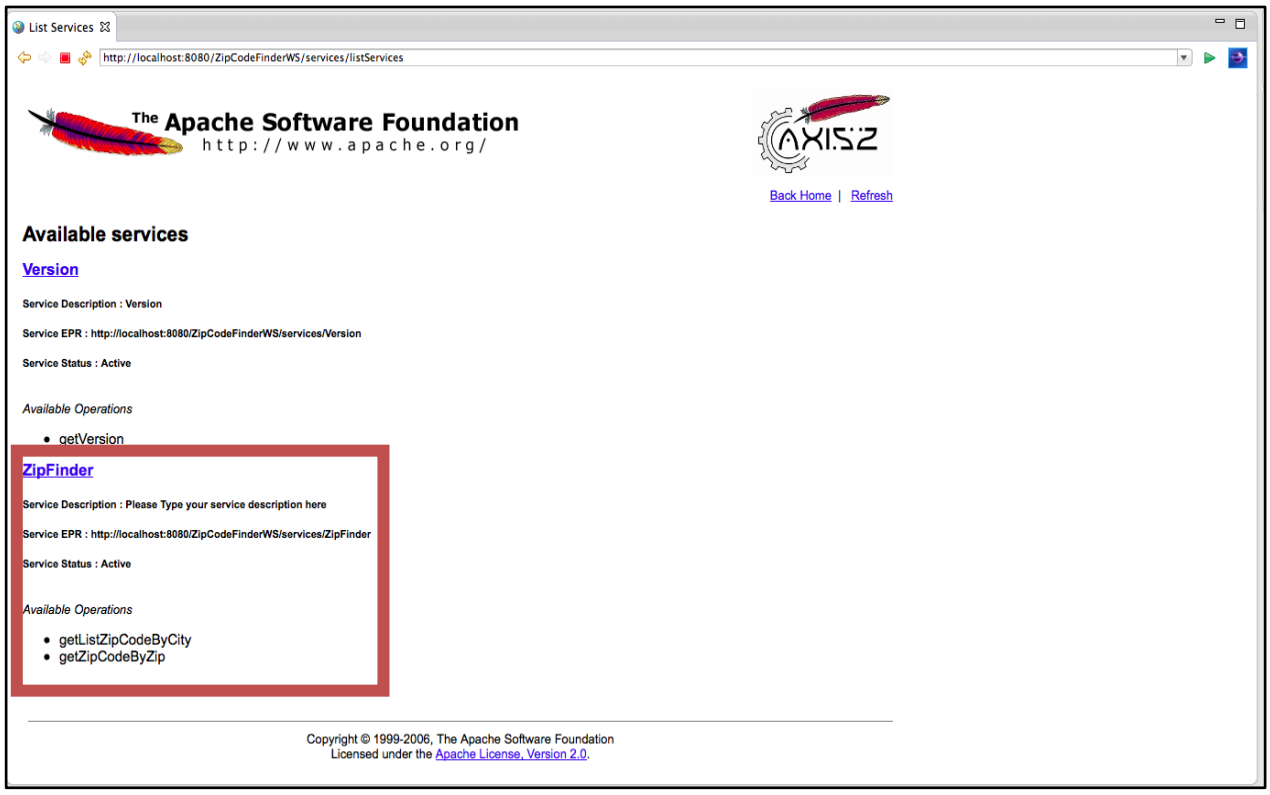


Figura 3-16. Publicación del servicio web, mediante tecnología Apache Axis2



4. Evaluación de la herramienta de desarrollo de Servicios Web Semánticos

En este apartado se presentan los criterios de evaluación de la herramienta, a travésde los artefactos generados, se selecciona una aplicación dentro de un dominio común del mundo empresarial y se aplican y dichos criterios en comparación con la escritura manual del código fuente.

El primer criterio de evaluación importante es que la herramienta cumpla con las promesas que aporta el desarrollo dirigido por modelos, que se describieron en el apartado 2.6: incremento en la velocidad de desarrollo, mejora de la calidad del desarrollo, optimización de la capacidad de gestión y complejidad del software y mejores prácticas de desarrollo. Mas que la evaluación de la herramienta en sí, aquí se evalúan los resultados del seguimiento de la metodología desarrollada en el segundo capítulo de este documento,

Por otra parte para realizar una evaluación técnica, en la ejecución de métricas de software sobre código fuente, como lo son las pruebas de Halsted [**9**], relacionadas con la producción de líneas de código y el esfuerzo de trabajo requerido para desarrollar una aplicación. La selección de este tipo de pruebas corresponde a que determinan de una manera específica aspectos relevantes en el desarrollo de aplicaciones como la complejidad del software, el esfuerzo de trabajo, la facilidad de mantenimiento y de realización de pruebas.

4.1 Selección de la prueba de concepto

La aplicación de ejemplo a generar a través de la herramienta es una aplicación de reserva hotelera. Tomada como implementación de referencia, pero esta vez no para la construcción sino para la generación en sí y su comparación con el código generado a través de la herramienta.

La aplicación consiste en realizar una consulta a las ofertas de hospedaje en una ciudad, consultar datos como: Características de las habitaciones, precio, disponibilidad, etc. Y seguidamente presentar la opción de realizar la reserva, con los datos del cliente como documento de identidad, nombre y apellidos, datos de contacto e información de pago.

Para esta aplicación es necesario exponer los siguientes servicios:

- Consultar hoteles
- Detallar habitaciones
- Realizar reserva

También se realizó la importación de ontologías relacionadas con el dominio de la aplicación de ejemplo. En este caso se hicieron los ajustes correspondientes para incorporar ontologías relacionadas con la aplicación seleccionada. [**15**]

Cómo se realizó en el capítulo anterior, en el diseño de la herramienta, se desarrollo una aplicación web que consumía el servicio web que se había construido a mano y el que se había generado. En esta ocasión se hizo de la misma forma para realizar la prueba del caso de estudio. En la figura 3-1 se muestra la interfaz de la aplicación que consumen los servicios.

Figura 4-1. Aplicación cliente que consume los servicios de la prueba de concepto

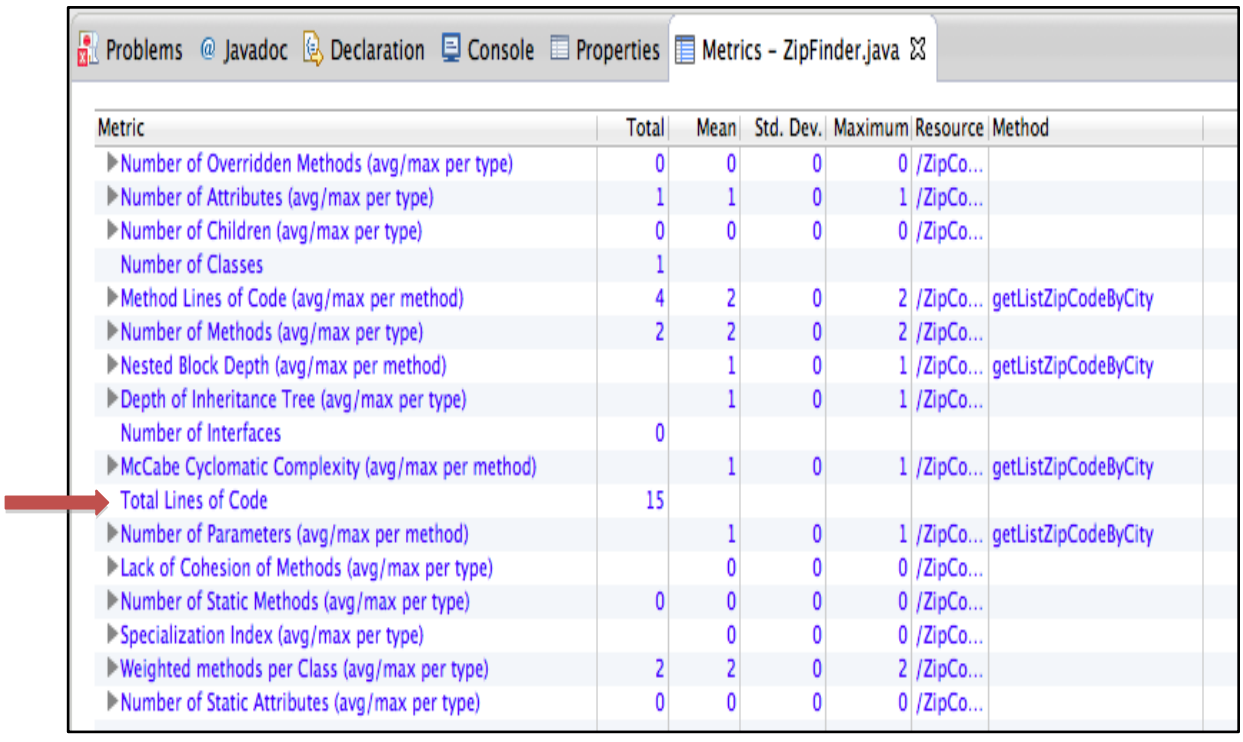
4.2 Análisis de resultados

Cómo se especificó anteriormente, se utilizaron métricas básicas de complejidad de software para la prueba de la herramienta. A nivel general se miden los archivos de código java generados y se realiza un conteo sobre los archivos de las especificaciones, debido a qué son archivos planos.

