UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

ESCUELA DE CIENCIAS DE LA COMPUTACIÓN

TEMA:

ESTUDIO DE LOS SERVICIOS WEB SEMÁNTICOS

Tesis previa a la obtención del título de Ingeniero en Sistemas Informáticos y Computación

AUTOR:

Carlos Rubén Herrera Plascencia

DIRECTOR DE TESIS:

Ing. Jorge López Vargas

CO-DIRECTOR DE TESIS:

Ing. Nelson Piedra Pullaguari

LOJA – ECUADOR

Ing.

Jorge López Vargas
DIRECTOR DE TESIS

CERTIFICA:

Que el Sr. Carlos Rubén Herrera Plascencia, autor de la tesis “Estudio de los Servicios Web Semánticos”, ha cumplido con los requisitos estipulados en el Reglamento General de la Universidad Técnica Particular de Loja, la misma que ha sido coordinada y revisada durante todo el proceso de desarrollo desde su inicio hasta la culminación, por lo cual autorizo su presentación.

Loja, 3 de diciembre de 2009

- - - - - - - - - - - - - - -
Ing. Jorge López Vargas

DIRECTOR DE TESIS

CESIÓN DE DERECHOS

Yo, **Carlos Rubén Herrera Plascencia,** declaro ser autor del presente trabajo y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales.

Adicionalmente declaro conocer y aceptar la disposición del Art. 67 del Estatuto Orgánico de la Universidad Técnica Particular de Loja, que en su parte pertinente textualmente dice: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad”.

Carlos Rubén Herrera Plascencia

AUTORÍA

Las ideas, opiniones, conclusiones, recomendaciones y más contenidos expuestos en el presente informe de tesis son de absoluta responsabilidad del autor.

Carlos Rubén Herrera Plascencia

DEDICATORIA

La culminación de esta tesis la dedico con todo cariño.

A DIOS y mi principal intercesora ante EL, la Virgen María, por ser como siempre mi apoyo incondicional, invisible ante los ojos de todos, pero un incandescente impulso para el corazón que la necesita.

Nuevamente a DIOS por poner en mi camino cada tropiezo, cada tristeza y por darme la fortaleza de saber sobrellevarlos, por regalarme cada alegría y cada sueño que engrandecen este momento de vida.

A mis padres por ser la enseñanza perfecta para la vida “EL EJEMPLO”, por alejarme de las zarzas del camino fácil y permitirme conocer la grandeza de DIOS a través de la humildad del ser humano.

A mi hermano, por brindarme con su ejemplo un camino a seguir, por sus errores y sus aciertos que han sido de gran ayuda para mi fortalecimiento personal.

A mis amigos y maestros, que en dos palabras diferentes conjugan la generosidad, compartiendo sus enseñanzas y sus vivencias, haciendo de la Universidad la mejor experiencia.

A mi princesita, por su dedicación y paciencia, por hacer realidad con su presencia mis más grandes anhelos.

Carlos Herrera

AGRADECIMIENTO

Mi agradecimiento a todas las personas que han contribuido de una u otra manera para la culminación de este proyecto; en especial al Ing. Jorge López y el Ing. Nelson Piedra quienes con su dirección, apoyo y atención han sabido enfocar este proyecto, y culminar con éxito el presente trabajo de investigación.

Carlos Herrera

ÍNDICE DE CONTENIDOS

RESUMEN

La presente investigación expone un Estudio de los Servicios Web Semánticos, así como las metodologías tradicionales para su desarrollo y la exposición concreta de las nuevas tendencias, las mismas que nos permiten complementar ambas áreas.

Se inicia con una introducción general de los conceptos relacionados a la Web Semántica y los Servicios web de manera separada y su correspondiente "Estado del Arte", para de esta manera exponer las bases fundamentales que han servido como punto de partida a la presente investigación.

Dentro de estos conceptos, se exponen las diferentes metodologías para el mejoramiento de la Web actual y su transformación en una Web Semántica enriquecida de conceptos e información estructurada que nos permita realizar búsquedas más eficaces al poseer una información organizada mediante ontologías. De forma paralela, se exponen las metodologías de desarrollo de los Servicios Web, herramientas para su organización, clasificación y ubicación en directorios, así como, los lenguajes utilizados para la descripción de sus métodos, entradas y salidas.

Se exponen también los avances desarrollados actualmente para integrar los conceptos Semánticos con la Web actual, diferentes tipos de lenguajes, metalenguajes, repositorios y servicios que cubren esta carencia de conceptos bien definidos; permitiendo mejorar la información pura de la web actual hacia una información bien organizada, descrita y de fácil accesibilidad.

Una vez expuestos todos estos conceptos principales y sus proyecciones a futuro, la investigación se centra en el Estudio de los Servicios Web Semánticos, iniciando con la descripción de su ciclo de vida y los pasos que lo conforman; las metodologías, lenguajes y criterios tradicionales, así como, el estudio de varios frameworks que permiten trabajar los pasos expuestos en el ciclo de vida.

Continuando con el estudio se realizarán las comparaciones y análisis fundamentado de los frameworks que nos permitan alcanzar los objetivos propuestos en la presente investigación, se realizará un reconocimiento y documentación del lenguaje que lo integra, así como sus capacidades, limitaciones y herramientas adicionales.

Caso seguido, mediante la utilización del Framework seleccionado se realizará un escenario de pruebas que involucra la creación de ontologías, servicios web y objetivos (metas) que deseamos lograr con dichos servicios; mediante la implementación de este escenario se realiza

una comparación fundamentada entre la metodología tradicional (sintáctica) y la metodología investigada (semántica) con el propósito de demostrar el objetivo general de la presente Tesis.

Finalmente, se realiza una exposición de varias alternativas, líneas de investigación y trabajos futuros relacionados con el framework sugerido dando por terminada la presente investigación con las conclusiones y recomendaciones del caso.

ORGANIZACIÓN DE LA MEMORIA

El desarrollo de la presente tesis está estructurado en cuatro fases. **En la primera fase** se exponen los principios básicos que componen el fundamento teórico para el Estudio de los Servicios Web Semánticos mediante el Estado del Arte de la Web Semántica y los Servicios Web.

También se realiza una explicación sobre los diversos avances, metodologías, lenguajes y criterios que han sido utilizados para lograr medianamente la integración de ambos conceptos, incluyendo entre estos, los lenguajes y principios que han sido aceptados por el World Wide Web Consortium.

La segunda fase, se centra en exponer los conceptos y principios que definen a los Servicios Web Semánticos mediante una explicación de las fases que conforman el Ciclo de Vida de los Servicios Web Semánticos.

Incluidas también en esta fase están una exposición de los principales frameworks para el trabajo con Servicios Web Semánticos, un análisis de los mismos y los criterios de selección utilizados para determinar la herramienta más adecuada acorde a los objetivos del proyecto; una explicación de los conceptos, principios y sintaxis del lenguaje que sirve de base para el trabajo con la herramienta seleccionada, para finalmente realizar una incursión en la utilización del framework y herramientas WSMO.

En la tercera fase se realiza la implementación de un escenario de pruebas conformado por Ontologías, Servicios Web y Metas que nos permitirán realizar la comparación entre las metodologías tradicionales y los resultados mejorados que podemos obtener mediante la utilización del framework WSMO; realizando en esta fase una discusión fundamentada de los logros obtenidos mediante la implementación de descripciones semánticas en los Servicios Web.

Adicionalmente se expone de forma general las líneas de investigación y proyectos en desarrollo, relacionados con la metodología WSMO y las expectativas para el futuro; finalmente, se presentan las conclusiones y recomendaciones obtenidas de la investigación.

INTRODUCCION

El notable crecimiento de la información encontrada en Internet y las diferentes áreas del conocimiento expuestas públicamente, foros, blogs, páginas publicitarias, artículos científicos, etc. dificultan la búsqueda precisa de una información determinada, pues, en su mayoría la información que se encuentra publicada no ha sido sometida a una categorización previa; es decir, solamente se realizaba la creación de un documento HTML o XHTML y su posterior publicación, sin existir previamente una organización que permita ubicar la información dentro de una categoría específica.

Tras el éxito alcanzado por Internet, la gran mayoría de empresas, organizaciones, colegios, universidades y personas particulares han expuesto información que ha su criterio es de gran valía; sin embargo, al momento de realizar la búsqueda de un tema específico (Ej.: Servicio) los mejores buscadores – basados principalmente en la coincidencia de palabras – nos devuelven una gran cantidad de enlaces hacia páginas conteniendo dicha palabra, lo cual de ninguna manera garantiza que sea una información valiosa para el investigador, pues, puede llevarnos a sitios publicitarios, foros de preguntas, organizaciones, etc. provocando una gran pérdida de tiempo al momento de realizar una investigación importante basada en contenido web.

Es este, uno de los puntos principales que dan cabida a la presente Tesis, pues actualmente se intenta dar solución a la falta de organización y significado de la información, a través de la Semántica con lenguajes, técnicas y tecnologías variadas que se exponen en los primeros puntos de esta investigación.

El segundo aspecto principal que se trata son los Servicios Web, pequeñas o grandes aplicaciones que exponen y comparten sus métodos con el propósito de lograr aplicaciones robustas basadas en una inteligencia colectiva a través de Internet; mediante la utilización de los Servicios Web los desarrolladores podrán centrarse en ***qué*** objetivos desean alcanzar y no en el ***cómo*** implementar el código para lograr dichos objetivos.

Ejemplos concretos de Servicios Web son el buscador de Google, que permite realizar búsquedas dentro del sitio web que lo implemente, Google Maps para la localización específica de un lugar permitiendo adicionar información e imágenes relacionadas, Amazon Web Service que ofrece toda la infraestructura de su tienda virtual; estos y muchos más Servicios Web pueden ser incorporados (manualmente) en nuestros sitios mediante el manejo de una API compartida por la empresa creadora del Servicio Web.

La utilización de los Servicios Web se ha venido realizando satisfactoriamente durante los últimos años, sin embargo, con el prominente enriquecimiento de la información a través de la

Semántica nace una nueva tendencia; los Servicios Web Semánticos, que pretenden lograr la cooperación de varios Servicios Web de manera automática, es decir, realizar el descubrimiento y utilización del mejor Servicio Web basándose en sus descripciones semánticas y ontologías con el objetivo de alcanzar una meta determinada.

Un ejemplo teórico claro de los Servicios Web Semánticos se expone mediante una “Agencia Virtual de Viajes” cuyo objetivo es informar al solicitante las vacantes de transporte, hospedaje, renta de vehículos desde su lugar de origen hasta su lugar de destino; para ello, deberá interaccionar (automáticamente) con Servicios Web de Hoteles, Líneas Aéreas y Rent a Car otorgando como parámetros de entrada fecha y hora de partida, esperar la confirmación por parte del Servicio Web, realizar la reservación y mediante un último Servicio Web realizar los cobros respectivos por tarjeta de crédito y subdividir los montos correspondientes a cada empresa, todo esto de manera transparente para el usuario.

Con la finalidad de ofrecer una guía para futuros trabajos realizados en este tema, se ha investigado estándares sugeridos por el W3C para el desarrollo de la Web Semántica, así como las tecnologías actuales para el desarrollo de los Servicios Web; centrándonos finalmente en el tema de la Tesis el “Estudio de los Servicios Web Semánticos” y mostrar de una manera concreta como se pueden alcanzar los diferentes objetivos.

OBJETIVOS

Debido a los intereses generalizados de investigación y desarrollo en el campo de los Servicios Web Semánticos, se ha tomado un mayor campo de investigación que la Semántica aplicada al contenido web y los Servicios Web por separado, con este propósito se plantean los siguientes objetivos:

Objetivo General

Implementar un caso de estudio que permita realizar el descubrimiento de Servicios Web tanto con metodologías semánticas (Servicios Web Semánticos) como con metodologías sintácticas.

Objetivo Específicos

- Investigar a través de varias fuentes sobre el Estado del Arte, la aplicación e implementación de Servicios Web, Web Semántica y los Servicios Web Semánticos.
- Investigar el alcance, cobertura, efectividad y confiabilidad de las diferentes herramientas que permiten el trabajo con Servicios Web Semánticos.
- Determinar una herramienta y metodología que ofrezca mejores alcances en el modelado e implementación de los Servicios Web Semánticos.
- Determinar la efectividad de las metodologías actuales en el descubrimiento de Servicios Web Semánticos.
- Realizar una comparación entre el descubrimiento de Servicios Web de manera semántica y sintáctica.

RESULTADOS ESPERADOS

Una vez concluido el presente proyecto de investigación, se espera determinar una mejora sustentable obtenida al complementar los Servicios Web con la Web Semántica y Ontologías, una herramienta adecuada para el trabajo con Servicios Web Semánticos, así como una muestra sustancial del mejoramiento en el descubrimiento de Servicios Web.

Para ello, se realizará un modelo ontológico que permita demostrar de manera general la utilización de Ontologías para el mejoramiento en la clasificación y descripción de la información.

Adicionalmente, se realizará la descripción sintáctica y semántica (por separado) de varios Servicios con el propósito de comparar el descubrimiento de Servicios Web con ambas tendencias, mostrando de manera tangible las mejoras obtenidas con la utilización de ontologías.

FASE 1

CAPITULO 1:
ESTADO DEL ARTE DE LOS SERVICIOS WEB Y WEB SEMÁNTICA

1.1. INTRODUCCIÓN

En la actualidad los mayores avances en el desarrollo de los Servicios Web están centrados en la metodología SOAP, que es un protocolo estándar que define como los objetos en diferentes procesos pueden comunicarse, esta metodología ha venido evolucionando de tecnologías para aplicaciones distribuidas como COM, CORBA, EJB hasta llegar a establecerse como recomendación W3C.

También, se está incursionando notablemente en el desarrollo de Servicios Web basado en la arquitectura REST que ha tenido sus inicios en el año 2000 con base en el uso de las URI's sobre el protocolo HTTP para intercambio de información.

Para tener una idea más amplia sobre la situación actual de los Servicios Web así como las tecnologías para su desarrollo considero necesario realizar una breve revisión acerca de los aspectos más relevantes.

1.2. LOS SERVICIOS WEB[1]

Con el avance en tecnologías de telecomunicaciones y el intercambio de información a través de las redes informáticas, se han desarrollado metodologías que facilitan este tipo de intercambio como son las RPC, sin embargo, ésta metodología no presta las facilidades y seguridades necesarias para el intercambio de información entre plataformas y lenguajes de programación distintos.

Con el surgimiento del lenguaje XML se ha podido superar fácilmente las restricciones inmersas en la utilización de diferentes plataformas y lenguajes, facilitando el intercambio de información entre aplicaciones.

Los Servicios Web están conformados por un conjunto de aplicaciones y protocolos como SOAP y REST que permiten la interoperabilidad entre ellos, existen los Servicios Web como proveedores de información, así como los Servicios cliente que son aquellos que consumen la información; generalmente, para un correcto intercambio de mensajes se utiliza estructuras de

1 La información expuesta ha sido resumida de **[1] [2]** y **[3]** donde se tiene mayor detalle

datos en lenguaje XML bajo el protocolo SOAP y el contrato o reconocimiento entre cliente y proveedor se lo realiza mediante el lenguaje WSDL. Un diagrama del funcionamiento de los Servicios Web se muestra en la Figura 1.

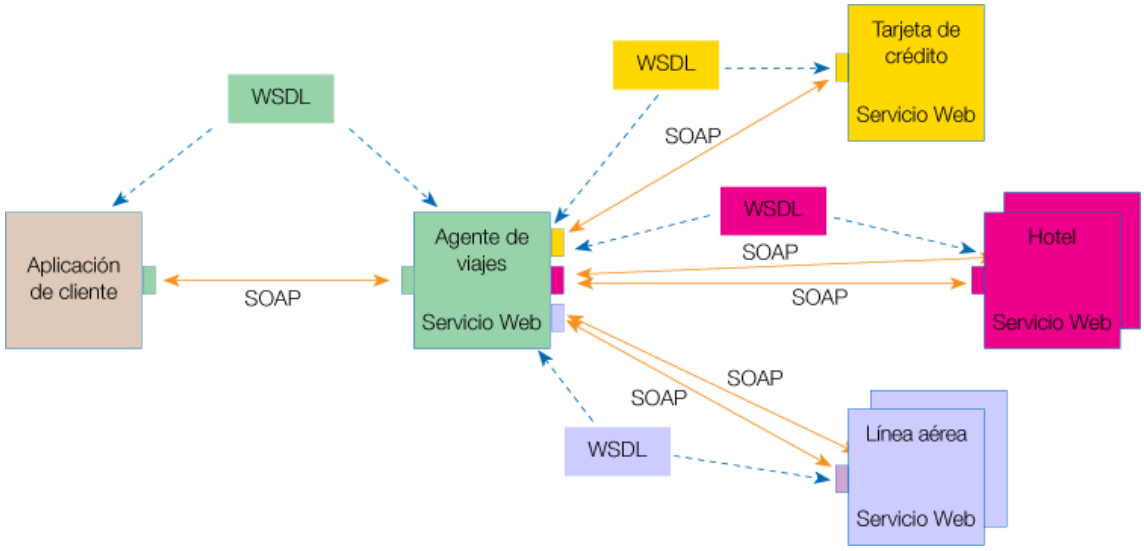


Figura 1. Funcionamiento de los Servicios Web[2]

Básicamente un Servicio Web intercambia información con otra aplicación o Servicio utilizando Internet como plataforma, el intercambio de información se basa en la utilización de mensajes XML que pueden ser interpretados por todos los lenguajes de programación.

Para la utilización de los Servicios Web y el intercambio de mensajes, se ha investigado la metodología SOAP que actualmente es un estándar W3C y la metodología REST o RESTful que se ha venido tratando y mejorando los últimos años, una explicación del funcionamiento de estas metodologías se trata en las siguientes secciones.

1.2.1. METODOLOGÍAS DE DESARROLLO

1.2.1.1. Metodología SOAP

SOAP es un protocolo para el intercambio de mensajes entre redes de computadores utilizando HTTP, se basa en el intercambio de mensajes XML pues permiten almacenar información más detallada y con una estructura definida facilitando su interpretación por parte de seres humanos y máquinas.

2 **Funcionamiento de los Servicios Web:** Ejemplo tomado de una agencia de viajes virtual **[1]**

Para el desarrollo de Servicios SOAP se requiere de la implementación de una arquitectura que consta de:

- Servicios Web
- Estándar SOAP para el manejo de mensajes
- Esquemas XML definidos
- Base de datos

La creación de los Servicios Web no es más que la implementación del código fuente de un programa o tarea definida por el proveedor sobre un conjunto de información que podría ser una Base de Datos, adicionalmente existe el documento descriptivo WSDL que contiene la información de métodos y parámetros ofrecidos por el Servicio Web.

La utilización de dicho Servicio implica la creación de un Servicio Web cliente que se conecte con el Servicio del proveedor y conozca la correcta utilización de los métodos expuestos en el documento WSDL, una vez implementados ambos Servicios se requiere del intercambio de mensajes.[3]

La estructura de un mensaje SOAP se muestra en la figura 2, mediante el cual se realizará la comunicación entre los Servicios (Cliente y Proveedor), la información del mensaje indica al Servicio las acciones solicitadas, tanto las acciones de petición y respuesta llevan la información necesaria para que exista la interacción entre Cliente y Proveedor.

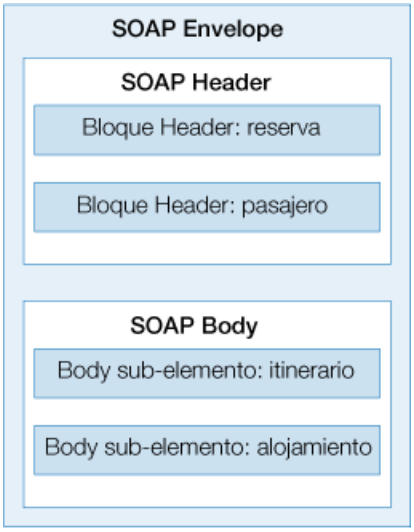


Figura 2. Estructura de los mensajes SOAP[4]

3 Algunas definiciones importantes pueden ampliarse en **[4]**
4 **Estructura SOAP**: Figura tomada de **[1]**

Un ejemplo de la sintaxis de los mensajes SOAP se muestra mediante la figura 3, que detalla un mensaje que ha sido tomado del repositorio de Servicios Web jUDDI durante la petición de búsqueda de un Servicio Web almacenado en dicho repositorio.

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body>
      <find_business maxRows="100" generic="2.0" xmlns="urn:uddi-org:api_v2">
        <findQualifiers>
           <findQualifier>***</findQualifier>
        </findQualifiers>
        <name>*****</name>
        <discoveryURLs>
           <discoveryURL>***</discoveryURL>
        </discoveryURLs>
        <identifierBag>
           <keyedReference tModelKey="***" keyName="***" keyValue="***" />
        </identifierBag>
        <categoryBag>
           <keyedReference tModelKey="***" keyName="***" keyValue="***" />
        </categoryBag>
        <tModelBag>
           <tModelKey>***</tModelKey>
        </tModelBag>
      </find_business>
    </soapenv:Body>
</soapenv:Envelope>
```

Figura 3. Mensaje SOAP[5]

La utilización de los métodos involucrados en un Servicio requiere del conocimiento de los parámetros de entrada y la información que se reciba como salida, en algunos casos, el documento WSDL define como parámetros de entrada tipo String, Integer, Float, sin embargo, existen métodos que requiere información estructurada en XML.

Tanto la información requerida como entrada y el resultado de los métodos puede ser expresada en XML, para ello se hace uso de los esquemas XML (XML Schema) que son plantillas definidas que permiten la comprensión de la información al manejar una misma estructura.

Los mensajes SOAP son independientes del sistema operativo, y pueden transportarse en varios protocolos de Internet como SMTP, MIME y HTTP. Para el transporte de los mensajes a través de Internet SOAP provee un mecanismo estándar de empaquetado de mensajes que se compone de un *sobre* (`envelope`) que contiene el cuerpo del mensaje más cualquier información de cabecera que se utiliza para describir el mensaje. **[1][2][4]**

1.2.1.2. Metodología REST

Tiene sus orígenes en el año 2000 en la tesis doctoral sobre web escrita por Roy Fielding uno de los principales autores de la especificación HTTP. REST es una técnica de arquitectura de

5 **Mensaje SOAP:** Tomado de las peticiones de búsqueda de servicios a través de la plataforma Apache jUDDI tratada la sección 6.1.1.

software para sistemas distribuidos que utilicen una interfaz XML y HTTP sin la necesidad de utilizar patrones para el intercambio de mensajes como se realiza en SOAP.

Los sistemas que utilizan los principios REST se conocen como REST*ful,* aunque la arquitectura REST aún no se ha tomado como estándar basa su trabajo en estándares como son el protocolo HTTP y el lenguaje XML.

Principalmente la metodología REST utiliza las URI's para el manejo de información, siendo cada URI un objeto diferente, de tal forma puede accederse a los objetos únicamente a través de su jerarquía superior de enlaces, permitiendo un fácil manejo de sesiones y estados sin necesidad de intercambiar constantemente mensajes XML.[6]

El uso de XML y HTML facilita la navegación entre los distintos recursos siguiendo enlaces que no requieren una infraestructura adicional. De la misma manera, toma las operaciones HTTP definidas para ser aplicadas a cada uno de los recursos, estas operaciones son:

- **GET:** Se utiliza para recuperar información de una URL conocida, es similar al método SELECT del lenguaje SQL.
- **POST:** Similar al método INSERT, es utilizado para insertar información en una URL desconocida, es decir, junto con este método puede crearse la URL necesaria.
- **PUT:** Permite la actualización de la información en una URL conocida, la función es similar al método UPDATE de SQL.
- **DELETE:** Utilizado para eliminar información de una URL conocida, el método es similar a DELETE de SQL.

Para la descripción de Servicios REST no existía un lenguaje estructurado similar a WSDL, por lo cual se utilizaba de manera análoga WADL un lenguaje no estandarizado, pero a partir de Junio de 2007 el consorcio W3C publicó la recomendación WSDL 2.0 que ha sido pensada para la descripción de Servicios Web basados en SOAP y REST. **[5][6]**

La información de los métodos en el documento descriptivo nos relaciona con las URL's asignadas a los recursos disponibles, y la utilización de los métodos puede ser incluida en las páginas que mostrarán dicha información, este es el caso de Google Maps, que permite manejar métodos Java Script con la finalidad de interactuar con un mapa incluido en el documento web.

La estructura del código provisto para hacer uso de los diferentes Métodos se muestra a continuación en la figura 4, donde se resalta la invocación de los métodos.

6 La descripción general de la Metodología REST puede ser ampliada en **[6]** Pág. 49 - 81

```
<script type=”text/javascript”
 src=”http://www.google.com/jsapi?key=ABCDEFG”>
</script>
    <script type=”text/javascript”
      google.load(“maps”, “2.x”);
//call this function when the page has been loaded
 function initialize (){
 var map = new google.maps.Map2(document.getElementById(“map”));
 map.setCenter(new google.map.LatLng(37.4419, -122.1419),13);
}
google.setOnloadCallback(initialize);
</script>
```

Figura 4. Código servicio REST[7]

1.2.2. DESCRIPCIÓN Y PUBLICACIÓN

Inicialmente las tecnologías como RPC y CORBA fueron utilizadas para realizar software que permitiera compartir ciertos procesos e información dentro de una misma empresa, estos paquetes de software podían fácilmente distribuirse, pues, al ser utilizados bajo una misma empresa no se requería de contratos adicionales para su utilización, así como, los desarrolladores se encontraban al tanto de los métodos y procesos que dichos paquetes utilizaban.

Con el surgimiento de los Servicios Web este tipo de distribución se complica, pues, la información que se comparte debe poseer un contrato y registrarse en servidores de acceso público donde se podrán exponer sus métodos, parámetros de entrada/salida, interfaces y las API’s para su correcta utilización.

Para tener un conocimiento más amplio del manejo de descripción y publicación de Servicios se detallan estos conceptos en las siguientes secciones.

1.2.2.1. Descripción

El conocimiento exacto de un Servicio Web es imprescindible para poder interactuar con el mismo, es por ello que, es necesaria la utilización del lenguaje WSDL para realizar la descripción sintáctica detallada de los servicios desarrollados bajo metodología SOAP, sin embargo, como se ha tratado anteriormente está en auge la metodología REST que no había sido considerada dentro de la especificación WSDL 1.1 por lo cual, en primera instancia se desarrollo el lenguaje WADL para realizar la descripción de los Servicios Web basados en REST; es por esta razón que el W3C concreta la versión WSDL 2.0 que involucra nuevas características que facilitan la definición de los Servicios Web REST para tomar dicho lenguaje como estándar.

Este lenguaje basado en XML permite realizar la descripción de la interfaz pública de los Servicios Web, así como, la forma de comunicación, requisito de protocolos y el formato de

7 **Código REST**: Ejemplo tomado de Google Maps API, más información en [http://code.google.com/apis/maps/documentation/]

mensajes entrada/salida que permite la interacción con un determinado servicio; los documentos WSDL definen de manera sintáctica como se puede interactuar correctamente con un Servicio Web, por esta razón, un documento WSDL utiliza los siguientes elementos durante la definición de un servicio:

- **types:** Permite la definición del tipo de dato que utiliza como (*string*)
- **message:** Involucra una definición escrita de los datos que están siendo transmitidos.
- **operation:** Descripción sintáctica de una acción permitida por el servicio.
- **port:** Punto de comunicación definido por la combinación de un enlace y una dirección de red.
- **portType:** Conjunto de operaciones permitidas a través de un determinado puerto.
- **binding:** Especificación del protocolo y del formato de datos para un tipo de puerto determinado.
- **service**: Es la colección compuesta de puertos relacionados.[8]

La figura 5 muestra de manera gráfica el funcionamiento de un documento WSDL.

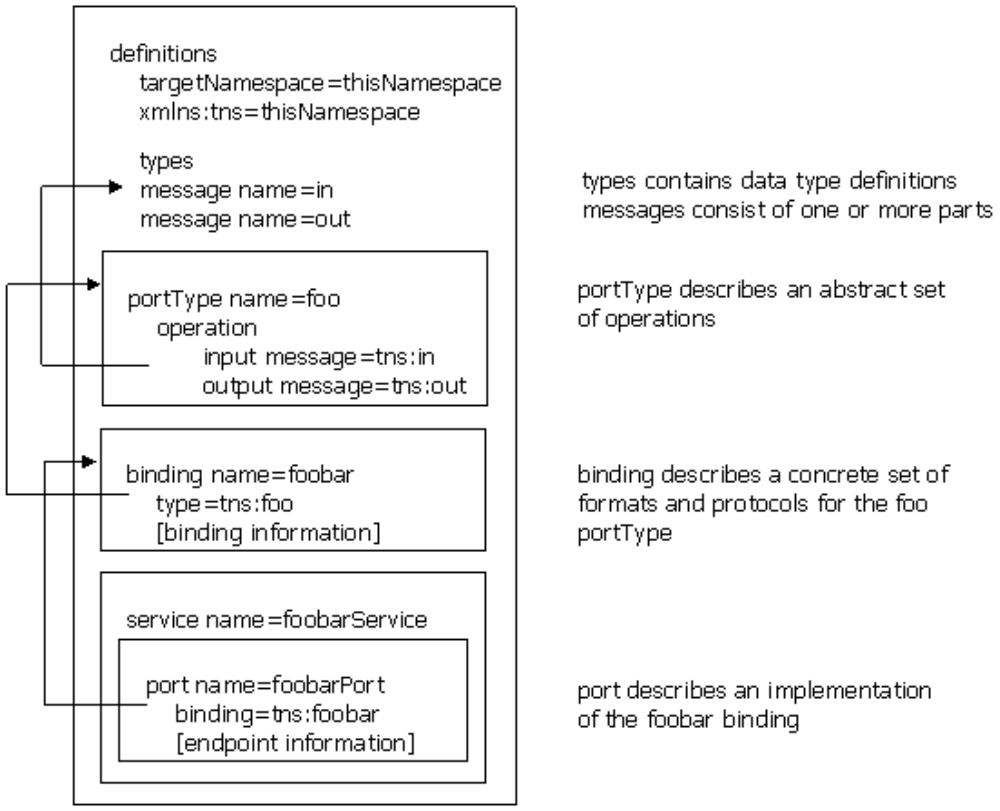


Figura 5. Esquema de documentos WSDL[9]

8 La descripción de los Servicios Web es tratada con mayor profundidad en la recomendación W3C del lenguaje WSDL 1.1 disponible en [http://www.w3.org/TR/wsd]

1.2.2.2. Publicación

La publicación de los Servicios Web se realiza a manera de Catálogo de servicios, donde se establece:

- Nombre del Servicio
- Proveedor
- Punto de entrada
- URL del contrato o documento descriptivo
- Protocolos aceptados
- Descripción general del Servicio

Con este objetivo, la empresa OASIS propone UDDI el cual, es un lenguaje basado en XML que nos permite realizar el registro de los Servicios Web como catálogo de servicios.

UDDI se ha conformado en uno de los estándares básicos para el trabajo con Servicios Web, es a su vez un framework independiente de la plataforma con características como:

- Permite almacenar las descripciones generales de los Servicios Web.
- Conforma un directorio para las interfaces relacionadas a los Servicios Web a través de la utilización de WSDL.
- La comunicación con su infraestructura se realiza a través de mensajes SOAP.
- Puede exponerse como repositorio público o local.

En otras palabras, UDDI provee una manera de localizar un servicio software, invocar el servicio y manejar los datos relacionados al servicio; un repositorio UDDI puede funcionar a su vez como un Servicio Web permitiendo la interacción con otros servicios a través de los mensajes SOAP. Las relaciones conceptuales entre UDDI y otros protocolos en el área de los Servicios Web se ilustran a través de la figura 6.[10] **[9]**

9 **Esquema WSDL**: tomado de[http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm]
10 La información generalizada sobre la publicación de servicios web puede ser ampliada en **[9]**

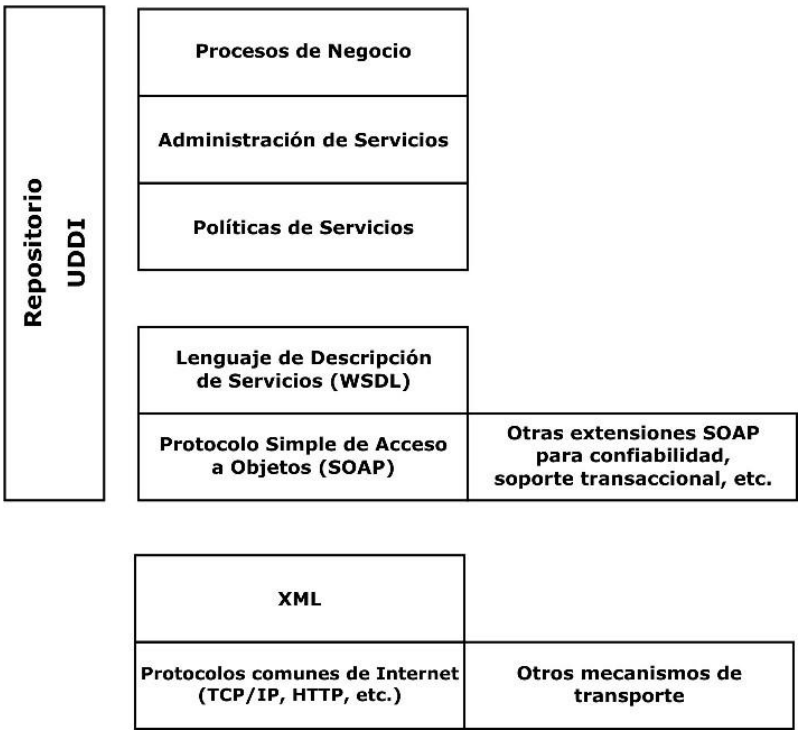


Figura 6. Esquema de trabajo UDDI[11]

1.3. LA WEB SEMÁNTICA

Actualmente Internet es una herramienta esencial para el desarrollo del conocimiento humano, sin embargo, el volumen de publicaciones es difícil de manejar y dificulta la búsqueda de información específica; los motores de búsqueda actuales basan sus resultados en coincidencias de palabras, y los resultados obtenidos no garantizan una relación directa con la búsqueda realizada.

La Web semántica se origina con la búsqueda de organización de la información bibliográfica, en catálogos tradicionales para la organización de material bibliográfico, con la descripción del documento, puntos de acceso (títulos, empresa, imprenta) con el fin de localizar más fácilmente la información física dentro de la biblioteca.

Dando con estos catálogos un inicio para el continuo mejoramiento en la búsqueda de información, existiendo varios niveles de descripción de contenidos, así como diferentes herramientas y metodologías más o menos formalizadas. De esta manera se citan varios niveles de descripción de información:

11 **Esquema de trabajo UDDI:** Adaptado de [http://uddi.xml.org/uddi-101]

- **Indización libre:** Se realiza mediante la generación de un listado o conjunto abierto de términos.

- **Indización controlada:** Realiza la identificación de los recursos por medio de un conjunto cerrado de términos.

- **Vocabularios controlados:** Es la existencia de un vocabulario de términos relacionados a un mismo tema con un significado concreto asignado.

- **Taxonomías:** Se utilizan para la clasificación de información a manera de árbol, organizándose los contenidos de manera jerárquica, así pues, las taxonomías adhieren una jerarquía a los vocabularios controlados. Un ejemplo claro es la clasificación de plantas y animales.

- **Tesauros:** Son taxonomías que adicionalmente poseen términos relacionados, pueden poseer varias jerarquías y notas añadidas para mejorar el significado de algunos términos. Los Tesauros ayudan a encontrar las palabras o frases precisas para realizar mejor las búsquedas.

- **Mapas temáticos:** Los mapas temáticos son la manera formal de declarar un conjunto de temas enlazados a documentos. De esta forma, se ayuda a los usuarios a encontrar información acerca de un tema a lo largo de una gran variedad de documentos.

- **Ontologías:** Las ontologías son tesauros avanzados que representan de forma conceptual unívoca un determinado dominio del conocimiento por medio de estructuras semánticas.[12]

Con la Web Semántica no solamente se procesarán los datos como entradas y salidas de información, sino en términos de su semántica (significado), por tanto, la Web Semántica conforma un camino claro para el razonamiento en la Web mejorando de esta manera las capacidades de Internet. De esta manera, se logra que las máquinas tengan la habilidad de resolver problemas bien definidos, a través de operaciones bien definidas que se llevarán a cabo sobre datos existentes bien definidos. **[14]**

1.3.1. ESTRUCTURA DE LA WEB SEMÁNTICA

La información expuesta en la web que hasta el momento ha sido publicada es solamente legible por seres humanos (HTML); por ello, surge la indispensable necesidad de fomentar el desarrollo de sitios web que permitan la interacción de software con dicha información. El esquema general de la Web semántica adopta las siguientes capas.

12 Estos breves conceptos se han tomado de [http://www.hipertexto.info/documentos/web_semantica.htm] donde se podrá obtener una información más detallada

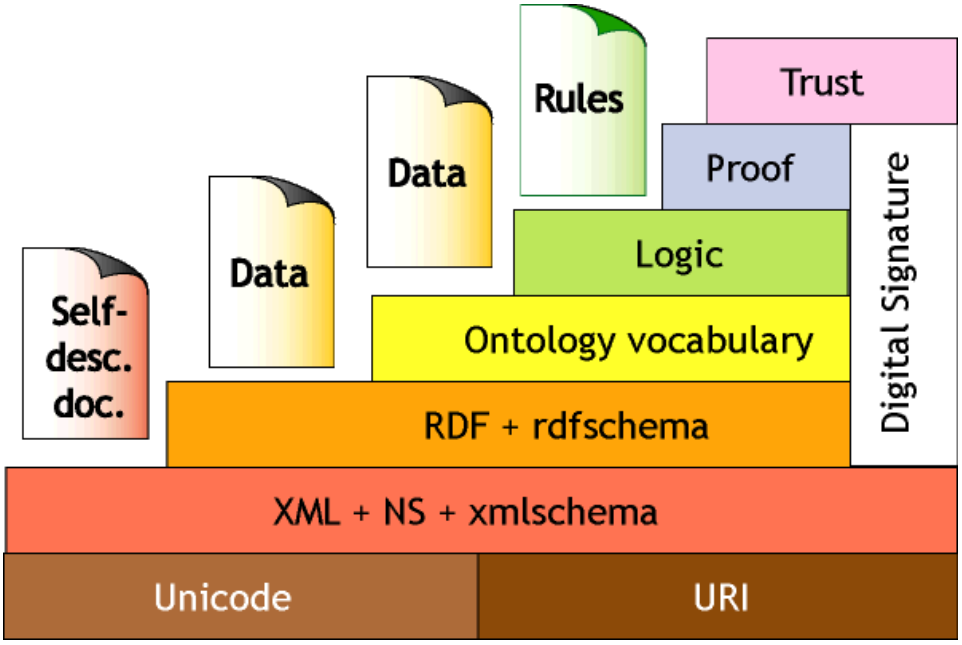


Figura 7. La web semántica[13]

Unicode: Codificación del texto para usar los símbolos de diferentes idiomas.

URI: Es el identificador único que permite localizar un recurso vía Internet. Se trata de la URL + URN.

XML + NS + Xml Schema: Permite la comprensión entre agentes software. XML ofrece un formato común para intercambio de información, NS sirve para organizar los elementos asociándolos con los espacios de nombre identificados por referencias URI y finalmente XML Schema permite estandarizar los documentos. De esta forma, aunque se utilicen diferentes fuentes, se crean documentos uniformes en un formato común y no propietario.

RDF + RDF Schema: Define un lenguaje universal para expresar ideas, conceptos y afirmaciones sobre los recursos permitiendo que los recursos puedan asociarse a una URI. Tanto esta capa como la anterior corresponden a la semántica (metadatos).

Vocabulario de Ontologías: Permite catalogar y clasificar la información describiendo los objetos y sus relaciones con otros objetos. Esta capa permite extender la funcionalidad de la Web Semántica, agregando nuevas clases y propiedades para describir los recursos.

13 Estructura de la web semántica propuesta por Tim Berners-Lee para la mejora de la información en Internet, tomada de: [http://www.hipertexto.info/documentos/web_semantica.htm]

Lógica: Precisa reglas de inferencia para que de esta manera un programa pueda realizar deducciones lógicas basándose en la información almacenada en capas anteriores.

Pruebas: Es necesario el intercambio de "pruebas" escritas en el lenguaje unificador, lenguaje que hace posible las inferencias lógicas de la Web Semántica.

Confianza: Los agentes software son muy escépticos acerca de lo que leen en la Web Semántica hasta que hayan podido comprobar de forma exhaustiva las fuentes de información.

Firma digital: Bloque encriptado de datos que serán utilizados por los ordenadores y los agentes software para verificar que la información adjunta ha sido ofrecida por una fuente específica confiable.[14]

1.3.2. METADATOS

Los metadatos son etiquetas HTML que permiten un fácil manejo y entendimiento, las mismas que son utilizadas dentro de los documentos de contenido HTML o XHTML, generalmente son añadidas en la cabecera del documento para agregar información ampliada referente al contenido del documento como:

- Autor
- Descripción
- Palabras clave
- Fecha de publicación, expiración y actualización de la información
- Sesión de derechos de autor
- Contenido del sitio
- Robots

Estas etiquetas ayudan a realizar una descripción más exacta del contenido del sitio web para que sea indexado y utilizado por los robots de los motores de búsqueda, indicándoles fechas importantes del documento, el tiempo de validez y actualización de la información.

La estructura de las etiquetas meta no tiene restricciones en cuanto a la información que esta etiqueta pueda o no incluir, pero las más extendidas y reconocidas por buscadores son las mencionadas anteriormente.

14 Conceptos que conforman la Web semántica, resumido de [10] [12] y [http://www.hipertexto.info/documentos/web_semantica.htm]

Su utilización es incorporada en la cabecera `<head>` del documento, donde la información que maneja es del tipo *propiedad:valor* a continuación se muestran unos ejemplos

<meta name=”author” **content=**”UTPL”>
<meta name="copyright" **content**="UTPL">
<meta name="robots" **content**="all,index,follow">[15]

1.3.3. LENGUAJE RDF

El lenguaje RDF es una recomendación W3C, se basa en el lenguaje XML por lo que se conoce como RDF XML, hace uso de diferentes etiquetas para conformar una sentencia que expresa un significado semántico haciendo uso de *tripletas* de la siguiente manera:

Sujeto (el recurso) – **predicado** (característica) - y **objeto** (valor) por ejemplo:
Sujeto (http://www.w3.org/home/lassila) – predicado (Creator) – objeto (Ora Lassila).

La sintaxis RDF básica toma la forma:

[1]**RDF**	::=[‘<*rdf*:RDF>’] description* [‘</*rdf:*RDF>’]
[2]**description**	::=[‘<*rdf*:Description’idAboutAttr?’>’ propertyElt* ‘</*rdf:*Description>’
[3]**idAboutAttr**	::= idAttr | aboutAttr
[4]**aboutAttr**	::=’about=” URI-reference”
[5]**idAttr**	::=’ID=”IDsymbol”
[6]**propertyElt**	::=’<’propName’>’ value ‘</’ propName ‘>’ | ’<’propName resourceAttr ‘/>’
[7]**propName**	::=Qname
[8]**value**	::= description | string
[9]**resourceAttr**	::=`resource=”URI-reference”
[10]**Qname**	::= [NSprefix ‘:’] name
[11]**URI-reference**	::= string, interpreted per [URI]
[12]**IDsymbol**	::=(any legal XML name symbol)
[13]**name**	::= (any legal XML name symbol)
[14]**NSprefix**	::= (any legal XML namespace prefix)
[15]**string**	::= (any XML text, with “<”, “>”, and “&” escaped)[16]

15 Ideas principales extraídas de **[17]** referirse a la fuente para ampliar dicha información
16 **Sintaxis general RDF**. Para mayor información: [http://www.w3.org/TR/1999/REC-rdf-syntax-19990222]

Donde el elemento RDF es utilizado como delimitador de un documento XML donde se incluirá el contenido de los diferentes significados semánticos y enlaces generados a partir de la información, el lenguaje RDF se describe más ampliamente en **[18]**

Para una mayor organización de la información semántica RDF implementa la utilización de vocabularios RDF-Schema que proporciona los elementos básicos para la descripción de información trabajando con clases y subclases dentro de un dominio específico del conocimiento. La figura 8 muestra un diagrama de las sentencias RDF relacionadas a la creación de un documento web.

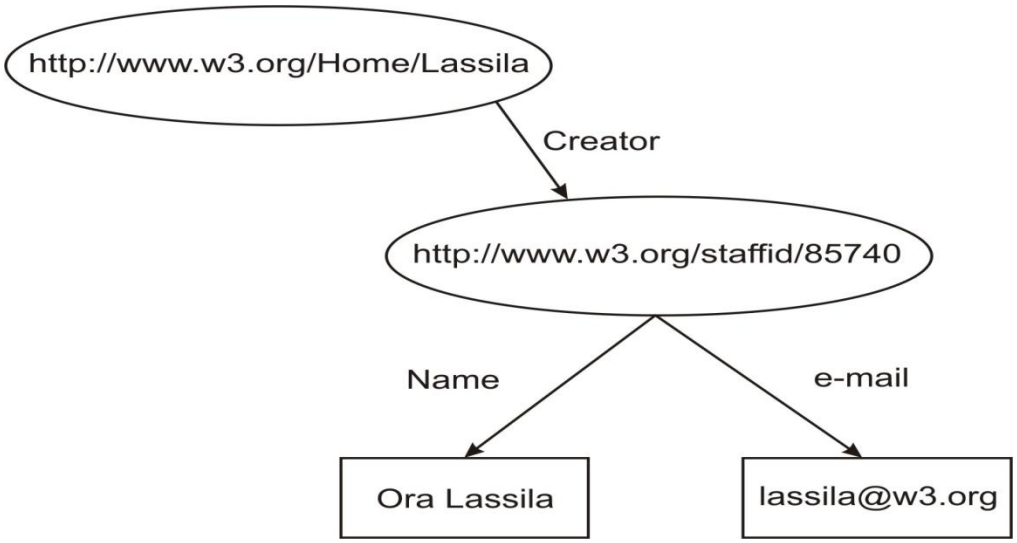


Figura 8. Diagrama RDF[17]

1.3.4. MICROFORMATOS

La utilización de los lenguajes RDF, RDF Schema y OWL involucran la necesidad de un conocimiento profundo de los mismos para realizar un manejo adecuado de la información ontológica, a esta complejidad se suma la existencia de medios como foros, blogs y wikis que permiten a los usuarios con poca o nula experiencia en la web semántica publicar información. De esta manera se sigue incrementando el volumen de información sin estructura, por ello se ha previsto la utilización de Microformatos.

Los Microformatos se utilizan dentro de las páginas HTML o XHTML, existen varios tipos de microformatos creados con la idea *Primero las personas, luego las máquinas* que permiten incluir información semántica a las publicaciones evitando la complejidad de lenguajes altamente estructurados como son RDF y OWL.

17 **Diagrama RDF:** Valor estructurado con identificador, mayor detalle en [http://www.w3.org/TR/1999/REC-rdf-syntax-19990222]

Este tipo de estructuras han sido añadidas a gestores de contenido social como Twitter, Plurk, Tanzania, Google Maps entre otros y están siendo reconocidos para la búsqueda en motores como Yahoo que da soporte a microformatos.

Dentro de los microformatos, los más representativos son: *vCard, hCalendar, hListing, hReview, hResume,* etc*.* **[19][20]**

1.3.5. LENGUAJE OWL

OWL es un lenguaje de marcado para publicar y compartir información utilizando ontologías, este lenguaje permite la construcción de modelos ontológicos basándose en XML, RDF y RDF Schema.

Facilita la interoperabilidad entre aplicaciones siendo independiente de la plataforma, permite la escalabilidad de la información y añade más vocabulario para describir propiedades y clases como unión, desunión, cardinalidad, igualdad, simetría, etc.

OWL proporciona tres sub lenguajes con diferentes niveles de complejidad acorde a las necesidades específicas de desarrolladores y usuarios; estos sub lenguajes son OWL Lite, OWL DL y OWL Full.

- **OWL Lite:** Diseñado para usuarios que requieres de una clasificación jerárquica y restricciones simples a manera de tesaurus y taxonomías como se ha visto en la sección 1.3.
- **OWL-DL:** Ofrece la máxima expresividad el lenguaje OWL permitiendo cardinalidad, restricciones, reusabilidad. Se denomina OWL-DL debido a su correspondencia con la Lógica de Descripciones (**D**escription **L**ogics, por sus siglas en inglés), un campo de investigación que estudia la lógica que compone la base formal de OWL.
- **OWL Full:** Otorga la mayor expresividad con libertad sintáctica RDF aunque sin garantías computacionales. OWL Full depende principalmente de las necesidades que los usuarios tengan sobre los recursos según el esquema RDF, por lo cual OWL Full puede considerarse como una extensión de RDF debido a su grado de expresividad.[18]

18 Resumen extraído de **[21]** y **[22]** para ampliar la información del lenguaje visite las recomendaciones oficiales de W3C citadas en dichas referencias

La figura 9 muestra un ejemplo del código OWL utilizado para describir semánticamente un producto, donde se puede observa su identificador, recurso, propiedades y restricciones; cada una de estas propiedades forman parte de una clase y los valores de las propiedades deben ser miembros de dicha clase.

Los recursos (rdf:resource) pueden ser enlazados unos a otros formando de esta manera un grafo conceptual que defina explícitamente un determinado recurso, esta definición será única permitiendo establecer las definiciones sin ambigüedad requeridas por la web semántica; estos enlaces pueden ser recorridos por los agentes software para determinar la información necesaria como las propiedades o nombres de un recurso específico.

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:label xml:lang="en">wine</rdfs:label>
  <rdfs:label xml:lang="fr">vin</rdfs:label>
  ...
</owl:Class>

<owl:Class rdf:ID="Pasta">
  <rdfs:subClassOf rdf:resource="#EdibleThing" />
  ...
</owl:Class>
```