Para ejecutar las métricas, se utilizó el plug-in del entorno de desarrollo Eclipse llamado *Metrics*[10], que permite realizar de manera automática las métricas seleccionadas (Métrica de línea de código fuente: de complejidad y Halsted).

En el conteo general, que se aprecian en las figuras 3-2 y 3-1, el número de líneas de código generadas por la herramienta es menor, a razón del 66%. Esto es, de manera sencilla, que el desarrollador requiere codificar un 34% de las líneas para completar la aplicación. Obviamente, sin tener en cuenta el código individual que requiere la aplicación para poder efectuar su objetivo.

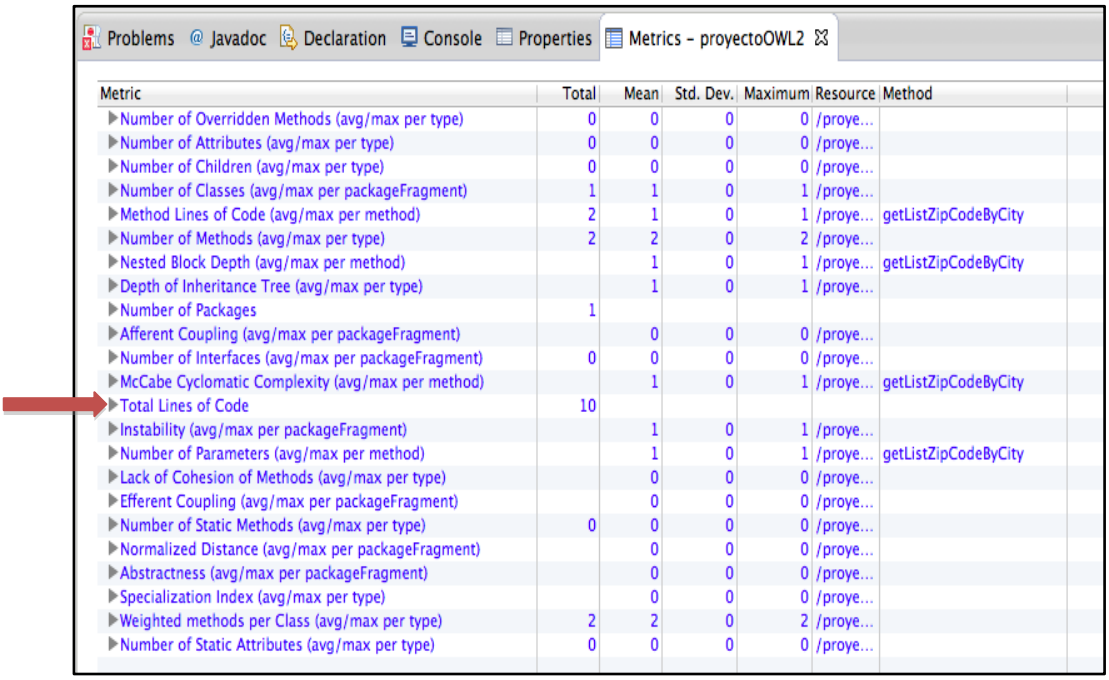
Figura 4-2. Métricas generadas para el código manual



Metric	Total	Mean	Std. Dev.	Maximum	Resource	Method
Number of Overridden Methods (avg/max per type)	0	0	0	0	/ZipCo...	
Number of Attributes (avg/max per type)	1	1	0	1	/ZipCo...	
Number of Children (avg/max per type)	0	0	0	0	/ZipCo...	
Number of Classes	1					
Method Lines of Code (avg/max per method)	4	2	0	2	/ZipCo...	getListZipCodeByCity
Number of Methods (avg/max per type)	2	2	0	2	/ZipCo...	
Nested Block Depth (avg/max per method)		1	0	1	/ZipCo...	getListZipCodeByCity
Depth of Inheritance Tree (avg/max per type)		1	0	1	/ZipCo...	
Number of Interfaces	0					
McCabe Cyclomatic Complexity (avg/max per method)		1	0	1	/ZipCo...	getListZipCodeByCity
Total Lines of Code	15					
Number of Parameters (avg/max per method)		1	0	1	/ZipCo...	getListZipCodeByCity
Lack of Cohesion of Methods (avg/max per type)		0	0	0	/ZipCo...	
Number of Static Methods (avg/max per type)	0	0	0	0	/ZipCo...	
Specialization Index (avg/max per type)		0	0	0	/ZipCo...	
Weighted methods per Class (avg/max per type)	2	2	0	2	/ZipCo...	
Number of Static Attributes (avg/max per type)	0	0	0	0	/ZipCo...	

[10] http://metrics.sourceforge.net/

Figura 4-3. Métricas de la código generado a través de la herramienta



Metric	Total	Mean	Std. Dev.	Maximum	Resource	Method
Number of Overridden Methods (avg/max per type)	0	0	0	0	/proye...	
Number of Attributes (avg/max per type)	0	0	0	0	/proye...	
Number of Children (avg/max per type)	0	0	0	0	/proye...	
Number of Classes (avg/max per packageFragment)	1	1	0	1	/proye...	
Method Lines of Code (avg/max per method)	2	1	0	1	/proye...	getListZipCodeByCity
Number of Methods (avg/max per type)	2	2	0	2	/proye...	
Nested Block Depth (avg/max per method)		1	0	1	/proye...	getListZipCodeByCity
Depth of Inheritance Tree (avg/max per type)		1	0	1	/proye...	
Number of Packages	1					
Afferent Coupling (avg/max per packageFragment)		0	0	0	/proye...	
Number of Interfaces (avg/max per packageFragment)	0	0	0	0	/proye...	
McCabe Cyclomatic Complexity (avg/max per method)		1	0	1	/proye...	getListZipCodeByCity
Total Lines of Code	10					
Instability (avg/max per packageFragment)		1	0	1	/proye...	
Number of Parameters (avg/max per method)		1	0	1	/proye...	getListZipCodeByCity
Lack of Cohesion of Methods (avg/max per type)		0	0	0	/proye...	
Efferent Coupling (avg/max per packageFragment)		0	0	0	/proye...	
Number of Static Methods (avg/max per type)	0	0	0	0	/proye...	
Normalized Distance (avg/max per packageFragment)		0	0	0	/proye...	
Abstractness (avg/max per packageFragment)		0	0	0	/proye...	
Specialization Index (avg/max per type)		0	0	0	/proye...	
Weighted methods per Class (avg/max per type)	2	2	0	2	/proye...	
Number of Static Attributes (avg/max per type)	0	0	0	0	/proye...	

En cuanto a los archivos de las especificaciones, la diferencia se encuentra en la forma de los documentos, más no en su estructura. Debido a que estos archivos tienen un formato plano, se les hizo una comparación simple a través de la herramienta *WC*[11] del sistema operativo Linux, que realiza un conteo de líneas palabras y caracteres. Con algunas pequeñas diferencias, los archivos son generados en su completitud para poder ser configurados con el servicio web.

Tabla 4-1. Conteo de líneas, palabras y caracteres de los archivos generados

Archivo	Líneas	Palabras	Caracteres	Especificación
hotelBookingGenerated.wsml	376	1061	13176	WSML
hotelBookingManual.wsml	407	1327	16190	
HotelBookingServiceGen.owl	436	789	19393	OWL-S
HotelBookingServiceManual.owl	594	1280	23516	

Este indicador para los archivos planos no proporciona mayor información para un análisis profundo. En general la herramienta genera menos líneas para los archivos de

[11]http://www.linuxmanpages.com/man1/wc.1.php

las especificaciones por que omite espacios y no genera líneas de comentarios que siempre realizan los programadores.

A nivel general los resultados arrojados por la herramienta, en comparación con la manera tradicional de construir los servicios web semánticos son discretos, pero cabe destacar que la si se analiza la efectividad de la herramienta en la generación de las dos especificaciones, el resultado es satisfactorio.

Para los criterios que tienen relación con las promesas que da el desarrollo dirigido por modelos, hay que tener en cuenta la forma tradicional de desarrollo de las aplicaciones de servicios web semánticos comparado con el desarrollo propuesto en este trabajo.

Para el desarrollo de servicios web semánticos, generalmente [**13**] [**21**] [**5**] se utilizan herramientas distintas para los componentes que hacen parte de servicio web semántico. Se utilizan herramientas como Protégé, *Visual Ontology Modeler*[12], *Web Service Modeling Toolkit*[13]entre otros, para la construcción o importación de ontologías. Luego se utiliza un entorno de desarrollo, como Eclipse, Netbeans[14] o IntelliJIDEA[15], para el desarrollo del resto de elementos del servicio web. Este mismo procedimiento se debe realizar para la generación del servicio web semántico en cada una de las especificaciones, OWL-S y WSMO. En esta propuesta se plantea, a través de una sola herramienta unificar esta forma de trabajar en una sola, a través de la codificación en un solo lenguaje (el lenguaje especifico diseñado en capítulo 3, base fundamental de la herramienta), se generan los elementos de manera automática desde un punto central, de una sola vez. Esto incrementa la velocidad en el desarrollo de los componentes, a medida que estos se vuelven más y más complejos.