Figura 9. Código OWL[19]

1.4. ANÁLISIS DEL ESTADO DEL ARTE

Durante la investigación y recopilación de la información relacionada al Estado del Arte de los Servicios Web y la Web Semántica tratadas independientemente se ha tomado como principales fuentes de investigación los lineamientos recomendados por el W3C, pues de manera global el desarrollo de Internet y los Servicios Web en sí, se basan en los estándares recomendados por dicho organismo como son XML, HTTP, WSDL, SOAP, etc.

De la misma manera, se ha tomado distintas fuentes de referencia para los nuevos conceptos y tendencias de mejoramiento de los Servicios Web Semánticos como son los servicios REST, el lenguaje de descripción WADL y las recientes mejoras para el WSDL 2.0. Esta recopilación de información ha sido extraída y resumida de las mencionadas referencias con el propósito de presentar una base teórica clara que podría ser ampliada mediante la consulta de las fuentes mencionadas en esta sección.

En lo relacionado a la Web Semántica se ha tratado de realizar una historia y búsqueda del mejoramiento de la información para concluir con un análisis de los lenguajes y tecnologías estandarizadas y recomendadas por el W3C como son RDF y OWL; de la misma forma se ha soportado la investigación recopilando información de fuentes heterogéneas que presentan

19 **Código OWL:** Tomado de la Guía OWL de W3C, mayor información disponible en [http://www.w3.org/TR/owl-guide]

nuevos meta lenguajes y estructuras de información sencillas que podrán ser aplicadas con mayor facilidad en los nuevos generadores de información web, estos son, los microformatos.

Pues, dentro de los portales generadores de información como blogs, wordpress, joomla, y redes sociales entre otros, permiten incorporar información semántica sencilla al contenido web a través de sus plataformas con la utilización de metadatos y microformatos, los mismos que han llegado a tener un reconocimiento global tanto en gestores de búsqueda como Yahoo!, así como en plataformas de trabajo como iGoogle, Google Docs, Google Calendar.

El estudio general de los microformatos concluye con su aplicabilidad orientada a mejorar la Semántica que describe a los Servicios Web expuesta en el Capítulo 2; mediante la utilización de los microformatos como trabajo de integración entre los conceptos de la Web Semántica y los Servicios Web, en esta sección se tratarán los microformatos más representativos y utilizados hasta la fecha de investigación de la presente Tesis.

CAPITULO 2:
AVANCES EN LA INTEGRACIÓN DE LA WEB SEMÁNTICA Y LOS SERVICIOS WEB

2.1. AGREGACIÓN DE SEMÁNTICA EN CONTENIDO WEB

Hasta el momento se ha expuesto el Estado del Arte de los Servicios Web y la Web semántica independientemente, sin embargo, con el propósito de facilitar la creación de contenido web semántico que permita lograr una Web Semántica más rápidamente empresas como Yahoo, Google, Wordpress y gran parte de las redes sociales incorporan en su creación de contenidos varias etiquetas de significado semántico que se montan de manera transparente al usuario, por lo cual, se crea contenido semántico sin mayor necesidad de conocimiento de ontologías o lenguajes especializados.

Cabe resaltar, que la utilización de este tipo de etiquetas tiene una expresividad muy limitada por lo cual la semántica agregada a este tipo de información es bastante básica y por tanto, no tiene mayor influencia en el desarrollo de la Web Semántica como tal; sin embargo, es un paso firme hacia la creación de plataformas más fuertes que permitan generar organización jerárquica completa de la información y de esta forma una semántica bien estructurada para beneficio de todos los internautas, a continuación se presenta las formas más sencillas de incluir información semántica al contenido web.

2.1.1. USO DE MICROFORMATOS

Como se había indicado los microformatos son etiquetas sencillas que permiten incorporar diversos metadatos en el contenido HTML o XHTML; principalmente se pretende dar una solución más sencilla a la carencia de información semántica, mediante la utilización de metadatos como vCalendar, hCalendar que se utilizan de manera generalizada en las redes sociales y sin que el usuario deba tener conocimientos de lenguajes ontológicos.

De esta forma, los gestores de contenido como WordPress, foros y redes sociales se transforman en generadores de contenido web semántico a través de la utilización de microformatos, mejorando la calidad de información con la agregación semántica ligera a los contenidos creados por el usuario, ésta técnica se complementa con el soporte para microformatos de buscadores como Yahoo, que permiten una localización más precisa de información basándose en estas etiquetas.[20]

2.1.1.1. iCalendar

iCalendar es un estándar RFC 2445 para realizar la publicación de información de calendarios de trabajo, eventos y citas personales, el estándar también se conoce como iCal por la nomenclatura utilizada por Apple Computer que fue la primera en implementarlo.

20 Conceptos e información más detallada se encuentra en **[19]** y **[20]**

Las características de este tipo de microformato permite realizar invitaciones a eventos, reuniones o asignar tareas a otros usuarios a través del correo electrónico, la utilización de este tipo de microformatos facilita la interacción entre usuarios, respondiendo al mensaje recibido y planteando nuevas fechas de reunión o la aceptación de la fecha propuesta.

Este microformato ha sido apliamente aceptado por las principales empresas de software a través de diferentes interfaces como iCal de Apple, Mozilla Calendar, Google Calendar, Lotus Notes, Microsoft Outlook entre otros.

iCalendar se implementa generalmente a través de correo electrónico, sin embargo, ha sido creado independiente del protocolo de transmisión, lo cual facilita su utilización dentro de la Web Semántica permitiendo su amplia utilización en redes sociales, foros y publicación de eventos; principalmente se manejan los microformatos hCalendar y vCalendar.[21]

hCalendar: permite mostrar una representación semántica XHTML de información de calendario en formato iCalendar acerca de un evento dentro del contenido web, permitiendo que analizadores de contenido web extraigan los detalles del evento, su sintaxis se muestra en la figura 10.

```
<p class="vevent">
    La <span class="summary">Descripcion</span>
    Fecha de inicio
    <abbr class="dtstart" title="2001-01-15T14:00:00+06:00">
    Hora de inicio
    </abbr>
    Fecha de terminacion
    <abbr class="dtend" title="2001-01-15T16:00:00+06:00">
    Hora de terminación
    </abbr>
    <span class="location">
    Lugar donde se realizara el evento
    </span>
    (<a class="url"
    href="http://en.wikipedia.org/wiki/History_of_Wikipedia">
    más información...
    </a>)
</p>
```

Figura 10. Sintaxis iCalendar

vCalendar: El microformato vCalendar sirve para intercambiar información acerca de las citas y programaciones con otras personas a través de mensajes de correo electrónico, también se puede utilizar para publicar información en sitios web con una sintaxis que utiliza las propias etiquetas HTML. Su principal utilización es a través de clientes de correo electrónico como Outlook, Mozilla y aplicaciones OpenSource; la sintaxis utilizada para incluir información de calendario en un sitio web es la mostrada en la figura 11.

21 Conceptualización ampliada del microformato iCalendar se puede encontrar en [http://www.worldlingo.com/ma/enwiki/es/iCalendar]. Fuentes de resumen **[23]** y **[24]**

```
<div class="vcalendar">
<div class="vevent">
<dl>
<dt><strong>Titulo</strong></dt>
<dd><span class="summary">
      Descripcion del evento</span></dd>
<dd><span class="description">
      Descripcion del evento</span></dd>
<dd class="location">
      Lugar del evento</dd>
<dt><strong>Hora</strong>:</dt>
<dd><span class="dtstart" title="20060614T161500">
      Día y hora de inicio <sup></sup>
      </span>
<span class="dtend" title="20060614T174500">
      Día y hora de finalización</span></dd>
</dl>
</div></div>
```

Figura 11. Sintaxis vCalendar

2.1.1.2. RDFa y eRDF

Tanto eRDF como RDFa basan su trabajo en el lenguaje RDF, con eRDF se puede generar un pequeño subconjunto de grafos RDF de un solo nivel dentro de un documento plano HTML, eRDF es una sintaxis que utiliza los atributos HTML existentes como son ***class, rel*** y ***title*** para enlazar la información semántica RDF, también incluye etiquetas como ***dry*** y ***location,*** sin embargo, al trabajar con un único nivel RDF posee una expresividad limitada para la semántica.

RDFa es una extensión del lenguaje XHTML que ha sido propuesta por W3C para la incorporación de información semántica en los contenidos web aprovechando los atributos *meta* y *link* que posee XHTML, con la utilización de RDFa puede definirse una correspondencia simple con documentos RDF de un solo nivel.

Los atributos principales utilizados por RDFa son los siguientes:

- **typeof**: para indicar el tipo de instancia descrita.
- **about**: la URL que direcciona al recurso que describen los metadatos.
- **rel**, **rev**, **href** y **resource**: atributos para indicar la relación existente entre los recursos descritos mediante cada metadato.
- **property**: define la propiedad para el contenido de un elemento.

La integración de estos atributos dentro del documento XHTML se lo realiza como muestra la figura 12.[22]

22 Extracción de resumen obtenido de las fuentes **[25]** **[26]** y **[27]**

```
<p xmlns:dc="http://purl.org/dc/elements/1.1/"
about="http://www.example.com/books/wikinomics">
In his latest book
<em property="dc:title">Wikinomics</em>,
<span property="dc:creator">Don Tapscott</span>
explains deep changes in technology,
demographics and business.
The book is due to be published in
<span property="dc:date" content="2006-10-01">
October 2006</span>.
</p>
```

Figura 12. Microformatos RDFa y eRDF (inclusión en XHTML)

2.1.1.3. GRDDL

GRDDL es un mecanismo para obtener una descripción semántica de recursos a partir de documentos XML relacionándolos con un documento RDF, es decir, hace relación de las etiquetas (bien definidas) en un documento XML con las etiquetas necesarias para generar un documento RDF.

La figura 13 muestra el proceso de obtención de un documento RDF a partir de un documento web; en la primera parte se muestra un documento general XML bien definido tomando en cuenta las convenciones RFC2731, caso seguido se muestra un diagrama del proceso realizado por el mecanismo GRDDL y finalmente, se muestra el documento RDF resultante de esta interacción.

```
<html xmlns="http://www.w3.org/1999/xhtml">
   <head profile="http://www.w3.org/2003/g/data-view">
      <title>Some Document</title>
      <link rel="transformation"
       href="http://www.w3.org/2000/06/dc-extract/dc-extract.xls"/>
      <meta name="DC.Subject"
         Content="ADAM; Simple Search; Index+; prototype" />
      ……
   </head>
   ……
</html>
```


```
<rdf:RDF xmlns:dc="http://purl.org/dc/elements/1.1"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="">
     <dc:subject>ADAM; Simple Search; Index+; prototype
     </dc:subject>
</rdf:RDF>
```

Figura 13. Proceso GRDDL[23]

23 La transformación de microformatos a RDF se explica más detalladamente en las fuentes **[28] [29]** y **[30]**, el

2.1.1.4. Comparación de metodologías

Una muestra de las características y limitaciones generales de los microformatos y los lenguajes semi estructurados como son RDFa y eRDF se muestran a través de la siguiente tabla:

Tabla 2.1. Tabla comparativa de tecnologías[24]

CARACTERÍSTICA	MF	eRDF	RDFa
Propiedad DRY (Don't Repeat Yourself)	SI	SI	MAYORIA
Validez en HTML4 / XHTML	SI	SI	NO
Permite mezclar etiquetas	NO	SI	SI
Descripción arbitraria de recursos	NO	SI	SI
Significado sintáctico explicito para descripción	NO	NO	SI
Soportado por W3C	PARCIAL	PARCIAL	SI
Normativa DCMI	NO	SI	NO
Especificación estable de sintaxis	PARCIAL	SI	SI
Fácil Mapeo RDF	MAYORÍA	SI	SI
Compatibilidad web	SI	MAYORÍA	MAYORÍA
Copia, agregación y reutilización confiable (Self-containment)	MAYORÍA	PARCIAL	PARCIAL
Soporta datos no textuales (decimales, fechas)	SI	NO	SI
Pueden generarse Tripletas RDF a partir de ellos	SI	SI	NO
Integración en namespace mediante XML	NO	NO	SI
Adoptado por mayoría de desarrolladores web	SI	NO	NO
Soporta nodos vacíos	NO	NO	SI
Sintaxis basado en HTML como address o rel/rev/class	SI	MAYORÍA	PARCIAL
Soporta rel=”nofollow” en las etiquetas	SI	NO	PARCIAL

Propiedad DRY (No repetirse a si mismo)

- RDFa: El sentido textual del formato RDFa es utilizado de manera redundante en el contenido de los documentos.

Soportado por W3C

- Microformatos, eRDF: Indirectamente soportados por W3C mediante GRDDL.

ejemplo ha sido tomado de **[30]** para una mejor comprensión del proceso

24 **Tabla comparativa:** Toma en consideración varios aspectos del manejo y estandarización W3C, analizado en base a [http://microformats.org/wiki/principles] así como **[26]** y **[27]**

Especificación estable de sintaxis

- Microformatos: Aunque los microformatos utilizan las estructuras HTML, el formato de la sintaxis posee claras diferencias, así cada microformato requiere reglas separadas para su análisis.

Fácil mapeo RDF

- Microformatos: Los microformatos pueden ser mapeados a diferentes estructuras RDF.

Compatibilidad web

- eRDF, RDFa: Requieren mejorar la compatibilidad para ser tratados de manera estándar.

Fácil mapeo RDF

- Microformatos: Los microformatos pueden ser mapeados a diferentes estructuras RDF.

Copia, agregación y reutilización confiable. (Self-containment)

- Microformatos: Algunos microformatos pierden fácilmente su significado semántico al equivocar el contexto.
- eRDF/RDFa: Únicamente ciertas partes de éstos pueden ser reutilizados de manera confiable.

2.1.2. REPOSITORIO Y SERVICIOS PARA AGREGACIÓN SEMÁNTICA

Una vez establecidos los principios de la Web Semántica y una idea general de la complejidad de trabajar con lenguajes ontológicos como RDF y OWL, se presenta también la posibilidad de utilizar repositorios semánticos con la finalidad de incorporar sentido semántico al contenido web tradicional, para ello se dispone de dos tipos similares de repositorios que poseen un funcionamiento de Servicio web, estos repositorios son Jena y Sesame.

2.1.2.1. SESAME

SESAME es un framework Open Source que trabaja como repositorio de información en RDF con soporte para RDF Schema, N-Triples y N3, razonamiento e interrogación; ha sido desarrollado principalmente por la empresa alemana Aduna para luego formar un prototipo de investigación dentro del proyecto On-To-Knowledge dedicado al mejoramiento ontológico de la información; en la actualidad Aduna se encarga de su mantenimiento y desarrollo en cooperación con la Fundación NLet, Ontotext y desarrolladores independientes que voluntariamente contribuyen con ideas y mejoras al código fuente.

Principalmente Sesame como repositorio de información ofrece comunicación local o remota a través de HTTP o RMI y un ambiente de consulta para el lenguaje SerQL como se indica en la figura 14, con la finalidad de realizar consultas e incorporar información semántica almacenada al contenido web que se vaya generando, este trabajo lo realiza como un Servicio Web a través de una interface HTTP basada en RESTful, la descarga de Sesame nos devuelve los paquetes necesarios para trabajar con el lenguaje Java.

La característica principal de Sesame es su gran flexibilidad para determinar el almacenamiento de información RDF, pues, permite el almacenamiento en memoria, indexación por palabras clave, archivos de sistema y bases de datos relacionales; soporta el lenguaje de consulta SparQL, pero, para mejorar su funcionamiento ha desarrollado su propio lenguaje de consulta SeRQL.[25]

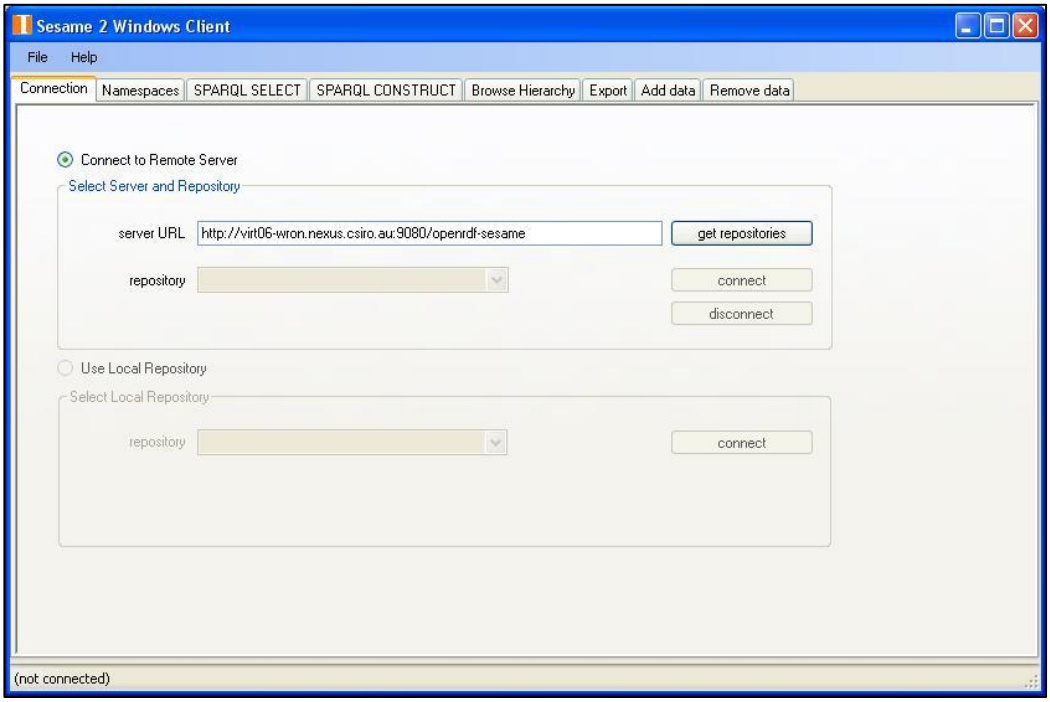


Figura 14. Ambiente SESAME[26]

25 Síntesis de la descripción SESAME como repositorio ontológico, para ampliar información referirse a **[32]** y **[33]**
26 Ambiente instalado del repositorio SESAME tomado del tutorial de instalación disponible en:

2.1.2.2. JENA

Conformado de manera similar a Sesame, JENA es un repositorio de información semántica desarrollado en Java que permite el almacenamiento de ontologías OWL, importación de archivos RDF y N3, sin embargo, JENA trata a los conceptos OWL no solamente como una ontología, sino también como un modelo RDF lo cual permite obtener un mayor grado de inferencia y filtrado de la información al momento de utilizar estas ontologías almacenadas.