El nivel de esfuerzo por parte del desarrollador es mucho menor, debido a que la herramienta genera los archivos necesarios para ambas especificaciones. Como la generación de archivos es inmediata, también le evita tener que utilizar herramientas de conversión de una especificación a otra, es más puede integrar en un solo proyecto de aplicación los dos formatos generados para los servicios web.

[12]http://www.sandsoft.com/products.html
[13]http://sourceforge.net/projects/wsmt/
[14]https://netbeans.org/
[15]http://www.jetbrains.com/idea/

A través de la utilización de modelos se puede aumentar la calidad de los productos generados. En los archivos de la especificación generada se encuentra menor cantidad de errores que los inducidos por la generación manual y, aunque menos probable, en las herramientas señaladas anteriormente. Además, la utilización de modelos permite la generación de caso de prueba para los componentes generados. También la validación del modelo mismo puede permitir el desarrollo de modelos más complejos con una mayor calidad [**4**].

5. Conclusiones

5.1 Conclusión general

En este trabajo se ha aplicado un enfoque para el desarrollo de servicios web semánticos, en las especificaciones WSMO y OWL-S. esto se logró comparando las especificaciones a generar y siguiendo los pasos de una metodología basada en el desarrollo dirigido por modelos.

De manera más técnica, se identificaron los factores comunes dentro de las especificaciones, que se encontraron en el acceso a servicios, se modelaron los componentes principales de las especificaciones, por ejemplo en la especificación OWL-S, los perfiles de servicio y los procesos, y en WSMO, en el lenguaje de ejecución, las metas, mediadores.

Se genera de forma individual los archivos que conforman cada una de las especificaciones, al igual que los archivos necesarios para formar el servicio web. En un ambiente de ejecución, como lo es el entorno de desarrollo integrado Eclipse, se organizan en un esquema de proyecto web dinámico y se ejecutan en un servidor de aplicaciones con capacidades de ejecución de servicios web.

5.2 Objetivos del trabajo

En este apartado se realiza un análisis final a los objetivos planteados y la determinación de cumplimiento de cada uno de éstos, en base desarrollado en el presente trabajo.

El primer objetivo, analizar los enfoques MDD para el desarrollo de Servicios Web Semánticos que apliquen dentro de sus modelos las especificaciones OWL-S y WSMO, se cumple completamente, en el desarrollo del primer capítulo del presente trabajo, específicamente en el apartado 1.7., donde, se revisaron los enfoques más relevantes para el desarrollo de servicios web semánticos, se destacaron los que correspondían a

las especificaciones OWL-S y WSMO y se hizo una breve descripción de ventajas y desventajas frente a la herramienta que se propuso.

El segundo objetivo, diseñar los componentes de la herramienta propuesta mediante enfoque MDD y utilizando la herramienta XText y un Lenguaje de dominio específico – DSL, se cumple de manera cabal en todo el desarrollo del segundo capítulo, a partir del seguimiento de la metodología MDD, se expone paso a paso, el diseño y prueba de la herramienta.

El tercer objetivo, Evaluar la aplicación de la herramienta propuesta a través de una prueba de concepto, se cumple totalmente, partiendo de la selección de un caso de estudio que se adapte a las características requeridas para la prueba, una aplicación acercada a la realidad, preferiblemente en el entorno empresarial, fue escogida para la prueba. Se revisaron algunas métricas básicas para medir la efectividad del código generado por la herramienta y finalmente se hace una comparación con el código desarrollado de forma tradicional.

5.3 Principales contribuciones

Los principales aportes que se presentan en este trabajo se puede destacar:

- Un documento del estado del arte del desarrollo dirigido por modelos centrado en la construcción de servicios web semánticos.
- La comparación a nivel de modelos de las especificaciones OWL-S y WSMO
- La aplicación práctica de desarrollo de servicios web semánticos, que queda documentada en el trabajo y sus anexos.

Si bien estos aspectos se pueden encontrar en la revisión de literatura sobre el tema de Desarrollo dirigido por modelos aplicado a los servicios web semánticos, este trabajo presenta una actualización de conceptos y un enfoque más claro en la aplicación de la metodología para que pueda ser un referente de consulta de mucha utilidad en estas áreas de conocimiento.

5.4 Trabajo futuro

Del desarrollo de este documento deriva la formulación de preguntas de de investigación que permiten continuar el trabajo sobre la línea de investigación en la que se fundamenta el desarrollo de este estudio.

Se puede realizar un modelo comparativo de enfoques dirigido por modelos para servicios web semánticos mucho más robusto, con unos elementos y criterios de evaluación objetivos y que se puedan automatizar, incluso, que apoye a la decisión de utilizar un enfoque dependiendo de factores como caso de uso, contexto, orientación hacia ciertas tecnologías , entre otros.

Un aspecto interesante de abordar y que ha quedado por fuera de los límites de este trabajo es propender por la realización de un modelo integrador de especificaciones de servicios web semánticos, al igual que un validador general para estas especificaciones.

Una actividad a completar sobre el desarrollo de este trabajo, es que la herramienta genere la estructura completa del servicio web, tanto su estructura de despliegue, como la estructura de desarrollo para los principales entornos de desarrollo. De igual forma, es conveniente generar de manera automática, el o los clientes que consumen los servicios, para dejar al programador un elemento de prueba frente al servicio web generado.

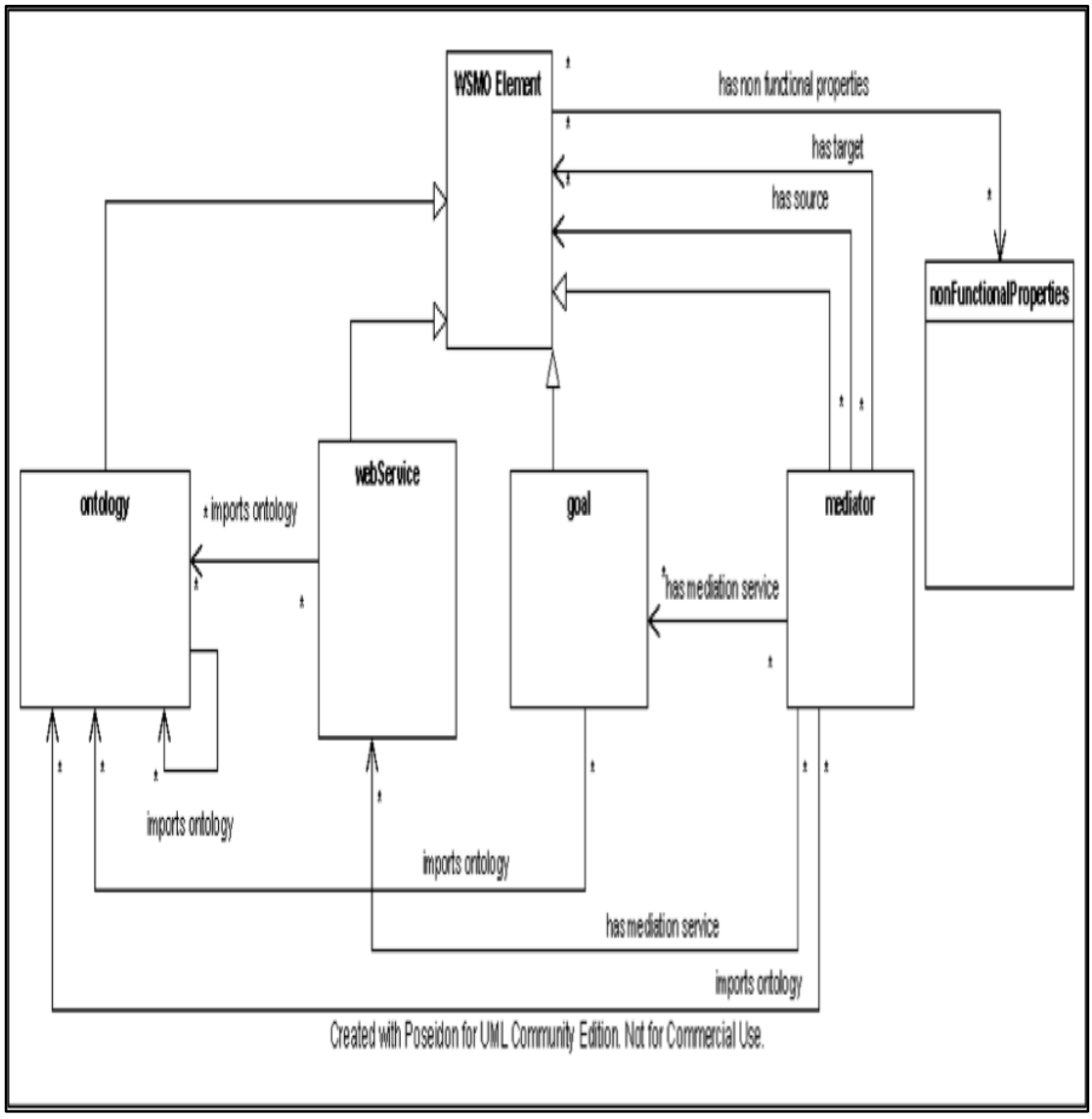
A. Clases y subclases de la especificación OWL-S

Perfil de servicios (ProfileClass)	presents
	presentedBy
	serviceName
	textDescription
	contactInformation
	hasParameter
	hasInput
	hasOutput
	hasPrecondition
	hasResult
Modelo de Servicios (ModelService Class)	<owl:Classrdf:about="#Parameter"> <rdfs:subClassOfrdf:resource="&swrl;#Variable"/> </owl:Class> <owl:DatatypePropertyrdf:ID="parameterType"> <rdfs:domainrdf:resource="#Parameter"/> <rdfs:rangerdf:resource="&xsd;anyURI"/> </owl:DatatypeProperty> <owl:Classrdf:ID="Parameter"> <rdfs:subClassOf> <owl:Restriction> <owl:onPropertyrdf:resource="#parameterType" /> <owl:minCardinalityrdf:datatype="&xsd;#nonNegativeInteger"> 1</owl:minCardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
Modelo de Servicios (Model Service Class) (continuación)	<owl:Classrdf:ID="Input"> <rdfs:subClassOfrdf:resource="#Parameter"/> </owl:Class> <owl:Classrdf:ID="Output"> <rdfs:subClassOfrdf:resource="#Parameter"/> </owl:Class>

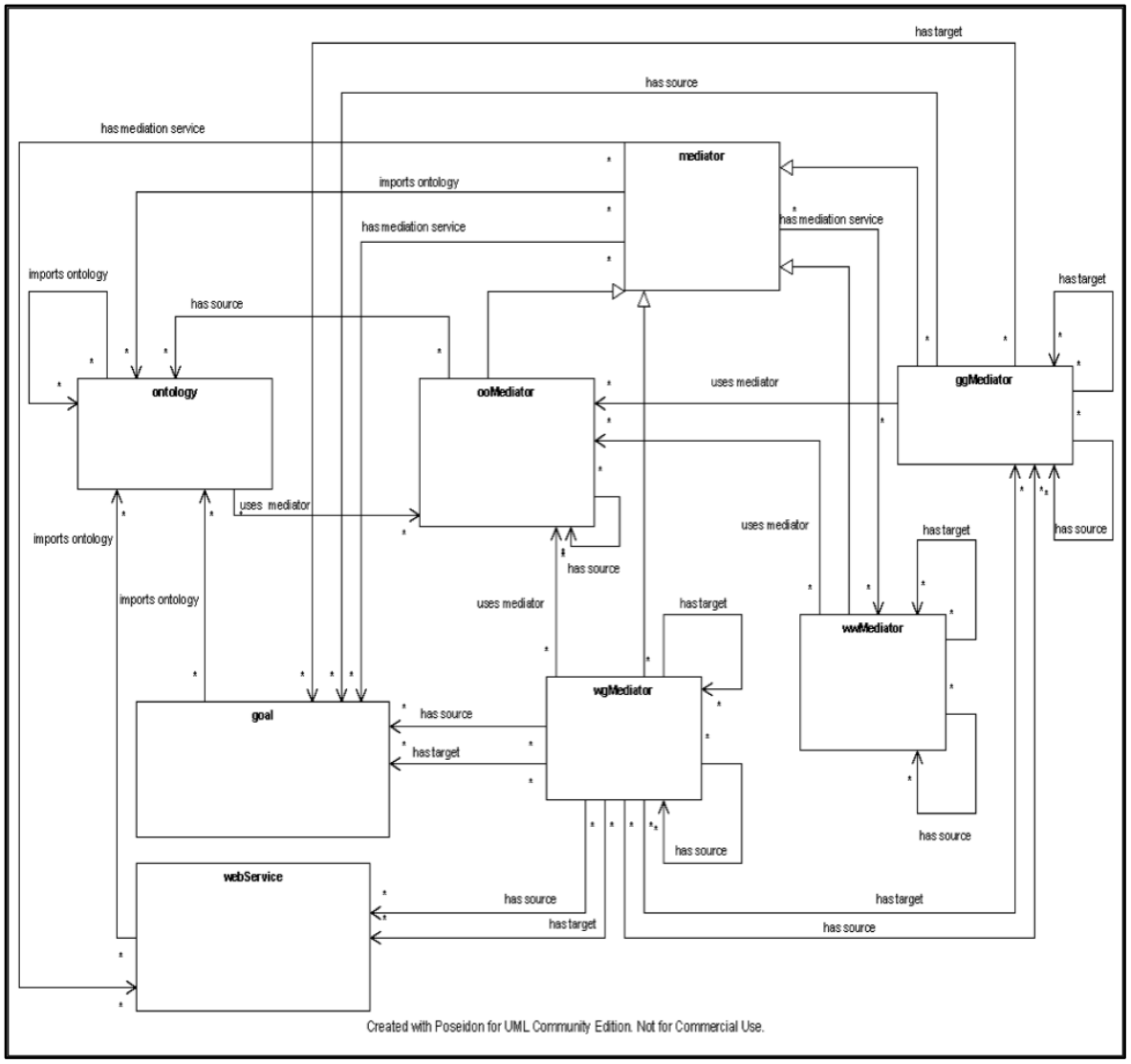
Modelo de Servicios (Model Service Class) (continuación)	<owl:Classrdf:ID="Expression"> <rdfs:subClassOf> <owl:Restriction> <owl:onPropertyrdf:resource="#expressionLanguage"/> <owl:cardinalityrdf:datatype="&xsd;nonNegativeInteger"> 1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf> <owl:Restriction> <owl:onPropertyrdf:resource="#expressionBody"/> <owl:cardinalityrdf:datatype="&xsd;nonNegativeInteger"> 1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class> <owl:ObjectPropertyrdf:ID="&expr;#expressionLanguage"> <rdfs:domainrdf:resource="&expr;#Expression"/> <rdfs:rangerdf:resource="&expr;#LogicLanguage"/> </owl:ObjectProperty> <Description rdf:about="#process2"> <hasPrecondition> <Expression expressionLanguage="&drs;#DRS"> <process:expressionBody> <drs:Atomic_formula> <rdf:predicaterdf:resource="&agnt;#Know_val_is"/> <rdf:subject> <drs:Functional_term> <drs:functionrdf:resource="&ecom;credit_card_num"/> <drs:term_argsrdf:parseType="Collection"> <swrl:Variablerdf:resource="#CC"/> </drs:term_args> </drs:Functional_term> </rdf:subject> <rdf:objectrdf:resource="#Num"/> </drs:Atomic_formula> </process:expressionBody> </Expression> </hasPrecondition> </Description> <owl:ObjectPropertyrdf:ID="hasPrecondition"> <rdfs:domainrdf:resource="#Process"/> <rdfs:rangerdf:resource="&expr;#Condition"/> </owl:ObjectProperty>
Acceso a servicios (Grounding Class)	wsdlVersion wsdlDocument wsdlOperation wsdlService, wsdlPort wsdlInputMessage wsdlInput wsdlOutputMessage wsdlOutput