Otra característica de JENA es su razonamiento taxonómico, el cual permite un análisis jerárquico de la información almacenada, facilitando de cierta manera la búsqueda por clases y subclases. El razonamiento real sobre las ontologías se trabaja con el lenguaje SparQL basándose en el modelo de razonamiento JENA mostrado en la figura 15; todas sus funciones y métodos se exponen al descargar los paquetes Java que lo conforman y su API para interacción.[27]

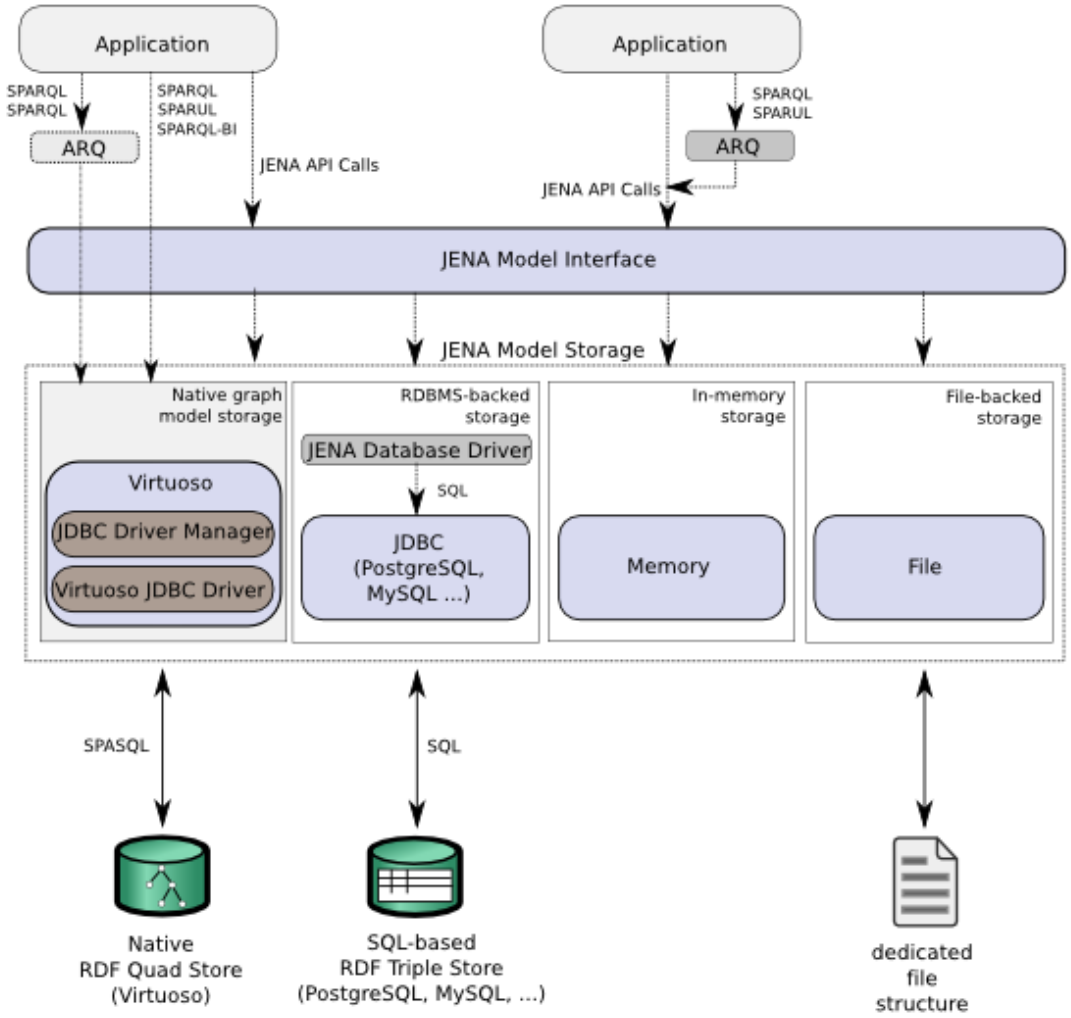


Figura 15. Diagrama general repositorio JENA[28]

[https://wiki.csiro.au/confluence/pages/viewpage.action?pageId=109314628]

27 Compendio desarrollado en base a **[34]** y **[35]** como medios de información ampliada

28 **Repositorio JENA:** Detalle de la estructura y conceptos disponible en [http://docs.openlinksw.com/images/ui]

2.2. DESCRIPCIÓN SEMÁNTICA DE LOS SERVICIOS WEB

Tomando en cuenta la gran cantidad de información que existe actualmente en Internet y las mejoras que hasta el momento se han logrado con la utilización de la Semántica, es indispensable iniciar con la descripción semántica de los Servicios Web para de esta manera intentar una mejor interacción, permitiéndose el descubrimiento, invocación y composición automática de los servicios con la utilización de agentes software y por tanto, reduciendo el tiempo de desarrollo de sitios basado en Servicios Web.

Para la descripción semántica de los Servicios Web se proponen varias alternativas que se detallan en los siguientes puntos.

2.2.1. OWL-S

OWL-S no es en sí un lenguaje, sino un vocabulario estándar para la descripción semántica de un Servicio Web que se basa en el lenguaje ontológico OWL para permitir la descripción semántica del perfil del servicio con la finalidad de su publicación y descubrimiento, este lenguaje permite la creación de una descripción detallada de cada operación que el Servicio Web realiza incluyendo el modo de interacción con dicha operación.

Para obtener una perspectiva de cómo interactuar con un servicio, dicho servicio tiene que ser visto como un proceso donde se envían y reciben mensajes simples o complejos, además cada método del servicio puede ofrecer características necesarias para otro método por lo cual se involucran tiempos de espera y secuencias de trabajo.

De esta forma, la descripción semántica de un servicio web se sigue realizando de manera compleja con la utilización de OWL requiriendo la utilización de ciertas propiedades como:

- **hasParticipant** para la declaración de un participante en el proceso
- **hasInput** que define los parámetros de entrada
- **hasOutput** que define los parámetros de salida
- **hasPrecondition** para las precondiciones necesarias
- **hasResult** describe los resultados devueltos por el método

En la figura 16 se muestra un documento OWL-S con un código similar a OWL que servirá como descripción semántica del Servicio Web, esta pequeña fracción de código indica algunas de las propiedades descritas anteriormente.[29]

```
<owl:ObjectProperty rdf:ID="hasParticipant">
  <rdfs:domain rdf:resource="#Process"/>
</owl:ObjectProperty>
……
……
<owl:ObjectProperty rdf:ID="hasClient">
  <rdfs:subPropertyOf rdf:resource="#hasParticipant"/>
</owl:ObjectProperty>
……
……
<process:Parameter rdf:ID="TheClient">
<process:Parameter rdf:ID="TheServer">
```

Figura 16. Ejemplo OWL-S

2.2.2. WSDL-S

El WSDL-S es un concepto aplicado al lenguaje de descripción de servicios WSDL, pues, se pretende incluir información semántica en el mismo documento descriptivo WSDL a través de enlaces dirigidos a una ontología OWL. Este enfoque ofrece varias ventajas sobre OWL-S, pues permite una mejor reutilización de los recursos y una mayor organización de servicios y ontologías al mantenerlos en documentos separados.

La creación de documentos con descripción semántica se realiza con la utilización de elementos de extensibilidad para WSDL, dichos elementos se presentan brevemente.

- ***modelReference:*** especifica la asociación entre un documento WSDL y un concepto ontológico.
- ***schemaMapping:*** adiciona un XML Schema con tipos complejos para el manejo de diferencias estructurales entre los elementos del Servicio Web y la ontología
- ***category:*** Involucra información de la categoría del servicio para ser publicada en un repositorio UDDI.
- ***precondition:*** es un subelemento de la operación que es utilizado para la conceptualización de requerimientos

29 Información ampliada disponible en **[36]**, ejemplo incluido en el mismo documento

- ***effect:*** también forma parte de una operación mostrando la conceptualización de los parámetros de salida.

Los dos últimos elementos pueden ser utilizados opcionalmente, pues su finalidad es el descubrimiento automático de los Servicio. [30]

2.2.3. SAWSDL

SAWSDL de forma similar a WSDL-S define la manera de agregar información semántica en varios elementos de un documento WSDL como las estructuras de los mensajes de entrada/salida, interfaces y operaciones; este tipo de especificaciones pueden ser publicadas en los repositorios UDDI.

Las anotaciones semánticas en un documento WSDL conforman información adicional que identifica o define un concepto dentro del contexto semántico; en SAWSDL las anotaciones semánticas son atributos XML incluidos en el documento sintáctico WSDL ampliando su información con el uso de una ontología.

De manera similar a WSDL-S se trabaja mediante el uso de etiquetas XML para incorporar la información semántica descrita en OWL o RDF; a continuación la figura 17 muestra un ejemplo del código SAWSDL[31]

```
<xs:element name="OrderRequest"
sawsdl:modelReference="http://www.w3.org/…purchasorder#OrderRequest"
sawsdl:loweringSchemaMapping="http:/www.w3.org/……RDFOntRequest.xml">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="***" type="xs:integer" />
                        <xs:element name="orderItem" type="item"
        minOccurs="1" maxOccurs="unbounded" />
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

Figura 17. Código SAWSDL

30 Tomado del documento W3C disponible en [http://www.w3.org/Submission/WSDL-S/]
31 Extraído del documento W3C disponible en **[38]** y analizado en la herramienta WSMO Studio

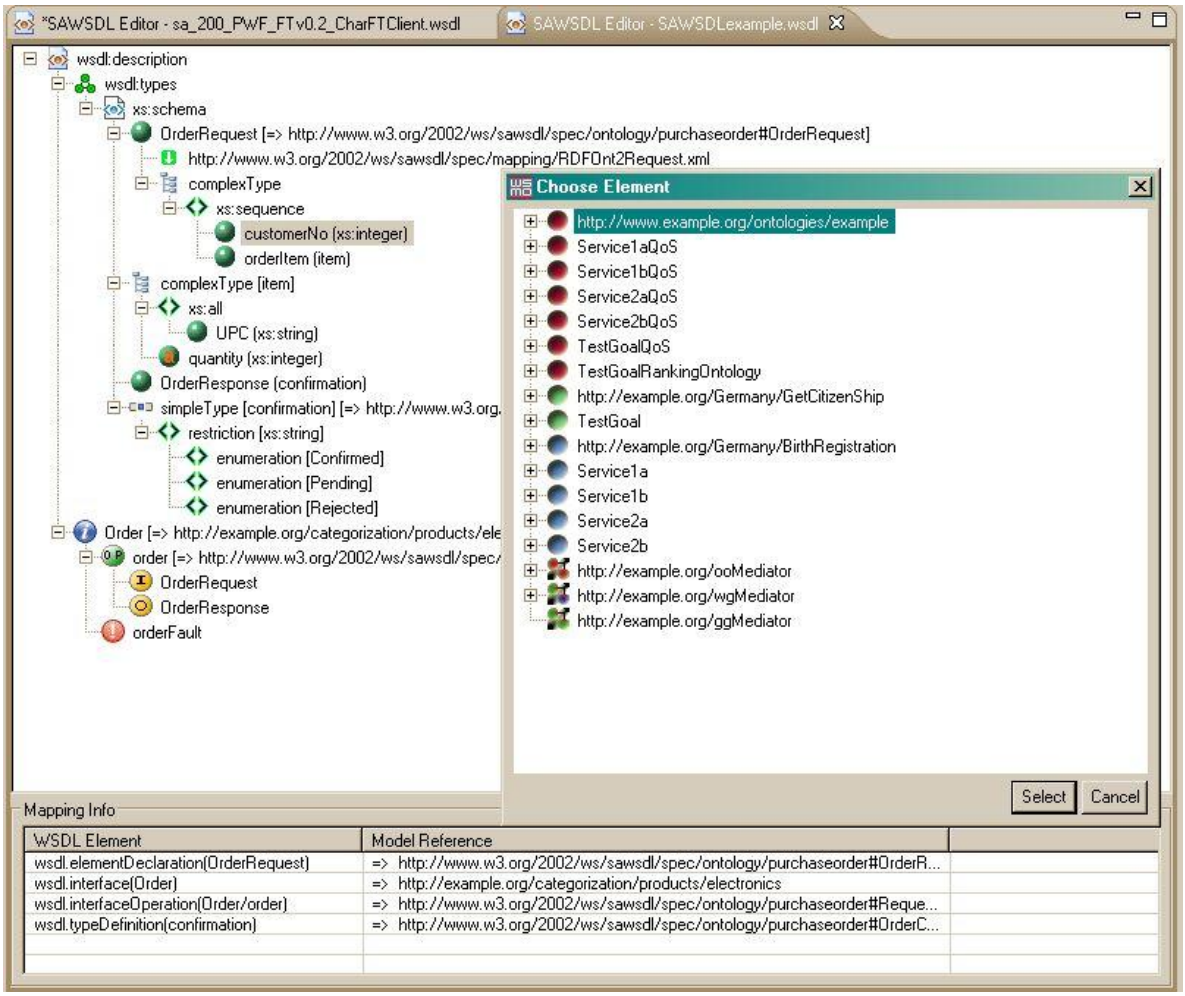


Figura 18. Herramienta WSMO Studio

El trabajo con la herramienta WSMO Studio (figura 18) nos permite el manejo de ontologías y el mejoramiento semántico de la descripción WSDL de un Servicio Web mediante la utilización de su herramienta SAWSDL.

2.2.4. SA-REST

SA-REST trabaja similarmente a WSDL-S al separar los modelos semánticos u ontológicos del código general de descripción del Servicio Web, para lo cual propone la utilización de cualquier tipo de modelo semántico OWL, RDF, RDFa y utiliza el mecanismo GRDDL para realizar la tarea de relacionar conceptos semánticos u ontologías con los Servicios Web de metodología REST.

El mecanismo de trabajo de SA-REST se basa en la utilización de tripletas (sujeto-predicado-objeto) para lograr una descripción similar a la obtenida por WSDL-S y SAWSDL, por ejemplo, el mensaje de salida de un método en un documento WSDL puede ser comentado con una URI hacia una ontología que representa dicha salida, de esta manera se puede realizar la inserción

de información semántica en métodos, parámetros de entrada, salida y errores que maneja un determinado Servicio REST.

La figura 19 a continuación muestra un ejemplo del código HTML que incluye información semántica mediante la utilización de SA-REST.[32]