B. Diagrama de clases de la especificación WSMO

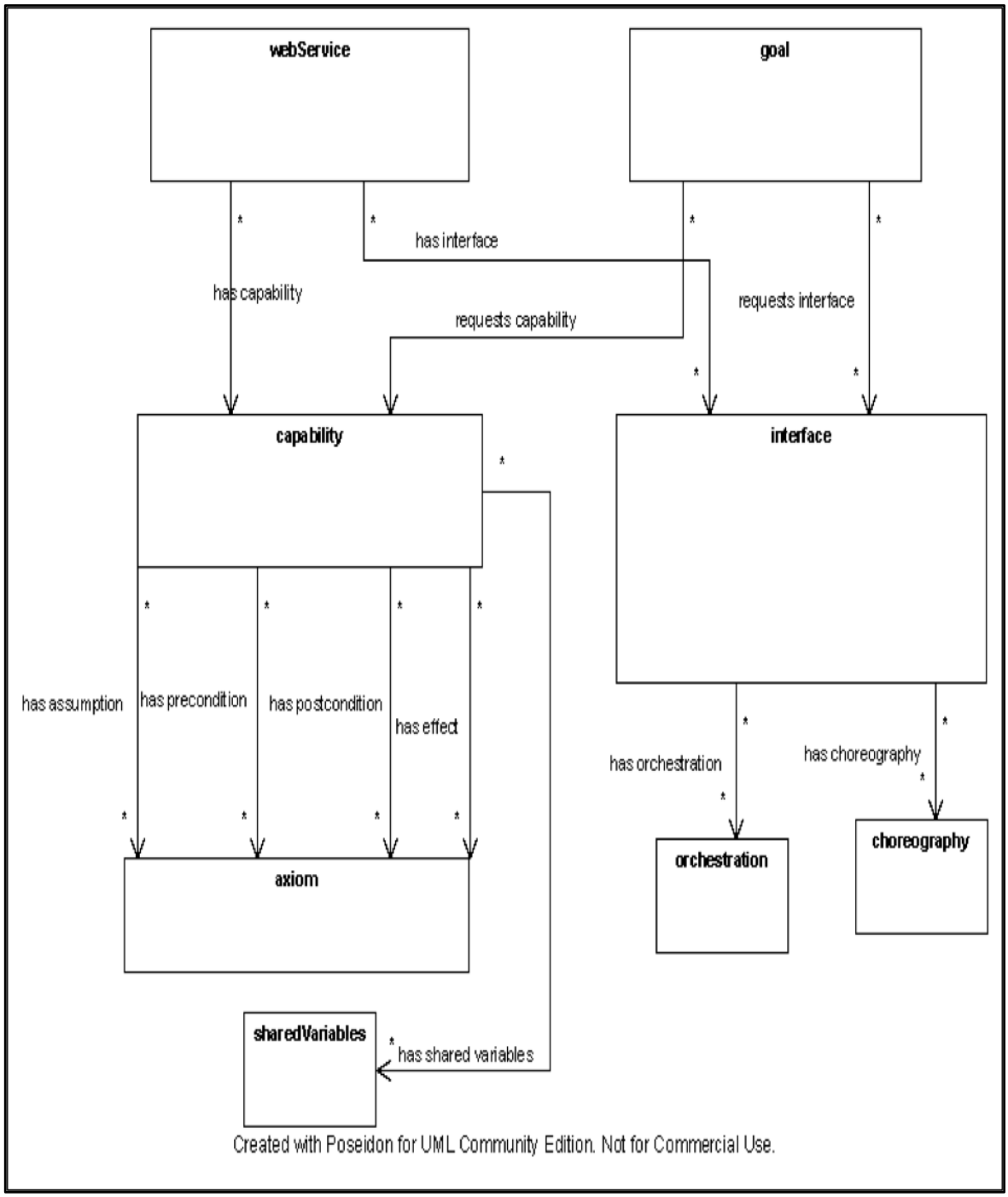
Elementos WSMO



Mediadores

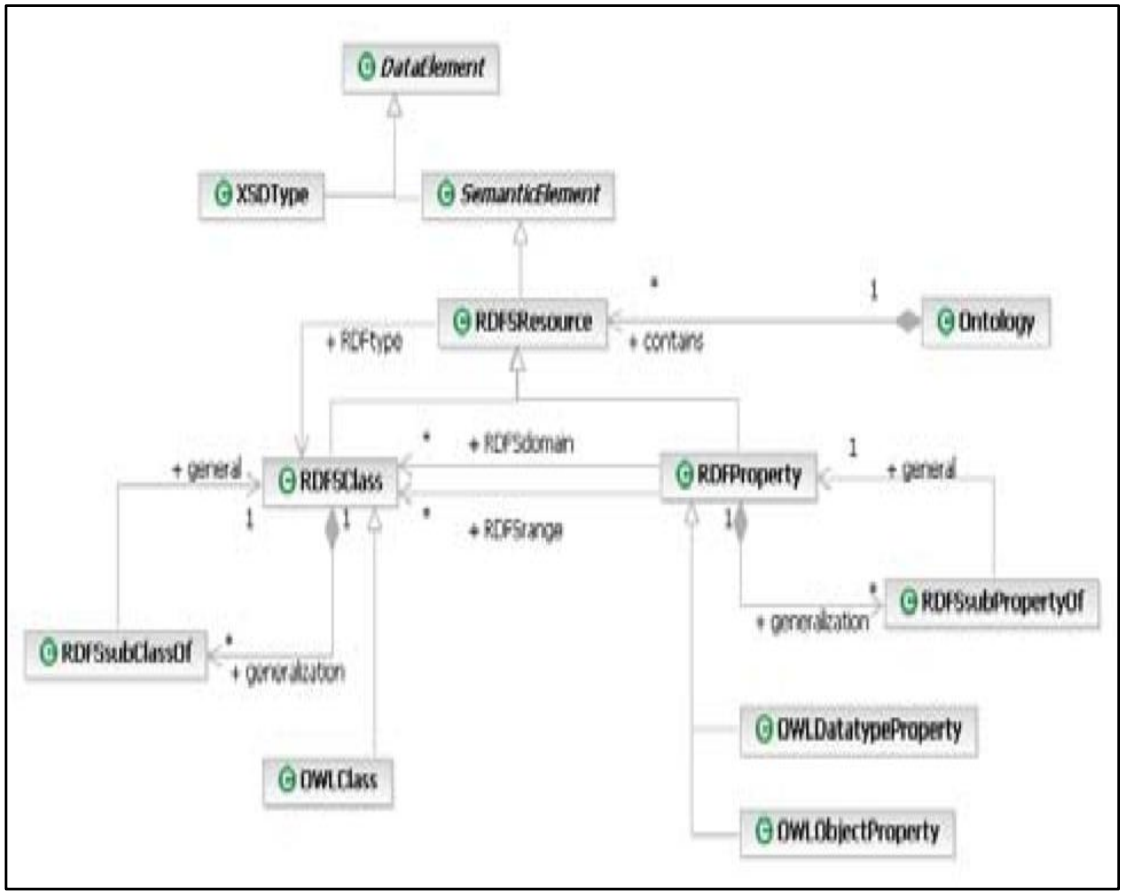

has target
has source
has mediation service
mediator
imports ontology
has mediation service
has mediation service
imports ontology
has source
has target
ontology
ooMediator
uses mediator
ggMediator
uses mediator
has source
imports ontology
has target
has source
imports ontology
uses mediator
has target
wwMediator
wgMediator
goal
has source
has target
has source
webService
has source
has target
has target
has source
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Metas y servicios

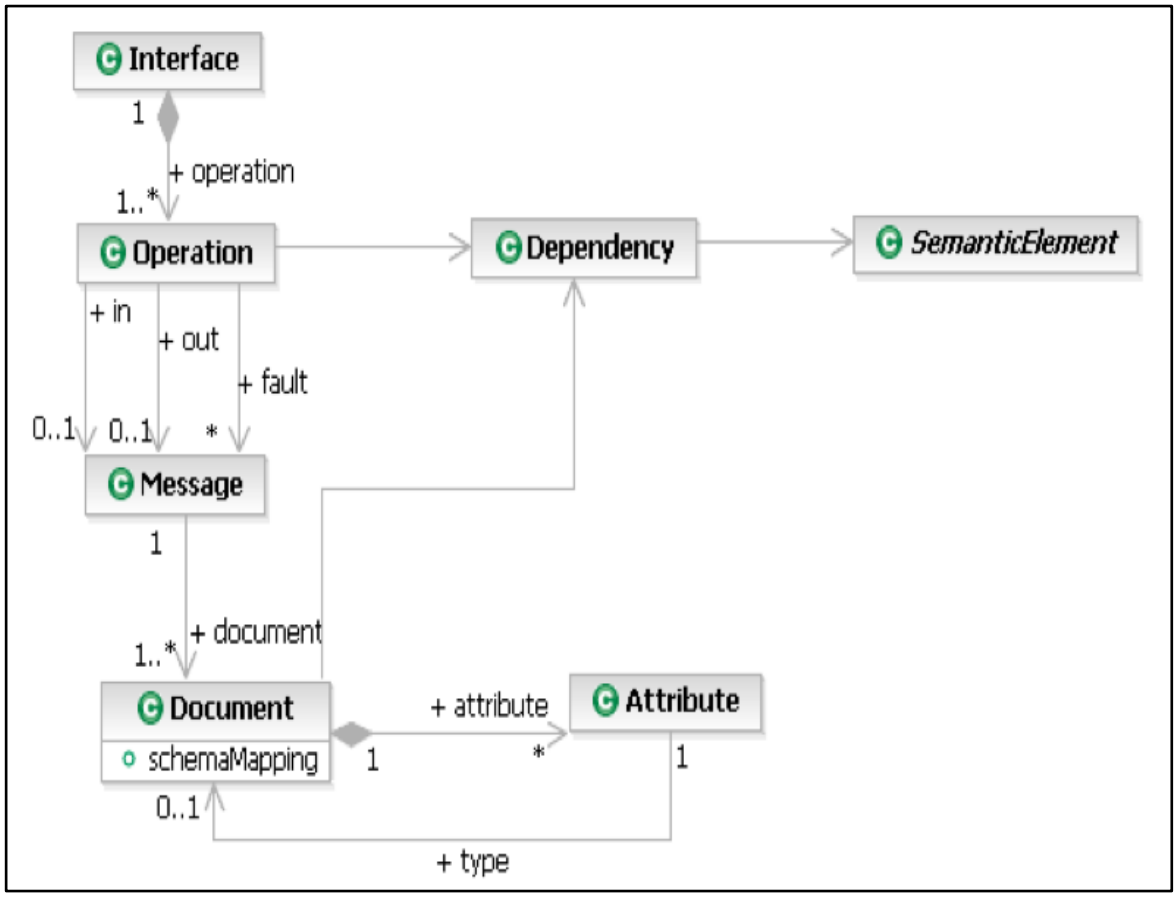


C. Meta modelos del enfoque propuesto por [21]

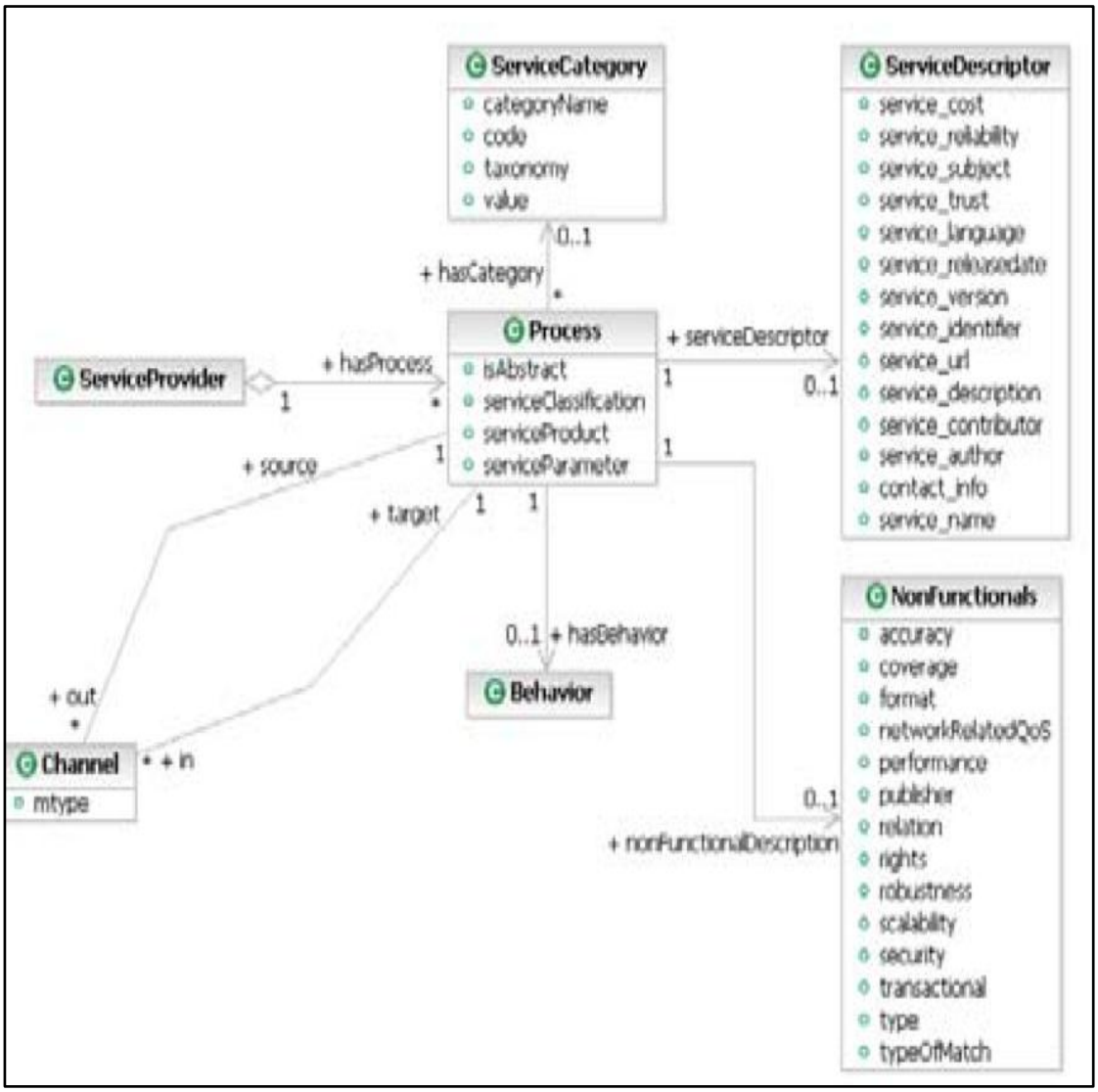
Meta modelo para la ontología



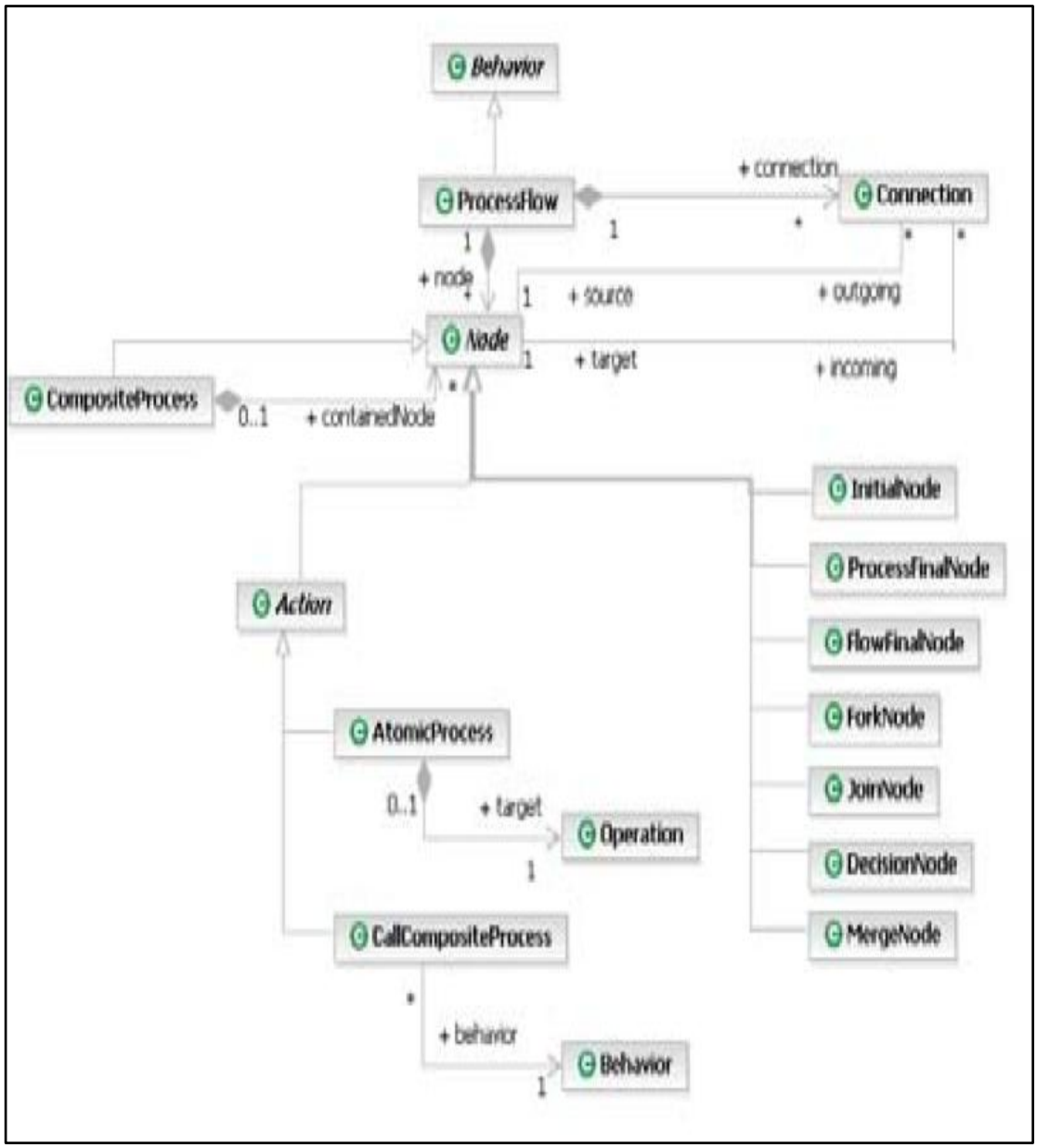
Meta modelo para las interfaces, operaciones y mensajes