```
<html xmlns:sarest=”http://knoesis.wright..edu/SAREST#”>
   <p about=”http://craigslist.org/search/”>
      The logical input of this service is an
      <span property=”sarest:input”>
           http://knoesis.wright..edu/ont.owl#Location_Query
      </span>
           Object. The logical output of this service is a list of
      <span property=”sarest:output”>
           http://knoesis.wright..edu/ont.owl#Location
      </span>
      Objects. This service should be invoked using an
      <span property=”sarest:action”>
           HTTP GET
      </span>
      <meta property=”sarest:lifting” content=”
http://craigslist.org/api/lifting.xls”>
      <meta property=”sarest:lowering” content=”
http://craigslist.org/api/lowering.xls”>
      <meta property=”sarest:operation” content=”
http://knoesis.wright..edu/ont.owl#Location_Search”>
```

Figura 19. Ejemplo de código SA-REST

2.3. CONCLUSIONES DE LA FASE

En los diferentes apartados que conforman la primera fase de esta Tesis se ha recopilado información que servirá como marco teórico para el desarrollo de los siguientes apartados, es por este motivo que se ha realizado un compendio, pretendiendo incorporar la información esencial de cada concepto generando definiciones claras y concisas que podrán ser ampliadas en las fuentes citadas.

El trabajo de organizaciones como W3C - que incluyen miembros representativos de las principales empresas de Software como IBM, HP, Microsoft por nombrar algunas - es realizar la innovación de información y mejoramiento de estándares que sirvan de guía para el desarrollo de Internet hacia una web enriquecida; por este motivo las principales fuentes de investigación mencionadas son los estándares actuales, recomendaciones y tecnologías tomadas en cuenta por dicho organismo.

Principalmente, se ha notado el acelerado incremento de tecnologías y plataformas que soportan los microformatos (Twitter, Plurk, Tanzania, Google Maps, etc.) y nuevas técnicas que permiten la extracción e incorporación de información semántica a través de los diferentes microformatos (SA-REST, SAWSDL, GRDDL); siendo un punto muy importante a tener en cuenta pues,

32 La información de procesos y funciones involucrados pueden ser ampliados en **[40]** y **[41]**

facilitará el camino hacia una web semántica global sin complicar en mayor medida el desarrollo de los sitios y plataformas web.

Con el continuo desarrollo de microformatos y tecnologías que integren la semántica a los Servicios Web se tendrá un mayor campo de aplicación para la innovación de los Servicios Web Semánticos objeto de esta investigación; pudiendo enfocarse en los diferentes métodos de generación de información semántica como son OWL-S, WSDL-S, SAWSDL y SA-REST; así como el manejo de repositorios de información semántica SESAME y JENA que a su vez conforman un Servicio Web, o finalmente la incorporación de microformatos a manera de Servicios Web para la generación de información pública bien definida a través de redes sociales y plataformas de manejo de contenido.

En las siguientes fases de la presente investigación se optará por trabajar con los Servicios Web Semánticos como núcleo, enfocándose hacia el manejo automático de información mediante la interacción de agentes software, bajo el trabajo de procesos y conceptos con lenguajes ontológicos propios y herramientas que provean de un ambiente estable para el trabajo de modelado de los Servicios Web Semánticos.

FASE 2

CAPITULO 3:

LOS SERVICIOS WEB SEMÁNTICOS

3.1. INTRODUCCIÓN

Los Servicios Web Semánticos serán una tecnología integrada para la próxima generación de las aplicaciones en la Web, en donde se combinen la tecnología de la Web Semántica y la tecnología de los Servicios Web, de manera que transformen la Internet que conocemos, de un repositorio de información para consumo humano a un sistema global de computación web distribuida, donde tanto usuarios humanos como agentes software puedan interactuar con la información expuesta en la Web.

La interacción entre Servicios Web se realiza utilizando como base la descripción WSDL de los Servicios, la misma que presenta una información operacional sintáctica de las funcionalidades que dicho Servicio presta, así como los mensajes de entrada/salida y posibles errores que se involucran en la utilización del Servicio; sin embargo, no incorpora información semántica.

Los Servicios Web Semánticos incorporan información semántica a la descripción WSDL de los Servicios Web con el propósito de evitar la ambigüedad en los requerimientos de entrada, valores de salida y operaciones realizadas por el Servicio, con esto se pretende alcanzar el descubrimiento e invocación automática de los Servicios por medio de agentes inteligentes.[33]

3.2. CICLO DE VIDA DE LOS SERVICIOS WEB

La utilización de los Servicios Web Semánticos principalmente se encuentra en los ambientes de producción de negocios complejos que requieren de la utilización de varios Servicios Web, es por ello que, se tomará como idea general un proceso de negocio para tener en consideración el ciclo de vida de los Servicios Web, el mismo que será tomado como base para el estudio de los pasos que conforman los Servicios Web Semánticos y una breve exposición de las mejorar logradas con la incorporación de Semántica en cada una de estas fases.

El ciclo de vida de los Servicios Web[34] mostrado en la figura 20, involucra las diferentes etapas que conforman un proceso de negocios relacionado a la producción de Servicios Web como son la Publicación, Descubrimiento, Selección, Composición e Invocación de dichos servicios; en cada una de estas fases se incorporará la Semántica para estudiar el comportamiento teórico de los Servicios Web Semánticos en un ambiente de producción. Para un conocimiento más preciso

33 Extracción de ideas principales de las fuentes **[41] [42] [43]** y **[44]** donde se podrá ampliar la información expuesta en este capítulo

34 **CARDOSO**, Jorge. "Semantic Web Services: Theory, Tools and Applications". Pág. 240

de la metodología de desarrollo de los Servicios Web Semánticos es necesario tener una idea general de los conceptos antes mencionados.

Figura 20. Ciclo de vida de los Servicios Web[35]

3.2.1. PUBLICACIÓN

La publicación de un Servicio Web Semántico se realiza de manera similar a los Servicios Web tradicionales mediante la utilización de los repositorios UDDI públicos o privados, donde se ingresa información general relacionada al Servicio Web expuesto, teniendo en cuenta aspectos como:

- Proveedor
- Descripción textual del Servicio
- Documento descriptivo WSDL
- Dirección URL de entrada para invocación del Servicio

El mejoramiento semántico en la fase de publicación incorpora la descripción semántica del servicio mediante la utilización de los lenguajes WSDL-S, SAWSDL, SA-REST como hemos tratado en puntos anteriores o con la utilización del lenguaje WSML que se tratará en capítulos posteriores. Mediante la descripción semántica de los Servicios Web en su etapa de publicación estamos transformándolos en Servicios Web Semánticos, por tanto se tratarán de manera diferente en las siguientes fases que conforman el ciclo de vida de los Servicios Web.

Esta publicación permite catalogar los servicios para que los clientes de una empresa puedan conocer de su existencia, examinen los métodos, procesos e interfaces para su utilización y de esta manera generar servicios más complejos basándose en Servicios Web ya existentes. La

35 Adaptado de: **CARDOSO**, Jorge. "Semantic Web Services: Theory, Tools and Applications" Página 241

publicación de Servicios Web se realiza en repositorios UDDI tanto públicos como privados, pudiendo publicarse también Servicios Web Semánticos en estos repositorios, pues principalmente se trata de la publicación del documento descriptivo WSDL del servicio, el mismo que podría incorporar información semántica como WSDL-S o SAWSDL.

Posteriormente en los capítulos 6 y 7 se analizará el repositorio WSMX dentro de las herramientas propias para el desarrollo de Servicios Web Semánticos, el mismo que se encuentra incluido en un ambiente de producción o puede ser tomado como repositorio público o privado independientemente. **[43] [44]**

3.2.2. DESCUBRIMIENTO

El descubrimiento de un Servicio Web es el proceso de búsqueda dentro de la información del servicio (descripción sintáctica), basándose principalmente en los requerimientos que el usuario espera encontrar en un Servicio Web para formar un nuevo servicio u obtener una respuesta esperada.

Este proceso realiza una búsqueda en el repositorio según los parámetros enviados por el solicitante, y nos devuelve una cantidad de Servicios Web relacionados al tema buscado; cabe destacar que el proceso de descubrimiento tradicional se lo realiza en base a la coincidencia de palabras o entradas y salidas que posee el documento WSDL. Sin embargo, al tratarse de Servicios Web Semánticos es prioritaria la incorporación de significado en los documentos descriptivos, pues, de esta manera se puede realizar un descubrimiento automático basado en consultas SparQL u otras herramientas de consulta como WSML Reasoner que facilitan la tarea de búsqueda semántica en Servicios Web que incorporan información WSML.

El descubrimiento automático de Servicios implica encontrar un Servicio dentro de un repositorio centralizado o distribuido que coincida con los requerimientos entregados. El momento de realizar la búsqueda puede surgir una coincidencia sintáctica (basada en el tipo y la estructura) o puede ser semántica (equivalencia de significados), pero para cumplir un propósito específico en los Servicios Web Semánticos es necesario que el Servicio coincida sintáctica y semánticamente.

Es decir, existen varios escenarios en el proceso de búsqueda y descubrimiento de Servicios Web dependiendo de la similitud semántica y sintáctica del Servicio con los requerimientos solicitados. Para citar varios casos a manera de ejemplo:

- Servicios que ofrecen un resultado idéntico pero requieren diferentes parámetros de ingreso.
- Servicios con descripciones y parámetros de ingreso similares pero que ofrecen una salida diferente.

- Servicios que ofrecen una salida idéntica y requieren parámetros similares, como la solicitud de un código, el cual podría trabajarse con códigos SKU de 12 dígitos o códigos UPC de 14 dígitos en empresas o clientes diferentes.

- Servicios que coinciden exactamente en sus parámetros de ingreso y sus resultados, éstos últimos son Servicios que coinciden sintáctica y semánticamente.[36]

3.2.3. SELECCIÓN

Una vez descubierto un Servicio Web basándose en los requerimientos se realiza la selección del Servicio más adecuado que cumpla con las necesidades del proceso, estos servicios no son todos idóneos por las razones expuestas en el Descubrimiento, el proceso de Selección se basa en la coincidencia semántica y sintáctica de los Servicios Web que han sido descubiertos.

La selección del mejor Servicio tiene que relacionarse con la similitud sintáctica del documento descriptivo WSDL para el caso de los Servicios Web tradicionales, y en la similitud semántica en los documentos WSDL-S, SAWSDL, WSML para el caso de los Servicios Web Semánticos; pues la agregación de información semántica en el documento descriptivo nos entrega significados claros de cada uno de los parámetros involucrados en el manejo del servicio.

El proceso de Descubrimiento puede devolvernos varios Servicios con diferentes grados de coincidencia, el proceso de Selección debe encargarse de elegir aquel Servicio que coincida sintáctica y semánticamente o en su defecto una coincidencia semántica con diferencias sintácticas.

Para realizar una correcta selección de un Servicios Web debemos basarnos principalmente en los conceptos ontológicos enlazados a dicho servicio, tomando como prioridad aquellos servicios que trabajen con una misma ontología y no sea necearía la traducción de una ontología a otra; los diferentes grados de coincidencia se muestran en la figura 21.[37]

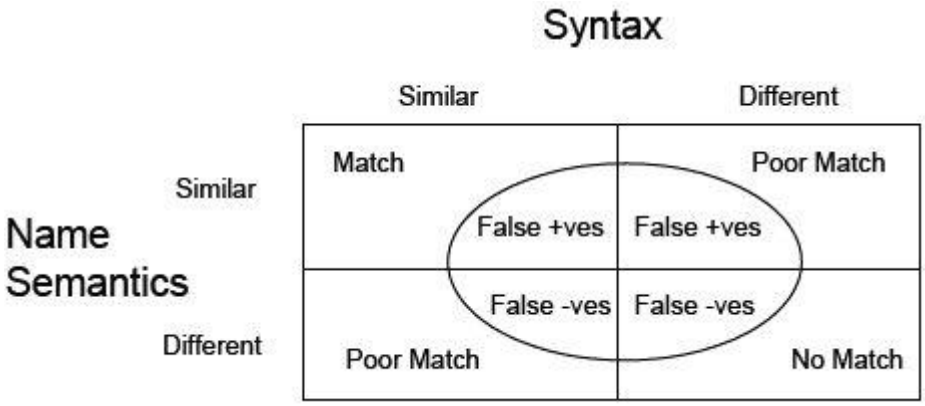


Figura 21. Tipos de correspondencias[38]

36 Proceso de descubrimiento extraído de **[9][44][60]** y **[67]**

37 Conceptos tomados de **[41]** donde se ofrece una descripción más amplia del proceso de selección

38 **Tipos de correspondencias:** Tomado de **[23]** Pag. 196. Existe mayor información de los tipos de correspondencia

3.2.4. COMPOSICIÓN

La composición de Servicios Web existentes con la finalidad de entregar nuevas funcionalidades es un requisito en muchos ambientes de negocios, esta composición de Servicios favorece al usuario, pues se realizarán de manera transparente complementando al descubrimiento e invocación.

En varios escenarios se requerirá la transformación de parámetros para coincidir con los solicitados por el Servicio, como por ejemplo cambiar un código UPC a un código SKU para cumplir con el requerimiento de un determinado Servicio, de manera similar permitir que el Servicio Web complete un formulario de solicitud necesario para invocar otro Servicio.

Estos procesos de transformación requieren de la intervención de un Servicio Web complementario, este es el principio de la composición de Servicios mostrado en la figura 22.

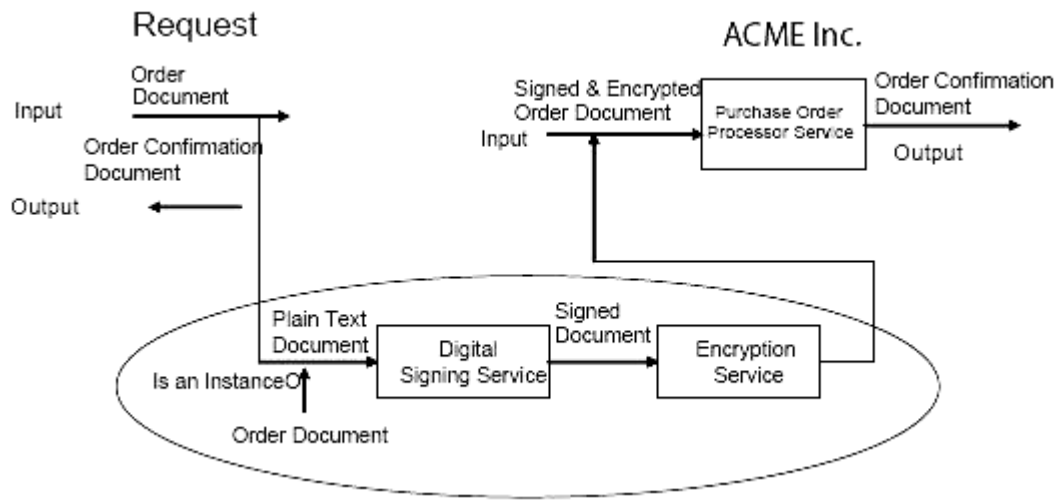


Figura 22. Trabajo de composición[39]

Dicha composición se realiza de manera manual en los Servicios Web tradicionales, mientras que, en el desarrollo de los Servicios Web Semánticos se pretende lograr una composición automática de dichos servicios, para lo cual es necesario retomar los pasos de Publicación, Descubrimiento y Selección en busca del servicio complementario; esta fase se encuentra en continuo avance con la incorporación de *mediadores*, Coreografía y Orquestación que se expondrán posteriormente en este mismo capítulo y serán ampliados en el capítulo 6 con el uso de una herramienta específica.[40]

39 **Trabajo de composición**: Tomado de **[23]** Pag. 197

40 Conceptos y teorías recopilados de **[41]** y **[18]** información más detalla en dichas fuentes

3.2.5. INVOCACIÓN

La invocación de los Servicios Web tradicionales se realiza de forma manual, pues los pasos que conforman el Ciclo de Vida de los Servicios Web deben ser llevados a cabo por el equipo de desarrolladores, concluyendo con la selección –manual- del mejor servicio según el análisis de requerimientos, entradas, salidas y factibilidad de los servicios publicados.

En los Servicios Web Semánticos durante el proceso automático de Invocación de Servicios Web es posible encontrarse con parámetros de ingreso diferentes a los que se posee, por ejemplo, se dispone de dos parámetros como son *nombre* y *apellido*, y el descubrimiento de Servicios nos arroja un Servicio Web que solicita como parámetro de ingreso un *nombre completo*, es decir nombre y apellido concatenados; la invocación de Servicios Web involucra resolver este tipo de problemas realizando un análisis más complejo de la semántica relacionada con los Servicios.

Este tipo de asociaciones pueden ser derivadas de un modelado semántico y deben ser resueltas utilizando Servicios complementarios o generando código que permita al Servicio la incorporación de parámetros alternativos para la invocación.

El proceso de invocación de un Servicio Web complementario es en sí, otro proceso del ciclo de vida de los Servicios Web, pues, en caso de requerir ser invocado automáticamente debe haber cumplido los pasos de publicación, descubrimiento, y selección. **[43]**

3.3. FASES COMPLEMENTARIAS

En el manejo automático de Servicios Web Semánticos se encuentran inmersos nuevos conceptos como son la Orquestación y Coreografía, que colaboran en el manejo y organización de mensajes entre los Servicios Web Semánticos involucrados; pues al momento de tratar con descubrimiento, selección e invocación automática es necesario controlar dichas tareas.

Como se ha analizado anteriormente, existen casos especiales durante los procesos de negocio donde se requiere la transformación de un parámetro en otro para completar los requerimientos que un Servicio Web solicita, este tipo de procesamiento debe realizarse de manera transparente al usuario, y por tanto, es un escalón adicional en la complejidad del manejo de los Servicios Web Semánticos, requiriendo para ello procesos adicionales.

3.3.1. COREOGRAFÍA

La Coreografía permite especificar las reglas de unión y trabajo colaborativo de los Servicios Web desde el punto de vista del cliente, tomando en cuenta que el cliente puede ser otro

Servicio Web o una interacción con un usuario humano, por tanto, la coreografía establece el comportamiento de un servicio frente a una invocación.

El proceso de Coreografía se basa en la utilización de varios servicios para lograr una única meta común, es un conjunto determinado de interacciones entre servicios, y por tanto requiere de una supervisión, esta secuencia de interacciones se controla automáticamente a través de la definición de una coreografía, donde se definirá ***que*** servicio realiza una determinada acción (petición o espera) y de esta manera evitar la pérdida de secuencia durante la composición de servicios.

La coreografía puede entenderse como un proceso público y no ejecutable: es ***público*** porque define el comportamiento común y globalmente visible entre los diferentes participantes en una interacción; por otro lado es ***no ejecutable*** porque no está pensado para ser llevado a cabo, sino para actuar como un protocolo de negocio que dicta reglas de interacción que deben ser cumplidas por las entidades participantes.

El manejo de procesos de negocios complejos da cabida a la necesidad de definiciones de coreografía, la metodología de trabajo WSMO incluye definiciones para coreografía dentro de la descripción semántica del Servicio Web. Una visión más clara del significado del proceso de Coreografía se muestra en la figura 23.

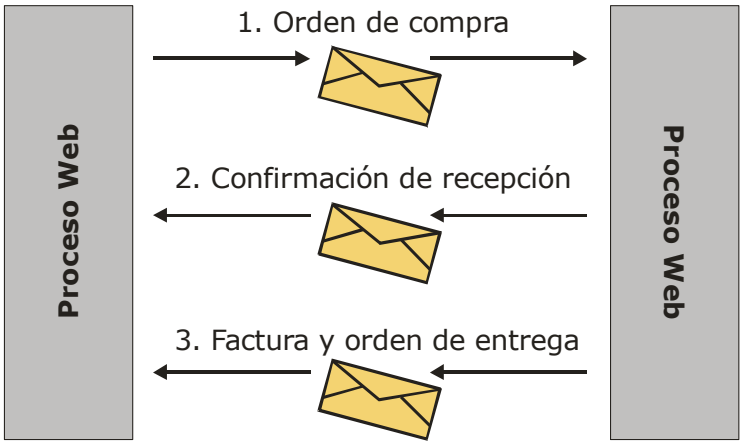


Figura 23. Coreografía de Servicios Web

3.3.2. ORQUESTACIÓN

La Orquestación en complemento a la definición de la coreografía establece y controla un mecanismo adecuado para el intercambio de mensajes entre los Servicios y la manera en que un servicio hace uso de otros servicios para lograr una meta determinada, como por ejemplo, en el comercio electrónico un servicio puede hacer uso de otro servicio que le permita realizar la

validación de una tarjeta de crédito, debitar el valor de cobro y asignarlo al cliente sin que éste último conozca las empresas involucradas en dicha transacción.

De esta manera la Orquestación controla y coordina el intercambio de mensajes entre Servicios verificando que cada Servicio envíe y reciba la información adecuada y una vez obtenido el resultado esperado, el servicio pasará un `token` al siguiente servicio involucrado en dicho proceso, por tanto, la Orquestación nos proporciona una manera de describir el comportamiento interno de ***que*** necesitan los servicios para trabajar conjuntamente y así poder crear otro servicio como composición de los primeros.

La figura 24 muestra de una manera precisa el trabajo realizado por el proceso de Orquestación en un conjunto de Servicios cooperativos.[41]

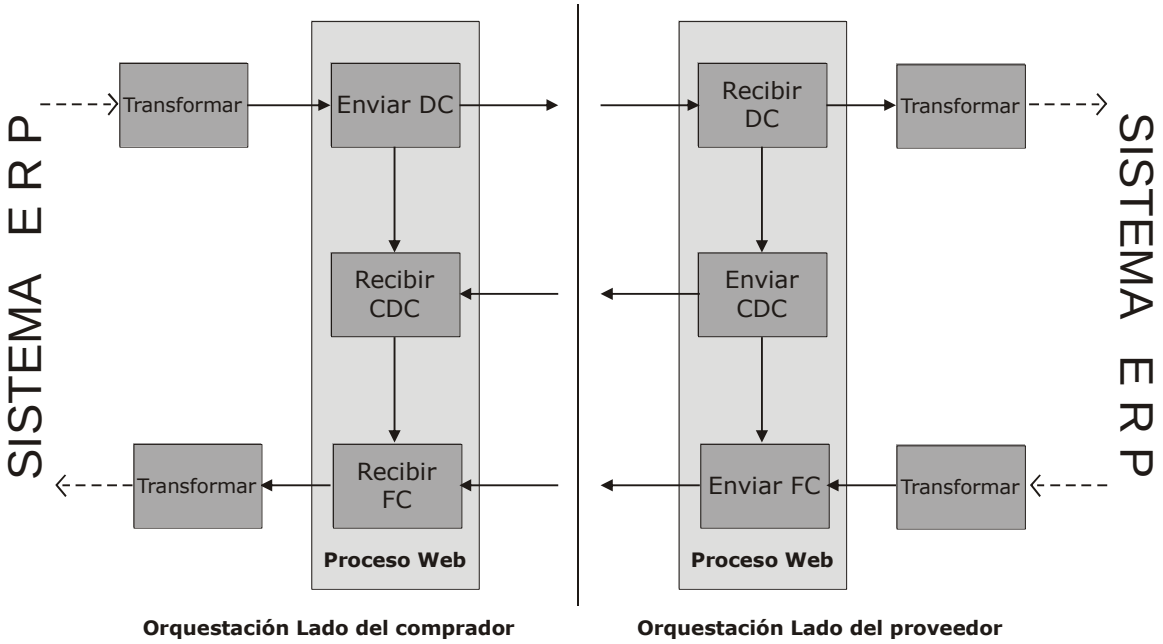


Figura 24. Orquestación de Servicios Web[42]

3.4. ANÁLISIS DE LOS SERVICIOS WEB SEMÁNTICOS

Como se ha expuesto a lo largo del desarrollo del presente Capítulo el Ciclo de Vida de los Servicios Web define claramente el camino a ser recorrido por los Servicios Web Semánticos, con la diferencia clara que, en los Servicios Web tradicionales cada una de las fases que

41 Las fases complementarias de Coreografía y Orquestación han sido tomadas de **[14] [18] [41]** y [http://www.lawebsemantica.com/orquestacionCoreografia.html] consultado 22 de Noviembre 2008

42 **Diagramas de Coreografía y Orquestación**: Obtenidas de: [http://www.lawebsemantica.com/orquestacionCoreografia.html]

conforman el Ciclo de Vida debe ser realizada manualmente por los desarrolladores; mientras que, en los Servicios Web Semánticos se pretende realizar estos pasos de manera automática.

Teniendo claros los principios que conforman el Ciclo de Vida de los Servicios Web tenemos una base sustentable para reconocer los diferentes avances realizados en el campo de los Servicios Web Semánticos como tal, posteriormente se procederá a realizar la comparación de varias herramientas existentes que permitan un trabajo semiautomático del modelado de los Servicios Web Semánticos.

El conocimiento teórico de cada una de las fases que integran este Ciclo de Vida, nos deja una enorme expectativa en los cambios que se vendrán a futuro en el manejos de los Servicios Web como tal, así como el desarrollo de agentes software que favorezcan el trabajo automático de los Servicios Web y su evidente mejora para los buscadores en línea transformándose poco a poco en buscadores de significado semántico de información.

Una vez más se iniciará el enfoque hacia el modelado, implementación y pruebas de los Servicios Web Semánticos como núcleo de la presente Tesis, para ello, en el siguiente capítulo se ha realizado un análisis de varias herramientas que permiten el trabajo con Servicios Web Semánticos, enfocados algunos al trabajo y control secuencial de Servicios Web sintácticos con agregación semántica; y otros que proveen una plataforma completa para el modelado de conceptos, ontologías y servicios, permitiéndonos la implementación de los casos de uso que servirán para cumplir los objetivos de Tesis.

CAPITULO 4:

HERRAMIENTAS PARA EL TRABAJO CON SERVICIOS WEB SEMÁNTICOS

4.1. BPEL4WS

4.1.1. GENERALIDADES

BPEL4WS es un lenguaje que pretende ser tomando como estándar para la composición de Servicios Web, surge como la combinación de dos tentativas anteriores WSFL de la empresa IBM y XLANG de Microsoft; este lenguaje permite la creación de procesos complejos de negocios donde se requiera el desarrollo e invocación de Servicios Web, la manipulación de información, manejo de errores e interrelación entre Servicios. Combina lo mejor de WSFL y XLANG, tomando la interfaz gráfica de WSFL para la creación de procesos y el trabajo de construcción estructural de XLANG a manera de XML. BPEL4WS puede trabajar bajo dos distintos escenarios de utilización:

- Implementando procesos de negocio ejecutables.
- Describiendo procesos abstractos no ejecutables.

4.1.2. DETALLES TÉCNICOS

El comportamiento de esta herramienta consiste en recibir un mensaje, luego invocar el Servicio y finalmente responder al cliente, estas tres acciones son definidas en BPEL mediante la utilización de **actividades**, existe una colección de actividades primitivas como son:

- **<invoke>:** Para invocar una operación dentro de algún Servicio.
- **<receive>:** Donde se reciben los mensajes de operación que permiten trabajar con la interfaz del Servicio.
- **<wait>:** Indica una espera por un momento determinado.
- **<assign>:** Copiado de datos de un lugar o Servicio a otro.
- **<throw>:** Captura las instancias de error sucedidas durante los procesos.
- **<receive>:** Generación de respuestas de una operación de Entrada/Salida.
- **<terminate>:** Terminación completa de la instancia del Servicio.

Sin embargo, la relación entre estas actividades debe estar bien definida para saber cómo y cuándo ejecutarlas. Estas relaciones se definen en BPEL como actividades estructuradas

definiendo restricciones en la manera de ejecutar las actividades, la finalidad es definir que las actividades sucedan una tras otra mediante la utilización de ***<sequence>***.

La figura 25 muestra un modelo para la ejecución de un proceso de aprobación de un préstamo. Donde los sobres de color negro representan la solicitud del préstamo y los sobres rojos el mensaje que contiene la respuesta a dicha solicitud, el orden del proceso se indica con número sobre las flechas.

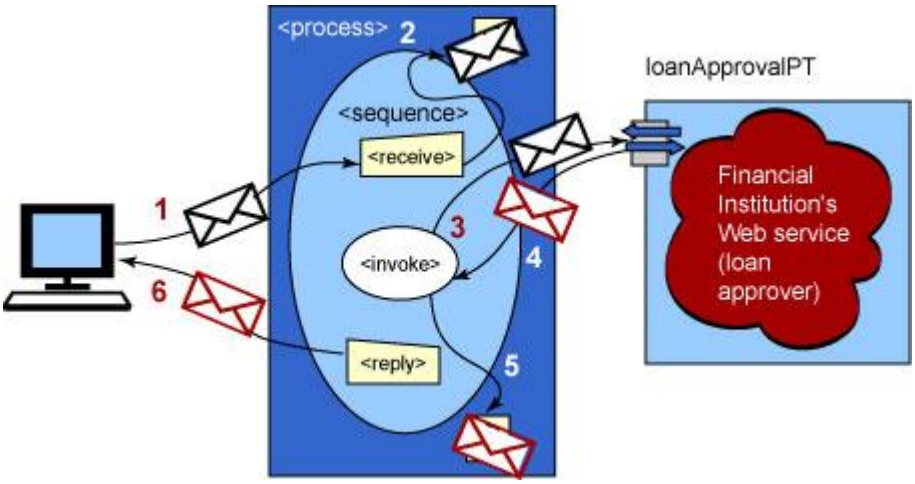


Figura 25. Ejecución BPEL4WS[43]

El editor BPWS4J permite la creación y ejecución de procesos basándose en el lenguaje BPEL4WS como proyectos BPWS, es una herramienta de descarga para trabajar bajo el entorno de Eclipse a manera de plug-ins. Esta perspectiva de los procesos de negocios se basa en los Servicios Web SOAP y sus descripciones WSDL.**[45]**

4.2. FRAMEWORK SWSF

4.2.1. GENERALIDADES

SWSF es una iniciativa desarrollada por W3C bajo la supervisión del *Semantic Web Services Language Committee*, utiliza como base para la descripción de Servicios Web el lenguaje WSDL 1.1, así como UDDI para la publicación de los mismos; adicionalmente incluye Semantic Web Services Lenguage (SWSL) y Semantic Web Service Ontology (SWSO).

4.2.2. DETALLES TÉCNICOS

SWSL se utiliza para especificar caracterizaciones formales de conceptos y descripciones individuales de los Servicios Web, trabajando con dos sub lenguajes SWSL-FOL basado en lógica (Fist-order Logic) y SWSL-Rules basado en reglas, mediante los cuales se propone

43 **Ejecución BPEL4WS:** Tomado del tutorial de la herramienta disponible en: [http://www.ibm.com/developerworks/webservices/library/ws-bpelcol2/]

trabajar con dos capas, las ontologías de los servicios y las reglas; permitiendo a los desarrolladores el intercambio y operación entre estas capas.

SWSO es un modelo conceptual mediante el cual los Servicios Web pueden ser descritos y puede crearse una representación formal (ontología) del modelo. La representación formal se realiza en SWSL-FOL especificando el significado preciso de los conceptos, posteriormente requiere de una traducción a las reglas SWSL-Rules.

Este concepto del SWSF se basa en el manejo de los Servicios Web creados y desarrollados en SOAP, puesto que su desarrollo incluye únicamente soporte para los documentos WSDL 1.1 y no se ha desarrollado nuevas versiones que soporten los documentos WSDL 2.0 que permiten la descripción de Servicios Web REST*ful.* **[46]**

4.3. FRAMEWORK WSMO

4.3.1. GENERALIDADES

WSMO es una base conceptual y un lenguaje formal para describir semánticamente todos los aspectos importantes de los Servicios Web con el propósito de realizar descubrimiento, combinación e invocación automática de Servicios en la Web.

WSMO tiene su base conceptual en WSMF y su objetivo ha sido mejorar y extender este framework desarrollando una ontología formal y un conjunto de lenguajes, WSMO posee fuertes principios de diseño necesarios para el acoplamiento con los estándares de la World Wide Web y las recomendaciones expuestas por el W3C.

Adicionalmente a sus principios de diseño WSMO se basa en los cuatro elementos principales del WSMF que son las Ontologías, los Servicios, las Metas y los Mediadores[44] como se muestra en la figura 26.

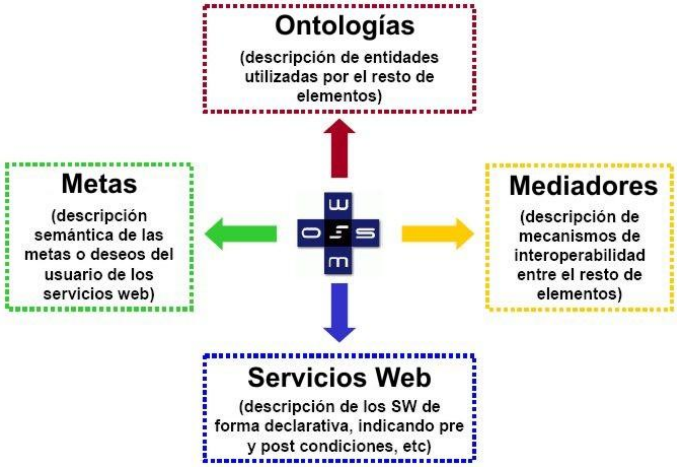


Figura 26. Conceptos de la Metodología WSMO[45]

44 Traducido de los conceptos WSMO: Ontologies, Goals, Web Services and Mediators

4.3.2. DETALLES TÉCNICOS

Como conceptos principales, WSMO requiere la incorporación de Ontologías, Servicios Web, Metas y Mediadores; los mismos que poseen las siguientes características:

- **Ontologías.** Proveen el significado de la terminología utilizada y son el elemento clave para el éxito de los Servicios Web Semánticos.
- **Servicios Web.** Realizan la descripción semántica de los métodos, parámetros de entrada y salida, capacidades, interfaces y funcionamiento interno de los Servicios Web tomando como base una Ontología declarada en WSMO.
- **Metas.** Describen los aspectos relacionados a las metas que el usuario desea lograr con sus peticiones, nuevamente tienen que utilizarse las ontologías para definir sin ambigüedad estas metas.
- **Mediadores.** Describe los diferentes mecanismos que manejan los problemas de interoperabilidad entre elementos como pueden ser las ontologías o servicios. Los mediadores son el núcleo para la solución de incompatibilidades en los datos, procesos y protocolos.

WSMO es en sí un conjunto de tecnologías para describir e interactuar con Servicios Web Semánticos, estas tecnologías incluidas en WSMO son:

- **WSMO.** Metodología conceptual para la descripción de los aspectos fundamentales de los Servicios Web Semánticos.
- **WSML.** Familia de lenguajes formales basados en diferentes formalismos lógicos para describir semánticamente los conceptos WSMO y los Servicios Web Semánticos.
- **WSMT.** Es un ambiente de desarrollo que permite el modelado de los conceptos WSMO bajo el lenguaje WSML y sus variantes.
- **WSMX.** Es el entorno de ejecución que permite la creación y composición de Servicios Web Semánticos basados en los conceptos WSMO.

---

45 Adaptado de: **[41]** Página 284

4.3.3. HERRAMIENTAS

El framework WSMO ofrece un conjunto de herramientas (WSMT) para el modelado de Servicios Web Semánticos, dentro de la cual nos permite la fácil elaboración de Ontologías, Servicios, Metas y Mediadores analizados anteriormente; este conjunto de herramientas también incluye:

- **WSML-Reasoner:** Funcionalidad que permite el razonamiento en base a una ontología seleccionada, trabajo sobre el lenguaje WSML.
- **WSML Discovery:** Permite seleccionar la una meta deseada y realizar el descubrimiento entre los Servicios Web almacenados.
- **Perspectiva Mapping:** Funcionalidad para realizar trabajos de mapeo entre dos Ontologías cliente y proveedor.
- **Perspectiva SEE:** Es un entorno de trabajo para la utilización de repositorios de Servicios como WSMX, IRS y otros.

La figura 27 muestra la herramienta WSMT y sus funcionalidades en la parte inferior del entorno de trabajo.

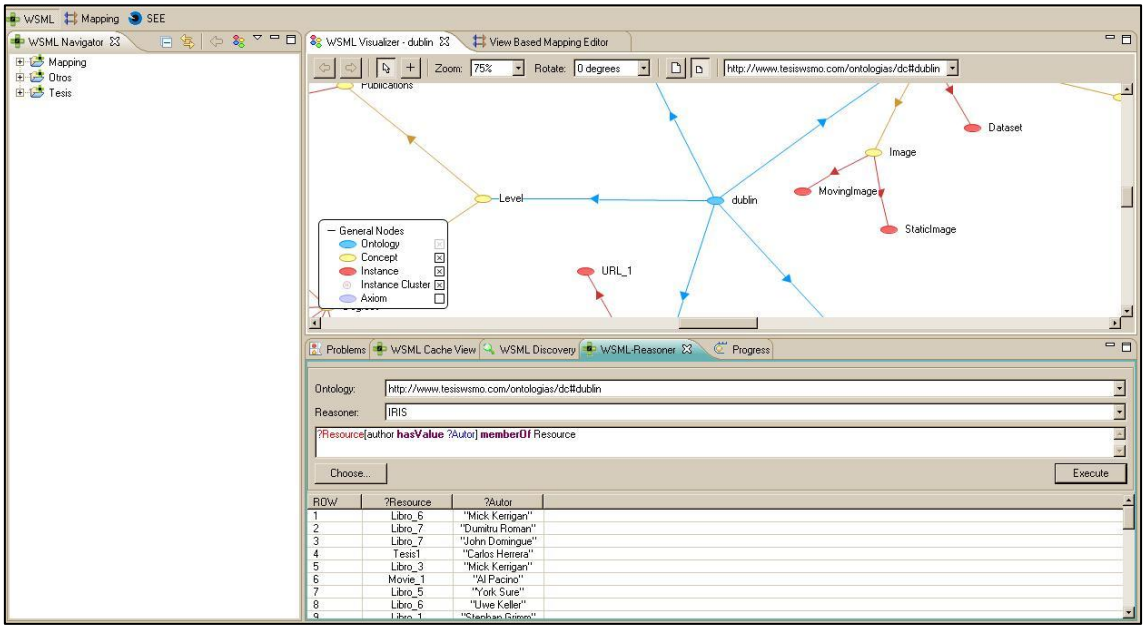


Figura 27. Entorno de trabajo WSMT

Asociado a los conceptos WSMO se incluyen los lenguajes WSML y el entorno de ejecución WSMX, este entorno de ejecución nos permitirá el registro de Servicios, Metas y Ontologías. A continuación la figura 28 muestra una imagen del entorno de ejecución.

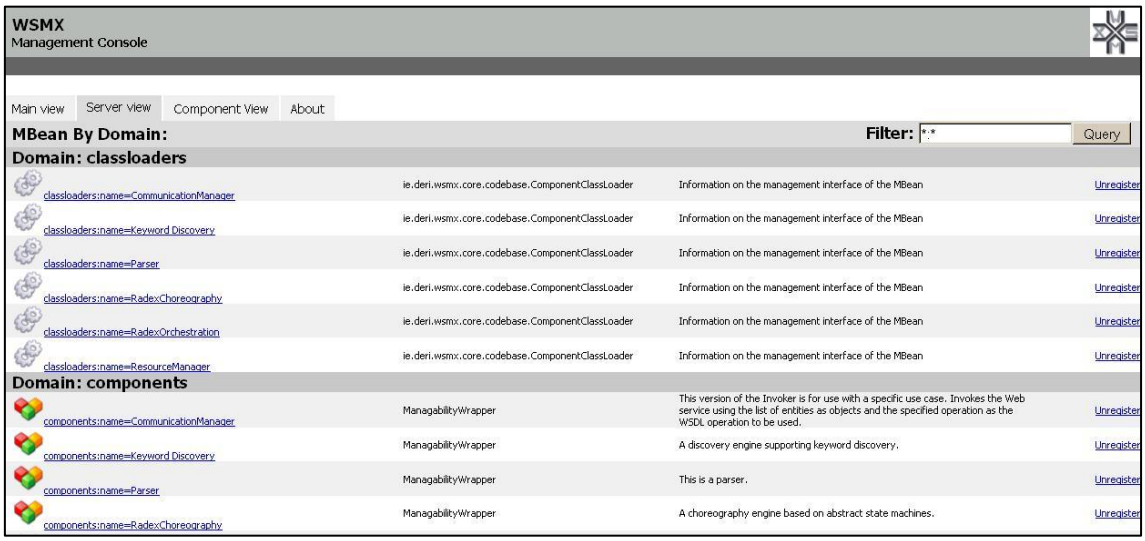


Figura 28. Vista de WSMX a través del Navegador

El conjunto de herramientas para modelado de Servicios Web WSMT presenta las siguientes características:

- Ambiente Integrado de Servicios de código abierto para Servicios Web Semánticos basado en el entorno de desarrollo Eclipse.
- Integra distintos plug-ins que cubren los diversos componentes de las especificaciones WSMO (edición y validación WSML, uso de razonadores, respositorios, etc).
- Interfaz Gráfica de Usuario para realizar la descripción de Metas, Ontologías, Mediadores y Servicios Web.
- Repositorio de ontologías, visualizador de ontologías mediante grafos, navegador para la estructura de las ontologías.
- Presenta continuas actualizaciones en sus versiones para adecuarse a los cambios en las especificaciones WSMO y WSML.[46] **[49][50]**

46 Una descripción sintetizada de las posibilidades del framework WSMO y sus herramientas, mayor detalle se puede encontrar en las fuentes **[49] [50] [59]**

4.4. FRAMEWOR IRS-III

4.4.1. GENERALIDADES

El descubrimiento e integración de servicios dentro de un ambiente de composición representa un conjunto de tareas complejas que deben realizarse adecuadamente, esta complejidad se debe a la ausencia de semántica en la descripción de los Servicios Web. IRS- III comprende una herramienta gráfica desarrollada en Java que soporta la creación de aplicaciones de Servicios Web Semánticos basándose en el modelado WSMO, permitiendo a los usuarios definir las composiciones dinámicas utilizando la recomendación de *metas* acorde al contexto solicitado en cada paso de la composición.

4.4.2. DETALLES TÉCNICOS

La arquitectura de la herramienta IRS-III está compuesta principalmente por tres componentes que se comunican a través del protocolo SOAP estos componentes son:

- **IRS Server** módulo encargado de la semántica de los Servicios Web mediante la descripción de componentes y el mapeo del Servicio.
- **IRS Publisher** es el encargado de generar un *wrapper* que permite utilizar Java o Lisp para invocar los Servicios Web o su descripción, además enlaza al Servicio Web con su descripción semántica en el IRS Server
- **IRS Client** permite la invocación y consulta de los Servicios Web almacenados.

Para lograr la composición de Servicios esta herramienta guía al usuario paso a paso en el proceso de composición seleccionando metas, mediadores y operadores de control de flujo. La composición se inicia con la selección de una primera meta seleccionada del listado provisto por el IRS Server.

Esta herramienta trabaja con la concepción general de que el usuario humano tenga el control de las definiciones en la composición, pero el trabajo laborioso como el descubrimiento de servicios acorde a las necesidades del usuario sea asumido por la máquina, durante el proceso de composición el usuario puede seleccionar más metas para lograr la conclusión requerida, cada una de estas metas servirá como entrada para el cumplimiento de una próxima meta. Una vez que se ha realizado la composición de Servicios la herramienta inicia el flujo de trabajo utilizando la API Java para la orquestación. **[47]**

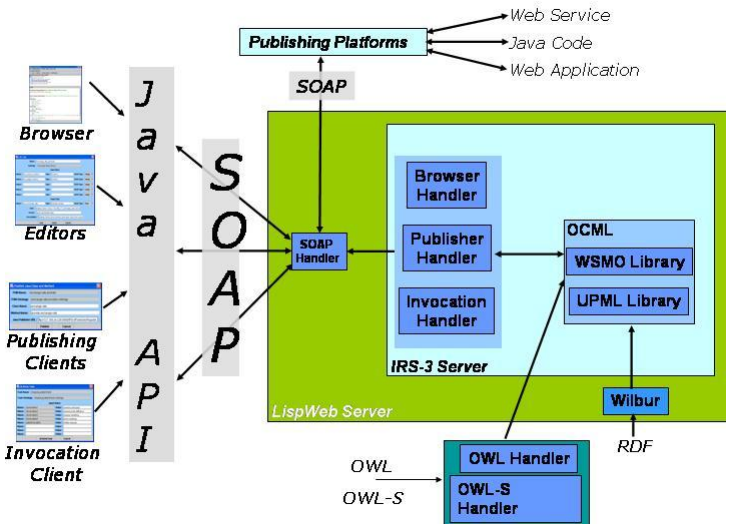


Figura 29. Framework IRS-III[47]

4.5. FRAMEWORK INFRAWEBS

4.5.1. GENERALIDADES

Es una herramienta de desarrollo, diseño y *goals*, consiste en diseñar, mantener y ejecutar Servicios Web Semánticos basándose en Servicios Web existentes diseñados acorde a WSMO Framework. Esta herramienta permite convertir de forma semiautomática los Servicios Web no semánticos a semánticos.

4.5.2. DETALLES TÉCNICOS

La arquitectura de INFRAWEBS basa su núcleo en la tecnología WSMO y por tanto la utilización de las herramientas WSMO, adicionalmente presenta los siguientes módulos:

- **Almacén Temporal** donde se guardan todos los ficheros producidos o utilizados por INFRAWEBS, entre estos ficheros se encuentran las descripciones WSML de todas las ontologías utilizadas para la descripción de servicios.
- **Navegador** es el módulo que permite el acceso a los demás módulos de INFRAWEBS y es responsable de crear el llamado Árbol de Servicios que nos permitirá navegar a través de los componentes del Servicio Web Semántico

47 **Framework IRS-III:** Tomado de [47] Pag. 6

- **Comunicador** es el responsable de la comunicación entre el usuario y los componentes de INFRAWEBS como son
- **OM** (Organitational Memory) utilizado para buscar descripciones de Servicios Semánticos similares al servicio en construcción.
- **DSWS-R** un almacén remoto de descripciones WSML de objetos semánticos y que se relaciona con el OM.
- **SIR** permite categorizar y anotar las descripciones de Servicios Web no Semánticos
- **Publicador** es el responsable de almacenar una descripción de SWS en un almacén externo para ser utilizado por otros servicios. **[47] [61]**

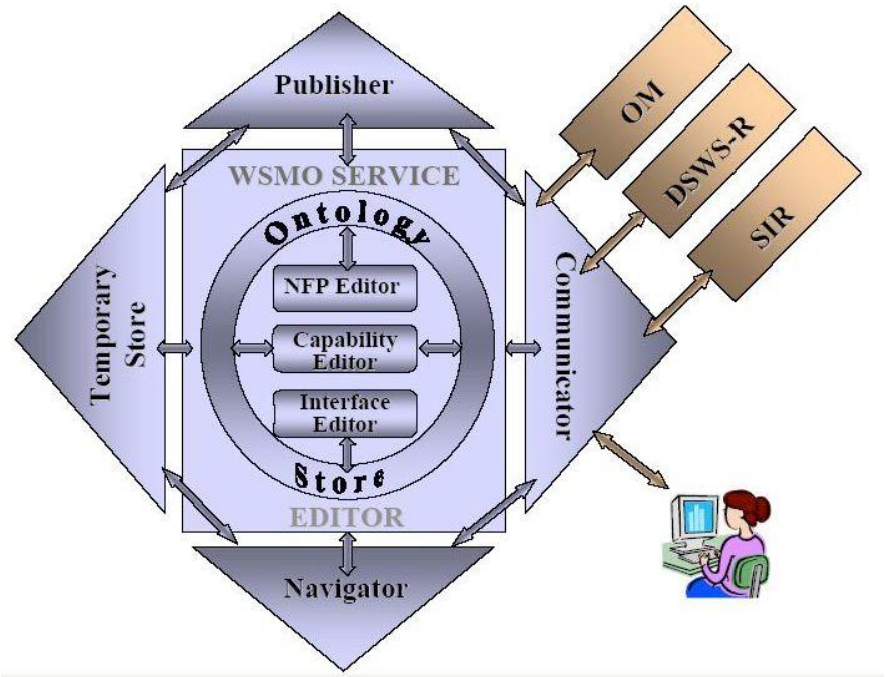


Figura 30. Esquema de trabajo INFRAWEBS[48]

4.6. CRITERIOS DE SELECCIÓN

Una vez analizados los diferentes frameworks y herramientas para trabajo con Servicios Web Semánticos es necesario realizar la selección de uno de ellos, el mismo que deberá permitirnos cumplir con el objetivo general y los objetivos específicos del presente proyecto.

Con el propósito de realizar una correcta selección de la herramienta se han tomado en cuenta varios criterios esenciales para el desarrollo de la presente investigación, buscando generar un

48 **Esquema Infrawebs:** Tomado de la guía Infrawebs designer en la fuente [61]

trabajo concreto, bien fundamentado y adicionalmente se espera conseguir información suficiente para mejorar las futuras investigaciones.

La Tabla 4.1. muestra los criterios de selección para los frameworks y herramientas antes descritas en este capítulo, dichos criterios serán ampliados posteriormente para fundamentar la selección de la herramienta con la que se trabajará la fase de pruebas de la investigación.

Tabla 4.1. Criterios de selección[49]

	FRAMEWORKS Y HERRAMIENTAS				
CRITERIOS	**SWSF**	**BPEL4WS**	**WSMO**	**IRS-III**	**INFRAWEBS**
CRITERIOS GENERALES					
Sugerido por W3C	SI	NO	SI	NO	NO
Soporta SW SOAP	SI	SI	SI	SI	SI
Soporta SW REST	NO	NO	SI	NO	NO
Soporta Ontologías RDF	NO	NO	SI	NO	NO
Soporta Ontologías OWL	NO	NO	SI	NO	NO
Permite el mapeo de ontologías	NO	NO	SI	NO	NO
CRITERIOS DE REFERENCIA					
Referencias de otros trabajos	Escaso	Poco	Varios	Varios	Escaso
Investigaciones enfocadas	Escaso	Poco	Varios	Poco	Escaso
Bibliografía	Escaso	Escaso	Varios	Varios	Escaso
CRITERIOS DE USO					
Herram. de modelado para SW	NO	SI	SI	SI	SI
H. de modelado para Ontologías	NO	NO	SI	SI	SI
Fácil manejo de la herramienta	NO	NO	SI	SI	SI
Ejemplos de uso	Escaso	SI	SI	Escaso	Escaso
Manuales de instalación	NO	SI	SI	Escaso	Escaso
Tutoriales de manejo	NO	Escaso	SI	Escaso	Escaso
Framework actualizado	NO	SI	2009	NO	NO

Mediante la tabla anterior se indican los principales criterios que han sido tomados en cuenta para la selección de una herramienta idónea para la conclusión de la presente investigación, cabe hacer hincapié en algunos puntos.

49 **Tabla de criterios de selección**: Criterios relacionados a la actualidad de la Web Semántica y los Servicios Web descritos en la primera fase, adicionalmente se toman criterios indispensables para el desarrollo de la investigación.

Soporte de SW SOAP/REST: Existe gran variedad de herramientas que permiten el trabajo con Servicios Web basados en SOAP, sin embargo, hemos sido testigos de la gran acogida de la metodología REST por lo cual es indispensable enfocarse en una herramienta que adicione esta posibilidad de trabajo.

Mapeo de ontologías: La innegable diversidad de Ontologías entre Servicios debe ser tomada en cuenta como un criterio primordial al momento de utilizar una herramienta, pues de esta manera tendremos un campo más amplio de trabajo.

Criterios de Referencia: Los principales criterios que se han tomado en cuenta son sobre líneas de investigación, trabajos formales y bibliografía relacionada al framework que deseamos seleccionar, pues de una u otra manera influenciará en el transcurso de la investigación.

Herramientas para SW y Ontologías: Se ha creído necesario tomar estos criterios debido a los objetivos planteados en el proyecto, pues es necesario realizar un caso de estudio que permita demostrar los conceptos de los Servicios Web Semánticos.

Framework actualizado: Se han analizado varias opciones que permiten el trabajo con Servicios Web Semánticos, sin embargo, algunas de ellas como es el caso de SWSF no han pasado de ser una línea teórica, BPEL4WS es únicamente expuesto en el portal de IBM y casos aislados, los casos de IRS-III e Infrawebs basan su núcleo en las teorías de WSMO. Mientras que, WSMO ofrece una herramienta Open Source (WSMT) para su descarga cuya última actualización -durante el transcurso de esta investigación- es de fecha 26 de Marzo de 2009.

En vista de los criterios planteados anteriormente, más la existencia de bibliografía como **[43], [44], [49]** y **[41]** dentro de las cuales se encuentran diversas explicaciones de los conceptos WSMO y una pequeña incursión en el uso de sus herramientas, se ha decidido orientar la investigación en el uso del framework WSMO y sus herramientas WSMT y WSMX, las mismas que serán expuestas con mayor detalle en secciones posteriores.

CAPITULO 5:

INVESTIGACIÓN Y MANEJO DE LA HERRAMIENTA WSMO

5.1. CONCEPTOS WSMO

Como se trató en el capítulo anterior la herramienta WSMO es una metodología basada principalmente en cuatro tipos diferente de conceptos Ontologías, Servicios Web, Metas y Mediadores; incluye también el lenguaje WSML para definición de dichos conceptos y herramientas de entorno gráfico para realizar un correcto modelado de los Servicios Web Semánticos.

WSMO proporciona WSMT, un conjunto de herramientas visuales y editores para la definición semántica de Servicios Web, Metas y Ontologías con el propósito de crear un completo ambiente de modelado que permita determinar el comportamiento de estos conceptos en un ambiente de producción real basada en WSMO.

La elaboración de cada uno de los conceptos WSMO requiere la selección de una de las variantes WSML tratadas en la sección 5.2., la variante WSML está asociada directamente a un tipo específico de razonador, el cual influye directamente en los resultados obtenidos al momento de realizar las consultas y el proceso de descubrimiento, pues éstas variantes WSML manejan diferentes grados de expresividad.

A continuación se expone más detalladamente cada uno de estos conceptos, los mismos que se han venido desarrollando durante la ejecución del caso de estudio generado para la comprobación de esta tesis.

5.1.1. ONTOLOGÍAS

Las ontologías son la piedra angular de esta metodología, pues, todos los demás conceptos requieren de su correcta definición para trabajar e interrelacionarse, las ontologías almacenan todos los conceptos necesarios a manera de superclases, clases y subclases, instanciación de dichas clases y atributos que otorgan una mayor expresividad en la declaración de conceptos.

La utilización de axiomas lógicos durante la definición de ontologías permite generar un razonamiento intrínseco, enriqueciendo de esta manera la información semántica expresada en la ontología; estos axiomas hacen uso de los conceptos, instancias y atributos declarados dentro de la misma ontología.

La herramienta WSML-Reasoner del conjunto de herramientas WSMT nos permite realizar consultas a la Ontología y de esta manera comprobar los diferentes conceptos e instancias que

se desarrollan, así como la correcta definición de sus axiomas; la sintaxis para el razonamiento basado en lenguaje WSML es similar a la sintaxis SQL, a continuación un par de ejemplos:

?Nombre_Concepto **memeberOf** `Concepto`
?Nombre_Concepto [`Atributo` **hasValue** ?Nombre_Atributo] **memeberOf** `Concepto`

Donde los campos ***?Nombre_xx*** es el nombre que recibirá el campo al momento de retornar la consulta, **Concepto** son las diferentes clases de conceptos definidas en la Ontología y **Atributo** son los diferentes atributos que posee un determinado Concepto.

5.1.2. SERVICIOS WEB

Para el trabajo adecuado con la metodología WSMO es necesaria la correcta definición semántica de los Servicios Web, estos servicios deberán poseer información ontológica de sus parámetros de entrada y su información de salida, así como sus interfaces para interacción y efectos de utilización.

La mayor parte de los Servicios Web utilizan varias operaciones y métodos que pueden ser invocados por el usuario, para realizar una definición semántica precisa de un Servicio Web de estas características WSMO V2.0. hace uso de capacidades (***Capability***), por tanto, cada una de las operaciones o métodos deberán ser descritas por separado apuntando hacia la misma URL del Servicio o del documento WSDL.

El manejo de **Interfaces** dentro de la descripción de un Servicio Web define la interfaz por la cual el Servicio Web puede ser invocado, es decir, en este punto se puede hacer distinción de la metodología de desarrollo a ser utilizada SOAP/REST, adicionalmente, la descripción del Servicio deberá contener las Precondiciones y Postcondiciones que definen los tipos de Entrada/Salida del Servicio Web.

Como punto inicial en el proceso de descubrimiento de un Servicio Web WSMO se utiliza las NFP (***No Functional Porpoerties***) que permiten realizar un reconocimiento de las palabras clave que describen el funcionamiento del Servicio, es por tanto primordial su utilización para obtener mejores resultados en el proceso de búsqueda de Servicios.

Estas definiciones semánticas de los Servicios Web serán utilizadas posteriormente para el descubrimiento del mejor Servicio en base a una Meta determinada y trabajando con la Ontología correspondiente.

5.1.3. METAS

Otro concepto esencial en la metodología WSMO es la definición de las Metas (**Goals**), las mismas que son utilizadas para describir semánticamente los objetivos que deseamos alcanzar

mediante la utilización de un Servicio; en la definición de las metas se requiere también la importación de la Ontología de referencia y la selección de una variante de lenguaje WSML.

Una meta consta de la definición de capacidades, precondiciones y postcondiciones, mediante las cuales se define los parámetros que se enviarán al Servicio Web y la información que esperamos recibir como resultado de la interacción con un determinado Servicio.

Con la definición de las Metas podemos utilizar la herramienta WSML Discovery incluida en el WSMT, la cual nos permite ejecutar el código WSML con el cual hemos definido la Meta, posteriormente, realiza una comparación de aquellos Servicios Web que trabajan con la Ontología que utilizamos en nuestra Meta y realiza una búsqueda en la coincidencia de parámetros, devolviéndonos como resultado final un listado de Servicios y su grado de coincidencia con la Meta definida. **[49]**

La implementación y resultado del descubrimiento de Servicios Web mediante el uso de la herramienta WSMT se expone posteriormente en la sección 6.2.3., así como en el Anexo 1 donde se incluye el código de los diferentes conceptos WSMO utilizados en el presente proyecto.

5.1.4. MEDIADORES

Los Mediadores son definiciones complementarias al descubrimiento de los Servicios Web, permiten realizar la interacción entre servicios web, ontologías y metas; definen similitudes entre los conceptos declarados en cada instancia.

Los tipos de Mediadores que WSMO posee son:

- **OO-Mediator:** Realiza la mediación entre ontologías, toma los conceptos de una ontología y los relaciona con los conceptos de la segunda ontología.
- **WW-Mediator:** Trabaja con los conceptos utilizados en dos Servicios Web diferentes para realizar su comparación y declaración de similaridad.
- **WG-Mediator:** Realiza la mediación entre un Servicio Web y una meta, este tipo de mediadores son utilizados para mejorar el descubrimiento de Servicios.
- **GG-Mediator:** Permite la creación de similitudes entre dos metas.

La utilización de los mediadores está directamente relacionada con la elaboración de mapeos entre ontologías diferentes (**Mapping**) que pueden ser utilizadas para equiparar conceptos WSMO que utilizan diferente Ontologías.[50]

50 Conceptos y procedimientos tomados del Tutorial SESA disponible en **[49]**, adicionalmente se ha iniciado con el uso de la herramienta y la comparación de conceptos investigados

5.1.5. MAPPING

La herramienta Mapping incluida en el WSMT nos permite realizar un mapeo entre dos ontologías, una de las cuales será la ontología recurso y la otra será la ontología objetivo; mediante el mapeo de ontologías se puede definir como sinónimos dos conceptos declarados de manera diferente en las ontologías.

Un ejemplo de mapeo de conceptos se muestra en la figura 20, donde la ontología del recurso posee definido el concepto moneda (currency), mientras que la ontología objetivo no posee el mismo concepto, sin embargo, tiene declarados Dolares y Euros como métodos de pago, para este caso ambos conceptos son similares.

El mapeo de ontologías es utilizado para el trabajo con varios Servicios que no utilizan la misma ontología, y de esta manera se ofrece un entorno más amplio de modelado y desarrollo de Servicios Web, la utilización del mapeo es incorporada a los Mediadores, los mismos que amplían la capacidad de descubrimiento de Servicios al involucrar conceptos mejorados en base al mapeo de Ontologías. La figura 31 muestra la vista ***Mapping*** de la herramienta WSMT.[51]

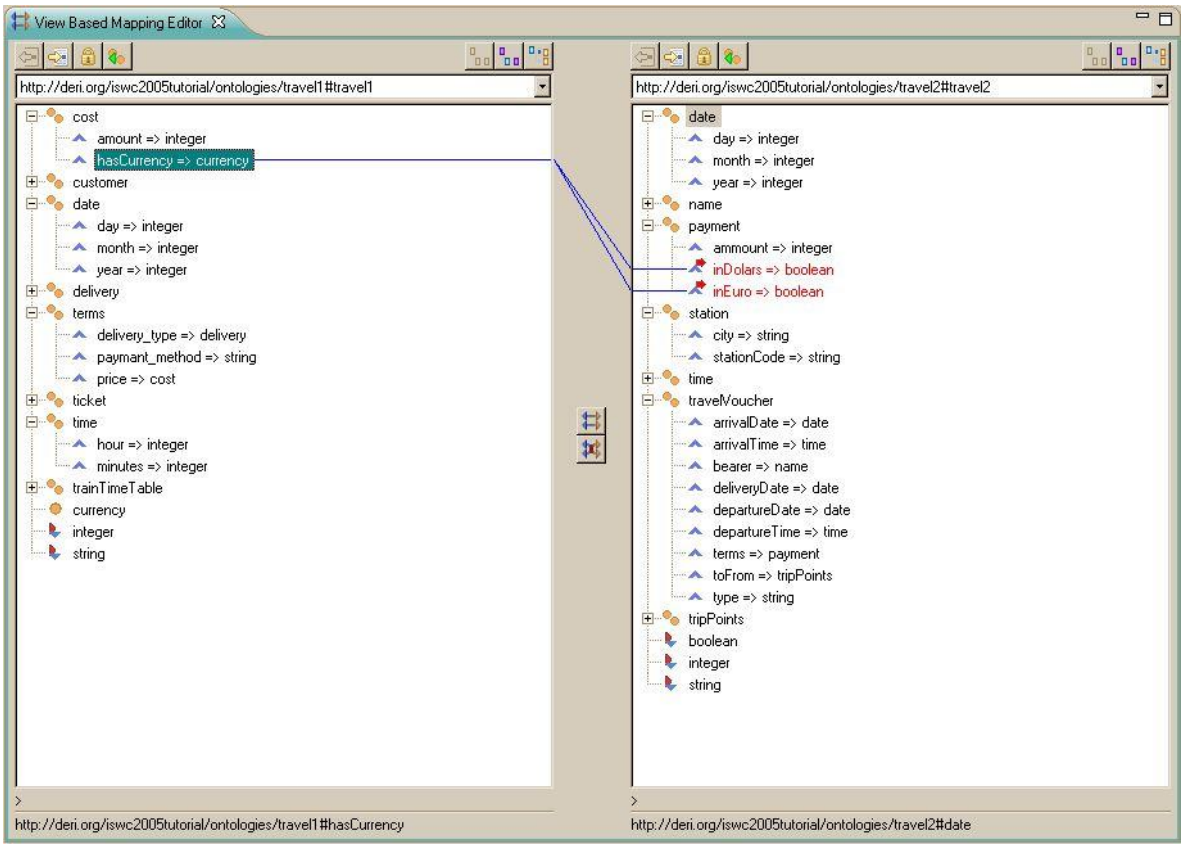


Figura 31. Vista Mapping de la herramienta WSMT

51 Ejemplo tomado del Tutorial SESA **[49]** y analizado en la herramienta WSMT

5.2. LENGUAJE WSML

El lenguaje WSML es el núcleo del trabajo con la metodología WSMO, pues, provee una sintaxis formal y semántica para la definición de todos los conceptos Ontologías, Servicios, Metas y Mediadores; además, es utilizado para el razonamiento y la declaración de axiomas lógicos para el enriquecimiento de los conceptos.

WSML está basado en diferentes lógicas formales, llamadas: Lógica de descripción (Description Logics), Lógica de Primer Orden (First-Order Logic) y Lógica de Programación (Logic Programming), que se usan para el modelado de Servicios Web Semánticos. WSML consta de un número de variantes basadas en estas diferentes lógicas formales llamadas WSML-Core, WSML-DL, WSML-Flight, WSML-Rule y WSML-Full.

WSML-Core es el núcleo del lenguaje y las otras variantes WSML ofrecen incrementos de expresividad en la dirección de la Descripción Lógica y la Lógica de Programación. Por último, ambos paradigmas se unifican en WSML-Full, la variante más expresiva de WSML.

WSML se especifica en términos de una sintaxis normativa legible por seres humanos. Pero además permite la importación/exportación de RDF y OWL para el intercambio sobre la web y para la interoperabilidad con aplicaciones basadas en RDF.

La figura 32 muestra las diferentes variaciones de WSML y las relaciones que se establecen entre cada una de ellas, el motivo de que existan estas variaciones, es para permitir a los usuarios elegir entre la expresividad que necesitan en sus aplicaciones y su evidente grado de complejidad. La herramienta WSMT incorpora diferentes razonadores para cada variante WSML lo que influye directamente en el proceso de descubrimiento de Servicios, retornando diferentes resultados que dependen de la variante WSML utilizada.[52]

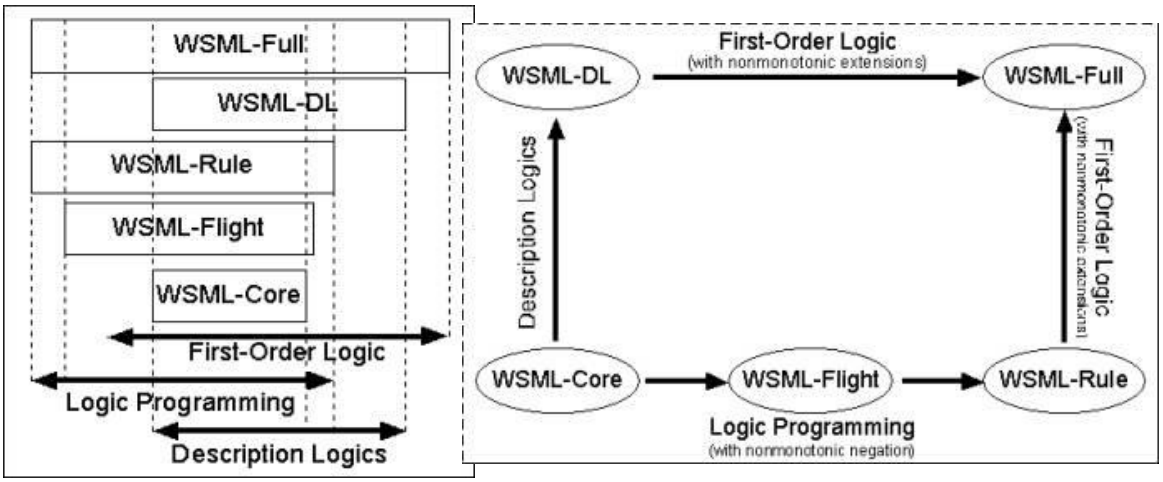


Figura 32. Variantes del lenguaje WSML[53]

52 Conceptos resumidos que se han extraído de las fuentes **[51]** y **[52]** donde se podrá ampliar dichos conceptos.

53 **Variantes WSML:** Disponible en: **[52]** Pag. 84. Contrastado con el documento W3C **[69]**

5.2.1. DECLARACIÓN DE ENTIDADES[54]

5.2.1.1. Ontologías

Para especificar una Ontología en WSML se hace uso de la palabra *ontology* que es seguida opcionalmente de un nombre o URL, la cual sirve para identificar de manera univoca a la ontología.

Las ontologías se conforman de conceptos, subconceptos, instancias y los atributos relacionados a estos conceptos e instancias, además se pueden incluir relaciones y axiomas para darle mayor complejidad a la definición de la ontología.

En la figura 33 se toma como un breve ejemplo de la definición de ontología una parte del caso de estudio del proyecto de tesis.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace { _"http://www.tesiswsmo.com/ontologias/dc#",
     dc _"http://purl.org/dc/elements/1.1/" }
ontology dublin

axiom Agent_ResourceAuthor
……
……
concept ISBN subConceptOf Identifier
     ISBN ofType _string
     valid ofType _boolean
……
……
instance Paper memberOf Text
……
……
```