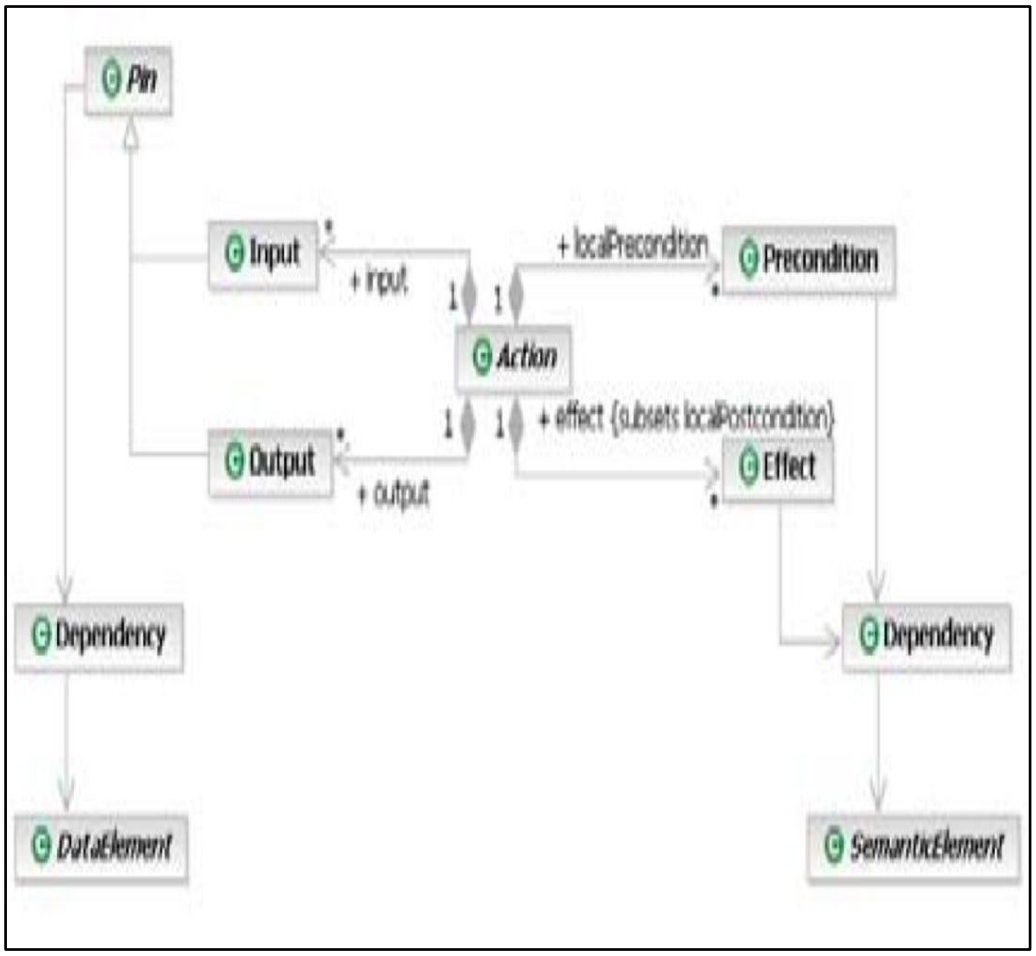
Meta modelo para los elementos que proveen el servicio