Figura 33. Definición de ontologías[55]

5.2.1.2. Servicios Web

La especificación de un Servicio Web se identifica en WSML con la palabra *webService* seguida de un nombre o URL que lo identifica de manera única, así también, el campo *namespace* es aquel que permite definir el punto de entrada hacia el Servicio o Método que puede ser también el documento WSDL o una URL para acceder por medio de REST.

54 Con el uso de la herramienta WSMT y el estudio de los conceptos WSMO citados en las fuentes **[49] [50] [51]** y **[52]** se puede describir claramente las necesidades y utilización de dichos conceptos

55 **Definición de ontologías:** Resumen de la sintaxis WSML tomada del caso de estudio de la Tesis

Para ampliar la información de un Servicio Web con el propósito de mejorar las posibilidades de descubrimiento se incluirán sus interfaces, capacidades, precondiciones y postcondiciones necesarias para interactuar con dicho Servicio Web.
La figura 34 muestra un ejemplo del código requerido para la definición de un Servicio Web sencillo tomado del caso de estudio del proyecto de tesis.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace { _"http://www.tesiswsmo.com/servicios#",
     discovery _"http://wiki.wsmx.org/index.php?title=DiscoveryOntology#",
     dcore _"http://www.tesiswsmo.com/ontologias/dc#" }
webService ResourceVendor
     importsOntology {dcore#dublin}

.....
.....
sharedVariables ?x

precondition ResourcePre
     definedBy
          ?x[dcore#title hasValue ""] memberOf dcore#Resource.

.....
.....
postcondition ResourcePost
     definedBy
          ?x[dcore#type hasValue dcore#Book, dcore#type hasValue
             dcore#Event, dcore#type hasValue dcore#Software] memberOf
             dcore#PhysicalResource.
```

Figura 34. Definición de un Servicio Web[56]

5.2.1.3. Metas

La especificación de una Meta se identifica en WSML con la palabra *goal,* seguida de un nombre o URL que la identifica. La descripción de una Meta incluye la descripción de precondiciones y postcondiciones requeridas, estas precondiciones y postcondiciones nos servirán para realizar posteriormente el descubrimiento de los Servicios Web más idóneos.

Dentro de la definición de una Meta se realiza la importación de la Ontología que será utilizada para el descubrimiento de los conceptos y atributos ontológicos, mediante la utilización de esta Ontología se asociará a los diferentes Servicios Web declarados en la herramienta WSMT y que tienen relación con lo solicitado en la Meta.

La figura 35 muestra una fracción del código WSML requerido para la declaración de una Meta sencilla tomada del caso de estudio de este trabajo.

56 **Definición de SW:** Sintaxis WSML extraída mediante el uso de la herramienta WSMT

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace { _"http://www.tesiswsmo.com/metas#",
     discovery _"http://wiki.wsmx.org/index.php?title=DiscoveryOntology#",
     dcore _"http://www.tesiswsmo.com/ontologias/dc#" }

goal RecursoFisico
importsOntology {dcore#dublin}
.....
.....

sharedVariables ?x

postcondition InteractivoPost
     definedBy
          ?x[dcore#needUser hasValue _boolean("false")] memberOf
            dcore#PhysicalResource.
```

Figura 35. Definición de una Meta[57]

5.2.2. AXIOMAS LÓGICOS

Los axiomas lógicos son utilizados para el razonamiento interno de las Ontologías, así como de los Servicios y Metas; el uso de los axiomas facilita el razonamiento interno del lenguaje WSML, es decir generar información nueva a partir de la información ingresada, para declarar un axioma lógico se hace uso de la palabra *axiom*, se realiza la definición del axioma incluyendo los elementos ontológicos que lo forman y posteriormente se incluye la implicación que genera el axioma lógico.

Los axiomas son comúnmente utilizados en la implementación de las Ontologías, pues mejoran el razonamiento de las mismas, permitiendo agilizar la información el momento de ser consultada; para mostrar un ejemplo de los axiomas se toma el ejemplo del axioma `Agent_ResourceAuthor` del caso de estudio desarrollado en la presente tesis.

La definición del axioma incluye un elemento *?x* miembro del concepto *Agent* posee un nombre *?y* y un elemento *?z* miembro del concepto *Resource* que posee como instancia de Autor al elemento *?x;* este axioma concluye con la implicación de que el recurso *?z* poseerá como nombre de autor el elemento *?y* atributo de *?x.*

El código de implementación del axioma lógico descrito anteriormente se muestra en la figura 36.

```
axiom Agent_ResourceAuthor
     definedBy
          ?x[name hasValue ?y] memberOf Agent
  and ?z[authorInstance hasValue ?x] memberOf Resource
implies
?z[author hasValue ?y] memberOf Resource.
```

Figura 36. Definición de un axioma (WSML)

[57] **Definición de metas:** Sintaxis WSML tomado del caso de estudio de esta Tesis

5.2.3. PRECONDICIONES Y POSTCONDICIONES

Para obtener una definición exitosa de Servicios y Metas es necesario involucrar las funcionalidades deseadas que en WSML de describen a través de capacidades (capability). Una capacidad deseada es parte de una meta y una capacidad proporcionada es parte de un Servicio Web; la forma y estilo en que el solicitante y el proveedor de la capacidad interactúan es descrita a través de una interface, que a su vez forma parte también de una Meta o Servicio.

Según esto, una capacidad constituye una descripción formal de una funcionalidad solicitada desde o proporcionada por un Servicio Web. Las precondiciones describen condiciones en la entrada del servicio, mientras que las postcondiciones describen la relación entre la entrada y la salida del Servicio; una Meta o Servicio Web en WSML solo puede tener una única capacidad, siendo opcional su especificación.

Para realizar la definición de una capacidad se utiliza la palabra ***capability*** seguida del nombre de la capacidad, un bloque de importación de la ontología con la que se está trabajando ***importsOntology,*** opcionalmente puede agregarse una descripción general mediante el uso de las propiedades no funcionales ***nonFunctionalProperties***, y un bloque para la utilización de mediadores ***usesMediator***.

El bloque denominado ***sharedVariables*** se utiliza para indicar las variables que se compartirá entre la precondición y la postcondición, que se definen mediante ***precondition*** y ***postcondition*** respectivamente. No hay limitación en el número de precondiciones y postcondiciones que se pueden definir, y al igual que las capacidades pueden incluir su propio bloque de propiedades no funcionales. **[49]**

5.2.4. COREOGRAFÍA Y ORQUESTACIÓN

Como se ha definido anteriormente la coreografía y orquestación determinan el comportamiento de un Servicio Web en relación con otros Servicios o invocaciones, para esto, se hace uso de la descripción de una interface que servirá como punto de partida para la definición de estos elementos.

La descripción de una Meta puede solicitar múltiples interfaces y un Servicio Web puede ofrecer múltiples interfaces, sin embargo, la especificación de las interfaces es opcional. La descripción de una interface inicia con la palabra *interface* seguida por un identificador, además puede incluir varios bloques opcionales como propiedades no funcionales para su descripción, importación de ontologías, uso de Mediadores, Coreografía *choreography* y Orquestación *orchestration*.

Cabe destacar que la orquestación y coreografía son únicamente enlaces URL hacia contenidos web externos y que WSML aún no soporta su definición explícita. Mediante la figura 37 se muestra un ejemplo para la definición de una interface.[58]

```
  interface
    choreography _"http://example.org/mychoreography"
    orchestration _"http://example.org/myorchestration"

  interface {_"http://example.org/mychoreography",
_"http://example.org/mychoreography"}
```

Figura 37. Definición de una interface [59]

5.3. METODOLOGÍAS DE DESARROLLO

El trabajo con Servicios Web Semánticos obliga a la utilización de un medio de comunicación entre los Servicios que vayan a ser utilizados, para esto se enfoca la información proporcionada en las secciones 1.2.1.1. y 1.2.1.2. para el uso con el framework WSMO y sus herramientas WSMT y WSMX.

Como se ha visto con anterioridad, los avances más significativos en el área de los Servicios Web Semánticos se han enfocado principalmente en la metodología de desarrollo SOAP, sin embargo, es necesario enfatizar también en RESTful que ha tenido una gran acogida en los últimos años sirviéndose de los proyectos y meta lenguajes representativos de SOAP como son SAWSDL para trabajar de manera similar con REST utilizando su propio meta lenguaje como es el caso de SA-REST que trabaja de manera muy similar.

La figura 38 muestra de manera general la integración de estas metodologías con la utilización de WSMO.

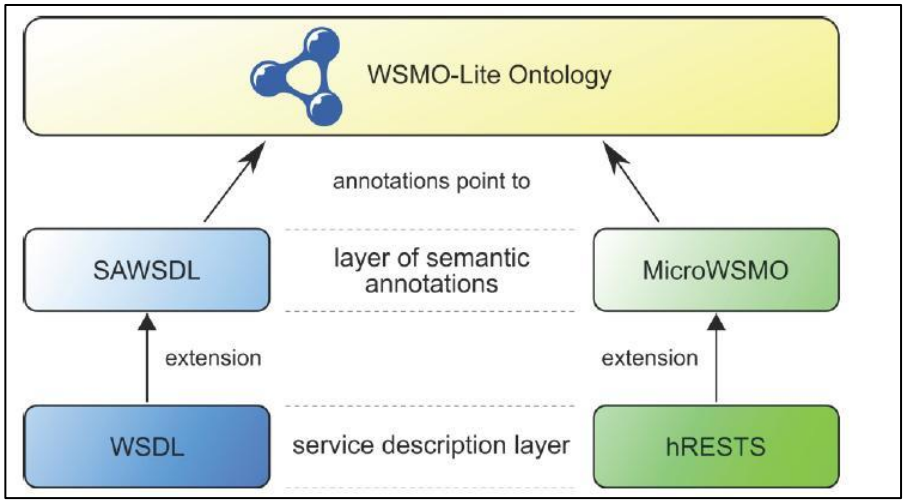


Figura 38. WSMO con REST y SOAP[60]

58 Principalmente se han tomado las ideas expuestas sobre Composición Orquestación y Coreografía de servicios tomado de las fuentes **[43] [49]** y **[44]**

59 **Definición de interface:** Sintaxis WSML tomada de [http://www.w3.org/Submission/WSML/].

5.3.1. hREST, WSMO-Lite y MicroWSMO

Para realizar una descripción de los Servicios Web basados en REST es necesario tener en cuenta que para este tipo de Servicios el núcleo son las URI's, pues cada listado, cada objeto o conjunto de propiedades está identificado inequívocamente por una dirección URI, es decir, un recurso.

Estos recursos generalmente hacen uso del lenguaje XHTML y están enlazados entre sí, para lograr una descripción semántica de estos recursos se ha creado un microformato llamado HTML RESTful o simplemente hREST el cual se utiliza para adicionar información semántica a dichos recursos. Esta información puede obtenerse y organizarse mediante el uso de RDFa, una recomendación W3C que especifica un mecanismo para incluir información RDF dentro de los documentos HTML.

La interacción de un cliente con un Servicio RESTful puede tratarse como un conjunto de operaciones donde el cliente envía requerimientos hacia un recurso, utilizando uno de los métodos HTTP, GET, POST, PUT o DELETE[61] y recibe una respuesta, la misma que no es más que un enlace hacia otro recurso.

Los conceptos claves para los Servicios RESTful son:

- **Servicio:** Es el conjunto de recursos relacionados con el cual el cliente trata.
- **Operación:** Es una acción simple que el cliente puede invocar sobre el Servicio.
- **Recurso:** Una dirección URI determinada donde se invoca la operación.
- **Método:** Captura el método HTTP utilizado para la operación.
- **Petición y Respuesta:** Son los mensajes enviados como entrada y salida de una operación.
- **Enlace:** Especialmente encontrado en los mensajes devueltos que dirigen al usuario hacia un recurso relacionado.

De esta manera se crea un documento descriptivo similar a los documentos WSDL utilizados para los Servicios SOAP donde se incluye la información esencial de cada recurso REST, sin embargo, es necesario agregar información semántica a este documento, tarea que se realiza a cabo mediante la utilización de MicroWSMO.

---

60 **WSMO con REST y SOAP:** Adaptado de: **[53]** Pag. 14 su funcionamiento se detalla en la sección 5.3.1.

61 Métodos se describen con detalle en el apartado 1.2.1.2.

MicroWSMO realiza la captura de información semántica utilizando los conceptos esenciales de WSMO-Lite para los aspectos semánticos de los Servicios.

- **Modelos de Información:** Dominio de la Ontología donde se representan los datos y especialmente los mensajes de Entrada y Salida.
- **Semántica Funcional:** Especifica ***que*** es lo que el Servicio Web realiza a través de las precondiciones y efectos.
- **Semántica de comportamiento:** Define la secuencia para la invocación de Operaciones y Servicios.
- **Descripciones No Funcionales:** Representa las políticas del Servicio y otros detalles específicos para implementar y ejecutar un Servicio.