Meta modelo para el flujo de procesos

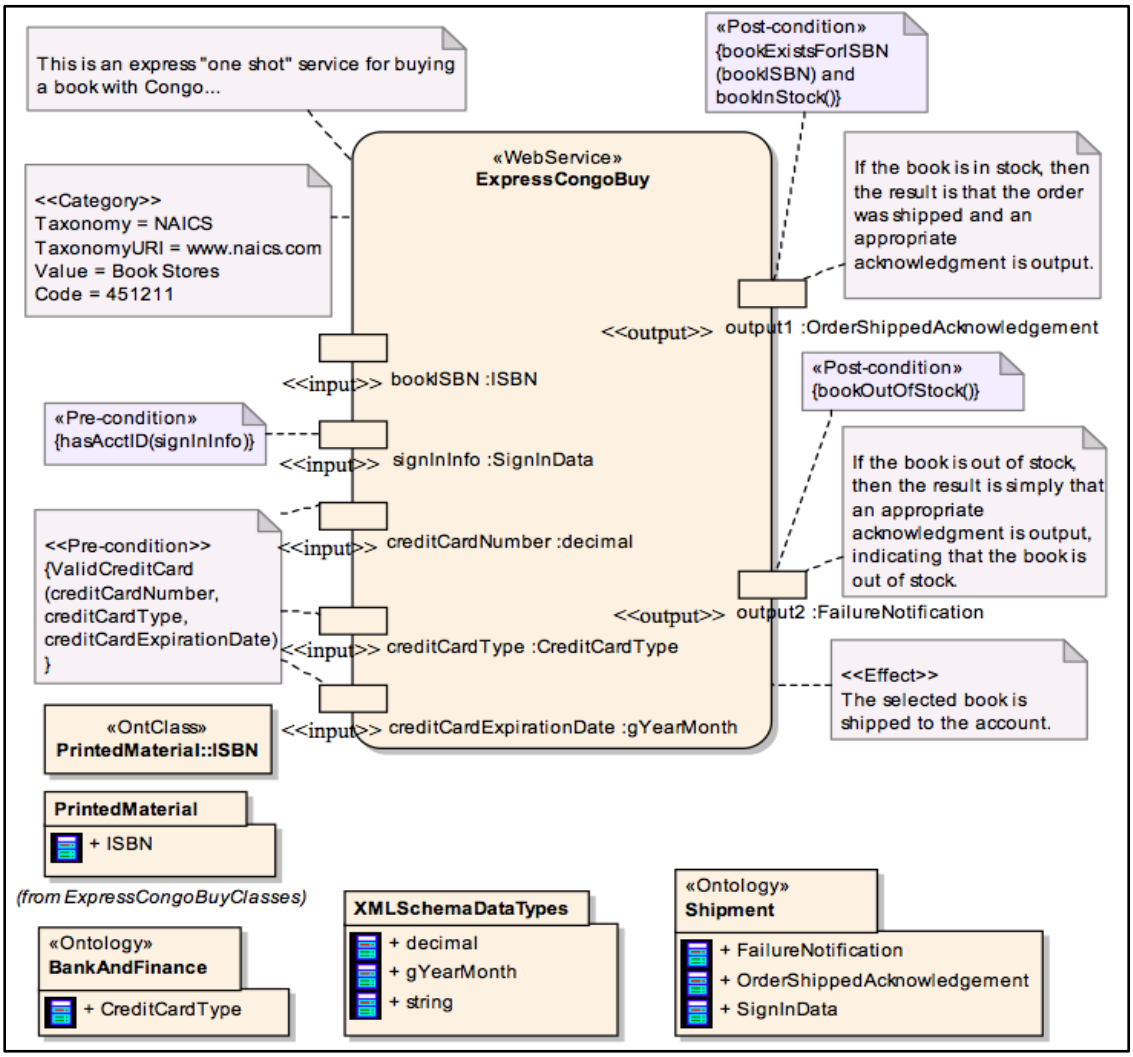


Meta modelo para la descripción de aspectos funcionales

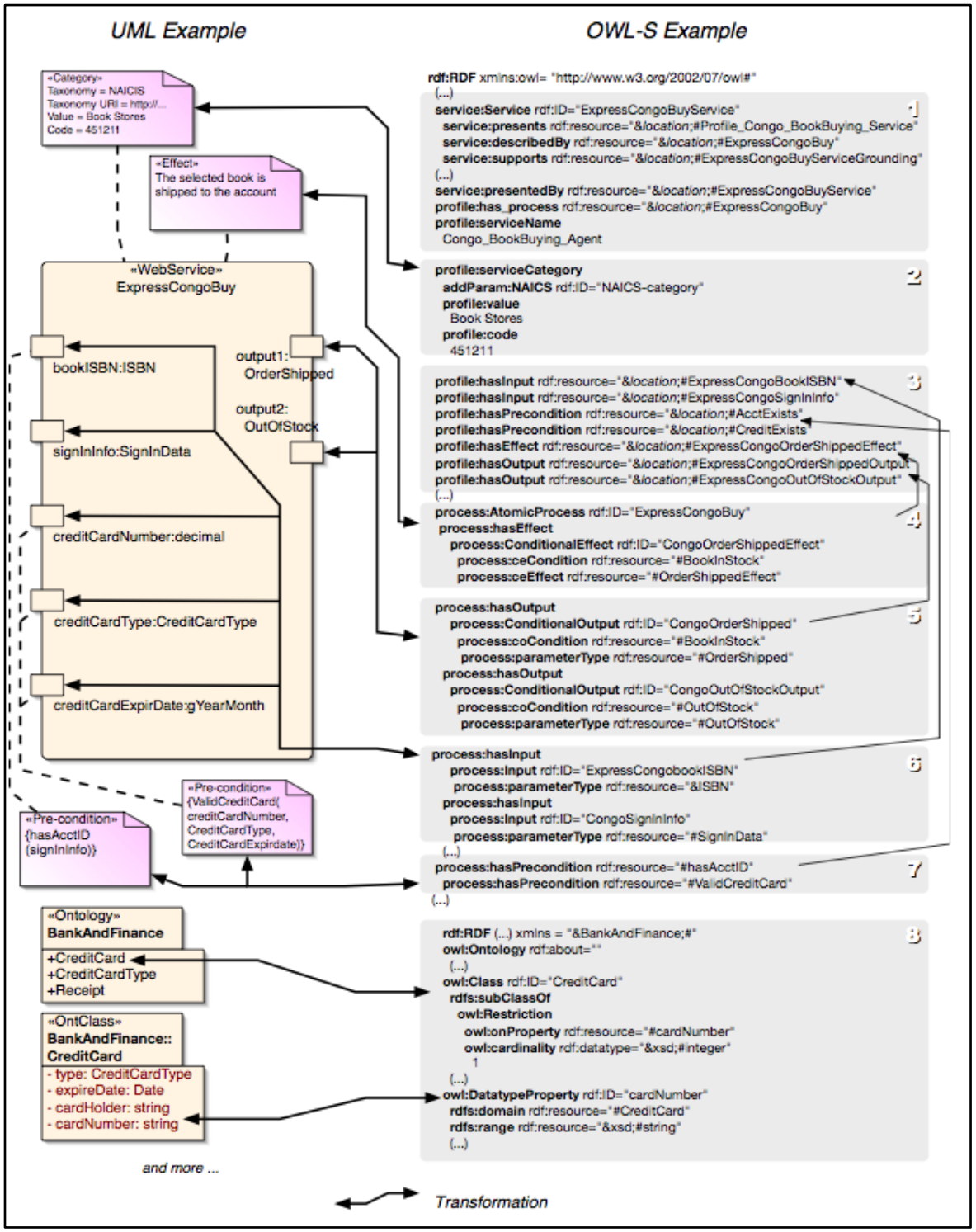


D. Ejemplos del perfil UML propuesto por [13]

Ejemplo de un servicio web semántico representado en el perfil UML.



Vista esquemática de las transformaciones entre UML y OWL-S



Bibliografía

[1] Cesar Javier Acuña and Esperanza Marcos Marcos, "Modeling semantic web services: A case study," in *ICWE '06 Proceedings of the 6th international conference on Web engineering*, Palo Alto, California USA, 2006, pp. 32-39.

[2] Fatima-Zahra Belouadha, Hajar Omrana, and Ounsa Roudiès, "A model-driven approach for composing SAWSDL semantic Web services," *International Journal of Computer Science Issues*, vol. 7, no. 2, pp. 7-15, March 2010.

[3] Djamel Amar Bensaber and Mimoun Malki, "Development of Semantic Web Services: Model Driven Approach," in *Proceedings of the 8th International Conference on New Technologies in Distributed Systems*, Lyon, France, 2008, pp. 40:1 - 40:11.

[4] Marco Brambilla, Jordi Cabot, and Manuel Wimmer, *Model-Driven Software Engineering in Practice*. California, CA, USA: Morgan & Claypool, 2012.

[5] Marco Brambilla et al., "Model-driven design and development of semantic Web service applications," *ACM Transactions on Internet Technology*, vol. 8, no. 1, p. 3, November 2007.

[6] Mark Burstein et al. (2004, November) OWL-S: Semantic markup for web services. [Online]. http://www.w3.org/Submission/OWL-S

[7] Marco Crasso, Juan Manuel Rodriguez, Alejandro Zunino, and Marcelo Campo, "Revising WSDL Documents: Why and How," *Internet Computing, IEEE*, vol. 14, no. 5, pp. 48-56, September 2010.

[8] Jos De Bruijn et al. (2005, June) Web Service Modeling Ontology (WSMO). [Online]. http://www.w3.org/Submission/WSMO/

[9] D.I De Silva, N Kodagoda, and H Perera, "Applicability of three complexity metrics," in *The International Conference on Advances in ICT for Emerging Regions - ICTer 2012:* , Colombo, Sri Lanka, 2012, pp. 82 - 88.

[10] Jing Dong, Yongtao Sun, and Yajing Zhao, "Hierarchical Composition of OWL-S Web Services," in *Sixth International Conference on Software Engineering Research,*

Management and Applications, 2008. SERA '08., Praga, 2008, pp. 187-194.

[11] Le Duy-Ngan, Ong Cheung Jye Desmond, and Goh Angela Eck Soong, "Converting WSMO to OWL-S system," in *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, Singapore, 2009, pp. 7-12.

[12] Dragan Gašević, Dragan Djurić, and Vladan Devedžić, "MDA-based Ontology Infrastructure," *Computer Science and Information Systems - ComSIS*, vol. 1, no. 1, pp. 91-116, February 2004.

[13] Roy Grønmo, Michael C. Jaeger, and Hjørdis Hoff, "Transformations Between UML and OWL-S," in *Proceedings of the First European Conference on Model Driven Architecture: Foundations and Applications*, Nuremberg, Germany, 2005, pp. 269-283.

[14] Thomas Robert Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition - Special issue: Current issues in knowledge modeling*, vol. 5, no. 2, pp. 199-220, June 1993.

[15] Martin Hepp. (2013, March) Accommodation Ontology Language Reference. Accommodation Ontology Metadata. [Online]. http://ontologies.sti-innsbruck.at/acco/ns.html

[16] Jin Hyuk Yang and In-Jeong Chung, "Automatic Generation of Service Ontology from UML Diagrams for Semantic Web Services," in *The Semantic Web – ASWC 2006*, vol. 4185, Beijing, China, 2006, pp. 523-529.

[17] Alaeddin Kalantari, Suhaimi Ibrahim, and Hamed Taherdoost, "A categorization of model-driven approaches for developing semantic Web Service," in *3rd International Conference on Data Mining and Intelligent Information Technology Applications (ICMiA), 2011*, pp. 395-404.

[18] Anneke Kepple, Jos Warmer, and Wim Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*, Anneke Kepple, Ed. Boston, MA, USA: Pearson Education, 2004.

[19] Il-Woong Kim and Kyong-Ho Lee, "A Model-Driven Approach for Describing Semantic Web Services: From UML to OWL-S," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 39, no. 6, pp. 637 - 646, November 2009.

[20] Ruben Lara, Dumitru Roman, Axel Polleres, and Dieter Fensel, "A Conceptual Comparison of WSMO and OWL-S," in *Lecture Notes in Computer Science*, vol.

3250, Erfurt, Germany, 2004, p. 254 269.

[21] Florian Lautenbacher and Bernhard Bauer, "Creating a Meta-Model for Semantic Web Service Standards," in *Proceedings of the Third International Conference on Web Information Systems and Technologies*, Barcelona, Spain, 2007, pp. 376-381.

[22] Stephen J Mellor, Anthony N Clark, and Takao Futagami, "Model-driven development: guest editors' introduction," *IEEE Software*, vol. 20, no. 5, pp. 14-18, September 2003.

[23] OMG, "MDA Guide Version 1.0," OMG, New York, NY, USA, ISBN, 2003.

[24] Claus Pahl, "Semantic model-driven development of web service architectures," *International Journal of Web Engineering and Technology*, vol. 4, no. 3, pp. 386-404, July 2008.

[25] Yue Pan et al., "Model-driven ontology engineering," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Berlin, 2006, pp. 57-58.

[26] Luca Panziera, Matteo Luigi Palmonari, Marco Comerio, and Flavio De Paoli, "WSML or OWL? A Lesson Learned by Addressing NFP-based Selection of Semantic Web Services," in *Non Functional Properties and SLA Management (NFPSLAM 2010) workshop*, vol. 4, Ayia Napa, Cyprus, 2010.

[27] Peter Salhofer and Bernd Stadlhofer, "Semantic MDA for E-Government service development," in *Proceedings of the 45th Hawaii International Conference on System Sciences*, Maui, HI, USA, 2011, pp. 2189-2198.

[28] Weijun Sun, Shixian Li, Defen Zhang, and YuQing Yan, "A Model-Driven Reverse Engineering Approach for Semantic Web Services Composition," in *WRI World Congress on Software Engineering, 2009. WCSE '09*, vol. 3, Xiamen, China, 2009, pp. 101 - 105.

[29] John T.E Timm and Gerald C Gannod, "A model-driven approach for specifying semantic Web services," in *IEEE International Conference on Web Services, ICWS 2005*, vol. 1, Orlando, FL, USA, 2005, pp. 313 - 320.

[30] Victoria Torres, Vicente Pelechano, and Oscar Pastor, "Building Semantic Web Services Based on a Model Driven Web Engineering Method," in *Proceedings of the 2006 International Conference on Advances in Conceptual Modeling: Theory and Practice*, Tucson, Arizona, USA, 2006, pp. 173-182.

[31] Markus Völter et al., *Model-Driven Software Development: Technology, Engineering, Management.*, 2nd ed., John Wiley & Sons, Ed. New York: Wiley, 2006, vol. 1.

[32] Hai H Wang, Nick Gibbins, Terry Payne, Ahmed Saleh, and Jun Sun, "A Formal Model of Semantic Web Service Ontology (WSMO) Execution," in *13th IEEE International Conference on Engineering of Complex Computer Systems, 2008. ICECCS 2008*, Belfast, 2008, pp. 111 - 120.