La figura 39 muestra un ejemplo de la descripción semántica de un recurso con la utilización de MicroWSMO.[62]

```
<div class="service" id="svc">
      <h1><span class="label">ACME Hotels</span> service API</h1>
      <p>This service is a
      <a rel="model" href="http://example.com/ecommerce/hotelReservation">
      hotel reservation</a> service.
      </p>
<div class="operation" id="op1">
      <h2>Operation <code class="label">getHotelDetails</code></h2>
      <p> Invoked using the <span class="method">GET</span>
      at <code class="address">http://example.com/h/fidg</code><br/>
      <span class="input">
      <strong>Parameters:</strong>
      <a rel="model" href="http://example.com/data/onto.owl#Hotel">
      <code>id</code></a>  the identifier of the particular hotel
      <a rel="lowering"
      href="http://example.com/data/hotel.xsparql">lowering</a>)
      </span><br/>
      <span class="output">
      <strong>Output value:</strong> hotel details in an
      <code>ex:hotelInformation</code> document
      </span>
      </p>
</div>
</div>
```

Figura 39. Ejemplo de código MicroWSMO[63]

62 La exposición de conceptos descrita en este apartado se basan en las definiciones y ejemplos tratados en **[53]** y **[54]** donde se podrá ampliar dicha información

63 **Código MicroWSMO:** tomado del documento MicroWSMO & REST disponible en: [http://sweet.kmi.open.ac.uk/pub/microWSMO.pdf]

5.4. UTILIZACIÓN DE LA HERRAMIENTA[64]

Una vez realizado un análisis de los conceptos WSMO y la sintaxis general del lenguaje WSML, se ha procedido a la descarga e instalación de los paquetes necesarios para el trabajo con WSMO, WSML, WSMT y WSMX.

La metodología WSMO nos indica la manera teórica de trabajar con los conceptos relacionados a los Servicios Web Semánticos (Ontologías, Servicios, Metas y Mediadores), para la definición, manejo e interacción de estos conceptos se requiere la utilización del lenguaje WSML.

El conjunto de herramientas WSMT incorpora el lenguaje WSML así como varias herramientas de modelado y razonamiento; el entorno de ejecución WSMX trabaja independientemente en un ambiente web. Todos estos conceptos se manejan con el lenguaje de programación Java por lo cual es necesario cumplir con la instalación de varios prerrequisitos los cuales se detallan en el Anexo 2 de este documento.

5.4.1. HERRAMIENTA WSMT

WSMT es un conjunto de herramientas que permiten el modelado de todos los conceptos relacionados con WSMO, ofreciendo varias herramientas gráficas para la creación de Ontologías, Servicios, Metas y Mediadores; adicionalmente incluye editores de texto que permiten reconocer los errores al momento de ir generando el código de los diferentes conceptos WSMO, todos estos conceptos pueden ser generados a partir del entorno gráfico, no obstante, la utilización de los editores de texto permiten manejar de manera más avanzada los atributos y axiomas principalmente.

Esta herramienta Open Source está disponible en http://sourceforge.org/projects/wsmt/ de donde se han utilizado las versiones 1.4.1. y finalmente la versión 2.0 que ha sido publicada el 26 de Marzo de 2009 y es la versión de la herramienta que ha sido utilizada para el desarrollo de los diferentes casos de estudio de la presente investigación.

El proceso de descarga e instalación de la herramienta así como los problemas encontrados y solucionados se detallan en el Anexo 4 de este documento, si la instalación ha sido correcta

64 Como base para el uso de WSMT se ha tomado el Tutorial SESA **[50]** aplicando a la definición y manejo de las ontologías Dublin Core y BIBO **[55] [57]** respectivamente

podemos empezar a trabajar con el entorno para modelado de Servicios Web Semánticos WSMT que se muestra en la figura 40.

Figura 40. Entorno de trabajo de la herramienta WSMT[65]

5.4.2. CASO DE ESTUDIO PARA LA CREACIÓN DE ONTOLOGÍAS

Para lograr el objetivo general de la Tesis, la implementación de un caso de estudio que permita realizar el descubrimiento de Servicios Web; se han tomado como referencia las ontologías Dublin Core[66] y la ontología BIBO[67], obteniendo como resultado una ontología relacionada a los recursos educativos, tomando los conceptos más significativos y contrastándolos entre las mencionadas ontologías y de esta manera conformar una ontología única que contemple los conceptos más utilizados sin extender demasiado las propiedades y atributos de los mismos.

La figura 41 muestra el diagrama de flujo que ha permitido la elaboración de la ontología resultante de contrastar Dublin Core y BIBO trabajando de la siguiente manera:

1. **Identificar concepto:** Se trata de determinar los conceptos que servirán de base para la ontología de recursos educativos utilizado para nuestro caso de estudio; cada uno de estos conceptos han sido extraídos de las ontologías Dublin Core y BIBO y comparadas entre sí para obtener un único concepto de mejor entendimiento.

65 **Entorno de trabajo WSMT:** Instalación y configuraciones descritas en el Anexo 2

66 **Dublin Core:** Ontología de metadatos estándares tomado de [http://dublincore.org /]

67 **BIBO Ontology:** Ontología Bibliográfica elaborada bajo la supervisión de Google disponible en [http://bibotools.googlecode.com/svn/bibo-ontology/trunk/doc/index.html]

2. **¿Es nodo final?:** En este punto se determinará si el concepto seleccionado forma parte de un grupo o Superconcepto, que permitirá categorizar de mejor forma nuestra ontología obtenida; como en el caso de estudio del Superconcepto "*Recurso"* posee dos concpetos inferiores *Recurso Digital* y *Recurso Físico*.

3. **NO es nodo final (concepto):** En caso de ser un concepto regresamos a la condición para buscar las categorías (Superconceptos) al cual pertenece el concepto, vinculándolo y heredando los atributos del Superconcepto.

4. **ES nodo final (instancia):** Al descubrir un nodo final, es decir una instancia, concepto final que no posee elementos inferiores a este y por tanto se ha llegado al final de la definición de conceptos; en nuestro caso de estudio los nodos finales *Video* y *Web* son instancias del concepto *Recuro Digital.*

5. **Analizar atributos:** Una vez determinados las instancias finales procedemos a determinar si el tipo de atributo es un concepto o un tipo de dato general.

6. **NO es concepto:** Deberemos asignarle un tipo de dato, que podrá ser String, Date, Boolean, etc.

7. **ES concepto:** Cuando un atributo se vincula a un concepto es necesario relacionarlo con una instancia de los conceptos mediante la opción ***Add Concept*** de la herramienta gráfica WSMT[68]

[68] Descripción del proceso realizado durante la investigación e implementación de la Ontología base para los casos de estudio de esta Tesis, tomando los fundamentos de las ontologías Dublin Core y BIBO

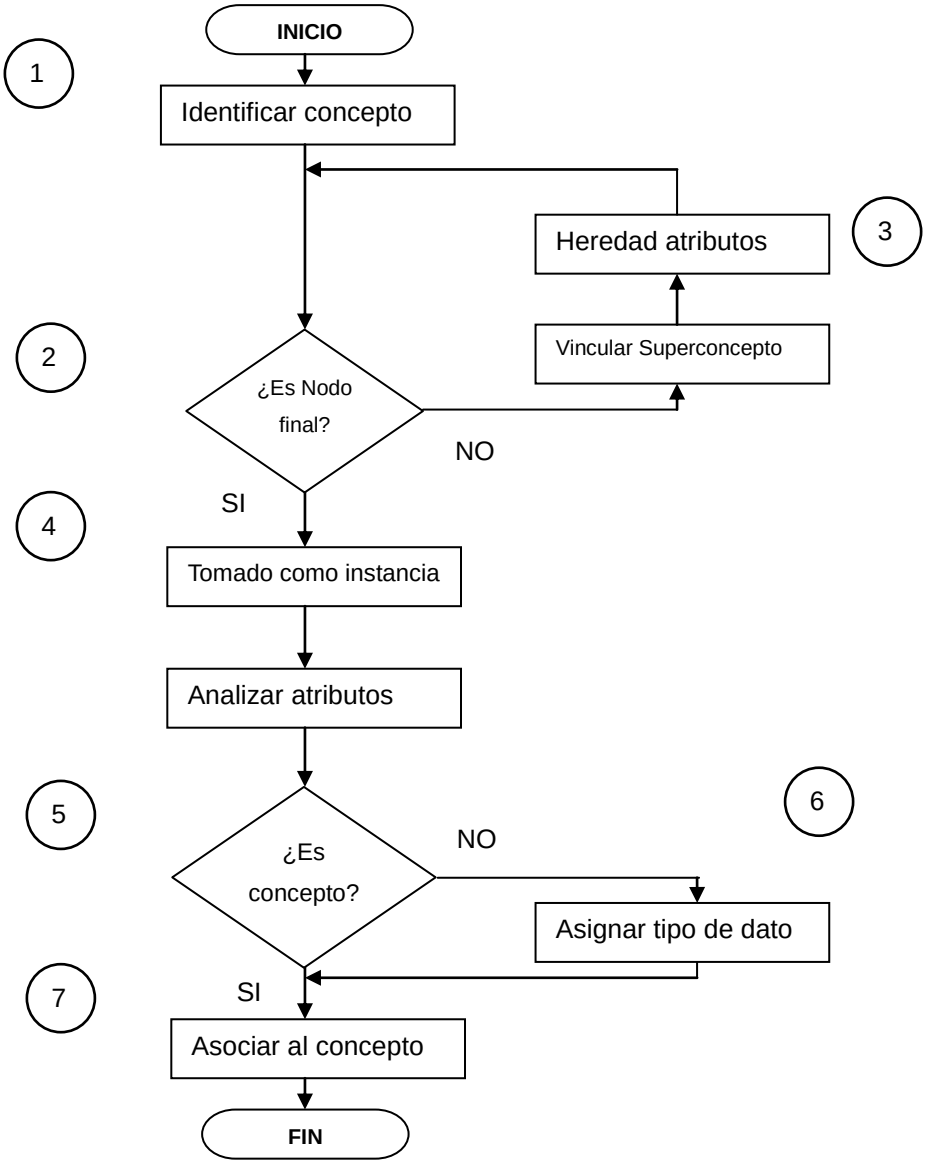


Figura 41. Diagrama de flujo para creación de ontología

Luego de realizado el análisis de los diferentes conceptos (y varios intentos por obtener la solución más adecuada) que se utilizaron en el caso de estudio se procede a implementarlo dentro de la plataforma WSMT y sus herramientas de edición textual y gráfica para la generar el código WSML, en el proceso de implementación de ontologías se han encontrado discordancias al momento de utilizar Dublin Core como la definición de términos y clases, por lo cual, se tomó la decisión de traducir las Clases Dublin Core presentadas en lenguaje RDF a conceptos en WSML.

Dublin Core implementa una ontología complementaria para la definición de términos y propiedades, las mismas que se trataron como atributos en la herramienta WSMT,

posteriormente, se centralizó la implementación de dicha ontología en los conceptos relacionados a Recursos de aprendizaje, bibliografía y Agentes, personas y organizaciones.

Finalmente, se complementó estos conceptos con la adhesión de atributos tomados de las propiedades Dublin Core y complementados con las propiedades de la ontología BIBO, el resultado gráfico de la implementación de la mencionada ontología se muestra en la figura 42.

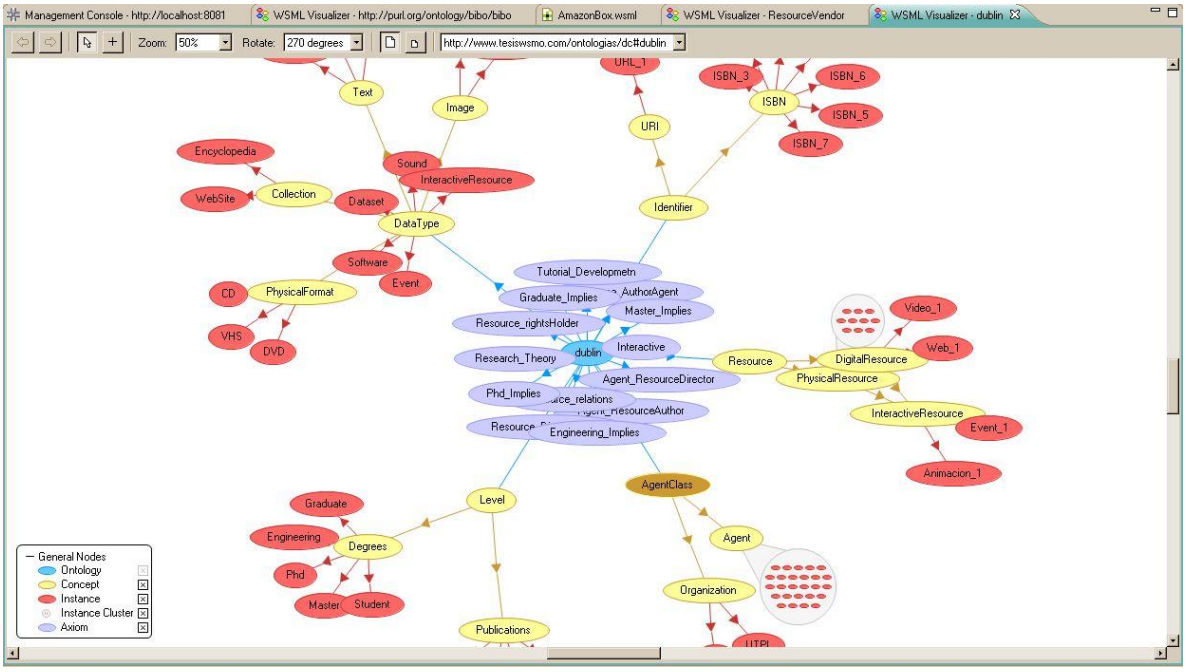


Figura 42. Vista gráfica de la ontología Dublin

El código WSML de esta ontología se encuentra mencionado completamente en el Anexo 1.

5.4.3. CASO DE ESTUDIO PARA LA DESCRIPCIÓN SEMÁNTICA DE SERVICIOS WEB

De forma similar a la sección anterior, se utilizará un diagrama de flujo para definir el proceso de descripción Semántica de los Servicios Web tomados como base para el cumplimiento del objetivo de la Tesis, estos Servicios se describen en el Anexo 2; el procedimiento realizado para la descripción Semántica de los Servicios Web se representa en forma gráfica en el figura 43 y el proceso es el siguiente:

1. **Selecciona SW:** El proceso de selección de los Servicios Web que han sido utilizados en este caso de estudio son Servicios relacionados a los recursos educativos y se han seleccionado de varios repositorios públicos mencionados en esta sección.

2. **Disponibilidad:** Una vez seleccionados los servicios web que serán tomados como base se verifica la disponibilidad y funcionamiento del Servicios, pues se requerirá obtener la información de uso y requerimientos WSDL.

3. **NO disponible:** En caso de no encontrarse disponible el Servicio Web no será tomado en cuenta para evitar conflictos al momento de enlazarse.

4. **Analizar WSDL:** Una vez determinada la existencia del Servicio Web se procederá a analizar el documento descriptivo WSDL para verificar los diferentes requerimientos que serán utilizado en el documento semántico WSML.

5. **Extraer condiciones:** Basándose en el documento WSDL de los diferentes Servicios Web se procede a extraer las precondiciones y postcondiciones para expresar semánticamente el Servicio Web mediante el lenguaje WSML del framework WSMO.

6. **Seleccionar WSML:** Para realizar la descripción semántica del Servicio Web es necesario determinar la variante WSML a ser utilizada, la que deberá ser la misma con la cual se ha realizado la definición de conceptos en la ontología central del caso de estudio.

7. **Definición del SW:** Finalmente se procede a la definición del Servicio Web en el lenguaje WSML, definiendo las precondiciones, postcondiciones y variables compartidas e interfaces.

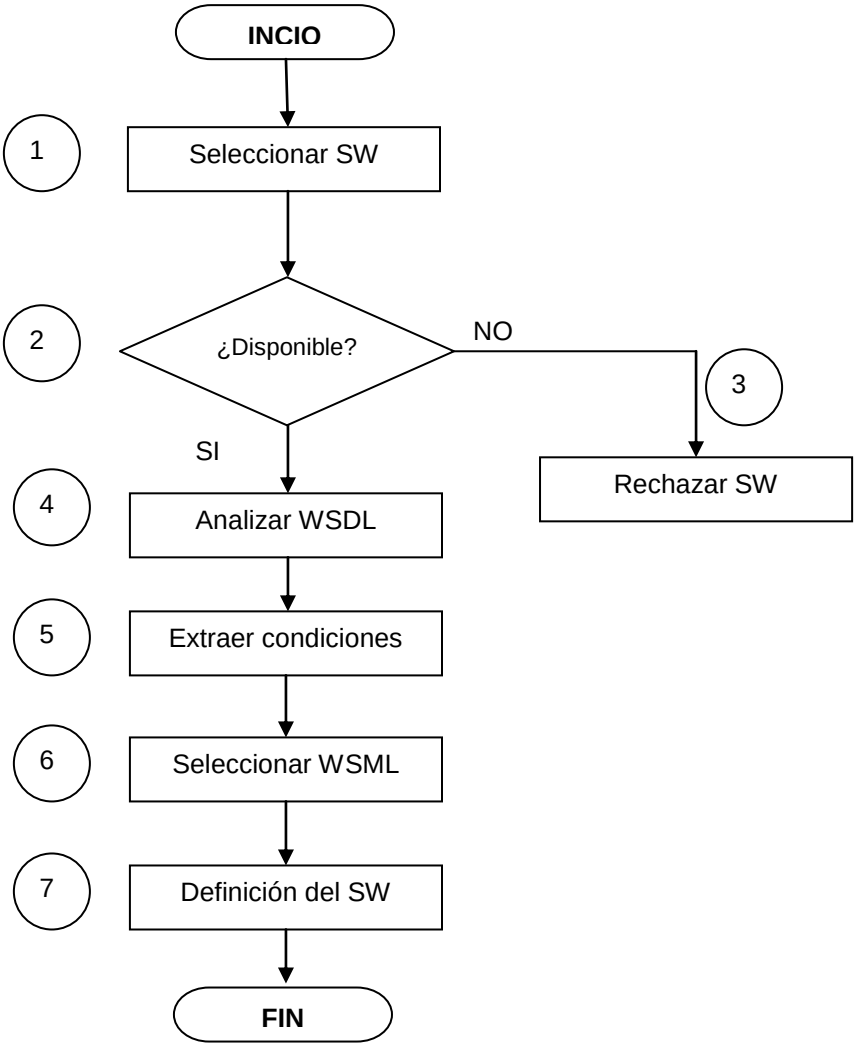


Figura 43. Diagrama de flujo descripción de WS[69]

Como se ha indicado en el punto anterior el tema central para los casos de estudio se ha orientado principalmente a los recursos educativos, de esta manera se realizó la recolección de Servicios Web públicos relacionados a este tema en particular.

Una vez obtenidos dichos servicios y los documentos con la descripción sintáctica correspondiente se procedieron a implementarlos en el lenguaje WSML, cabe destacar, que ciertas características encontradas en los documentos WSDL de los Servicios Web no han sido incorporadas en la nueva descripción WSML pues carecían de importancia semántica.

Dentro de los Servicios Web implementados tenemos:

- AmazonBox
- BookStoreService

[69] Creación del diagrama y descripción basado en la experiencia del manejo de la herramienta WSMT y los conceptos teóricos aplicados

- Events
- LookUpBookData
- StoreService
- VideoWebService
- YouTubeDownload_Service

Todos estos servicios han sido tomados de repositorios UDDI públicos como seekda.com, xmethods.com, webservicex.net; los documentos WSDL y las direcciones de entrada de los Servicios Web se encuentran detallados en el Anexo 2.

La incorporación de dichos Servicios Web en la herramienta WSMT se muestra en la figura 44.

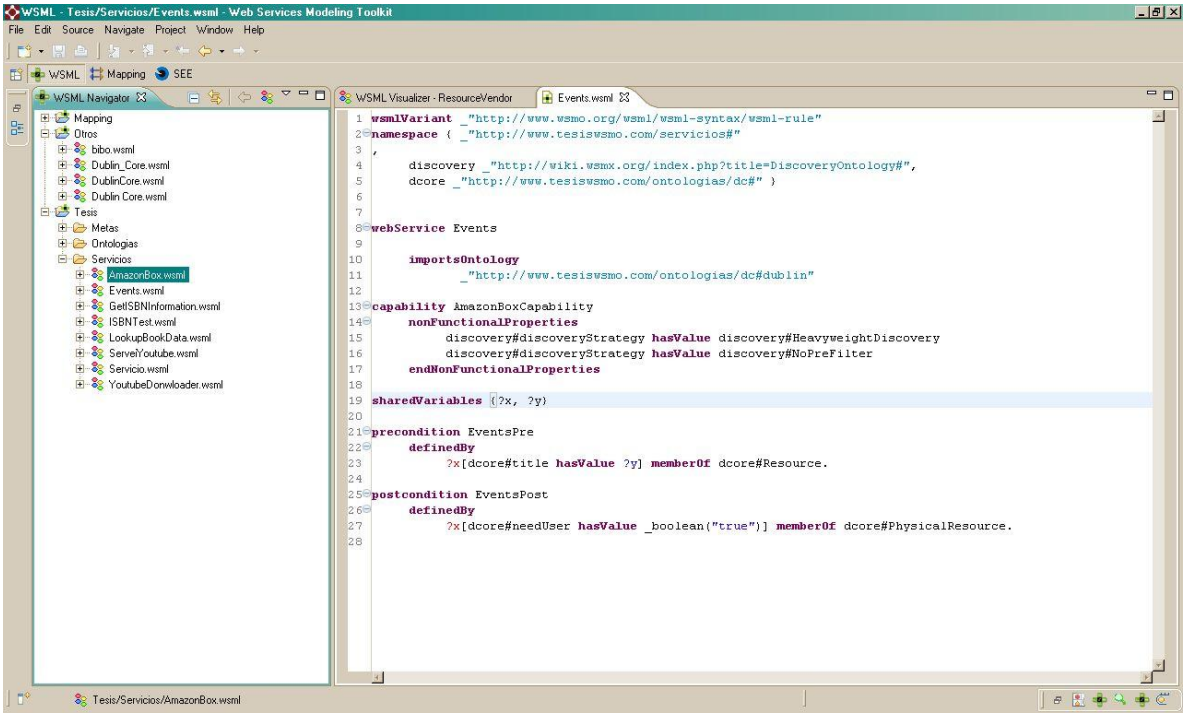


Figura 44. Servicios Web implementados

5.4.4. CASO DE ESTUDIO PARA LA CREACIÓN DE METAS

La implementación de las Metas se ha desarrollado con el objetivo de mostrar el funcionamiento de la herramienta WSMT y sus capacidades de descubrimiento semántico, las precondiciones y postcondicones descritas en cada una de las Metas utilizan como referencia la ontología “Dublin” creada a partir de las características de la ontología Dublin Core y de la ontología BIBO para servir de base en los casos de estudio de esta Tesis.

Estas Metas proponen el descubrimiento de varios Servicios Web almacenados con diferentes grados de coincidencia, al tratar la ontología y los Servicios Web del mismo tema en general - Recursos de aprendizaje - se puede mostrar resultados más eficientes con el propósito de comprobar las teorías investigadas. La incorporación de las Metas en la herramienta WSMT se muestra en la siguiente figura 45.

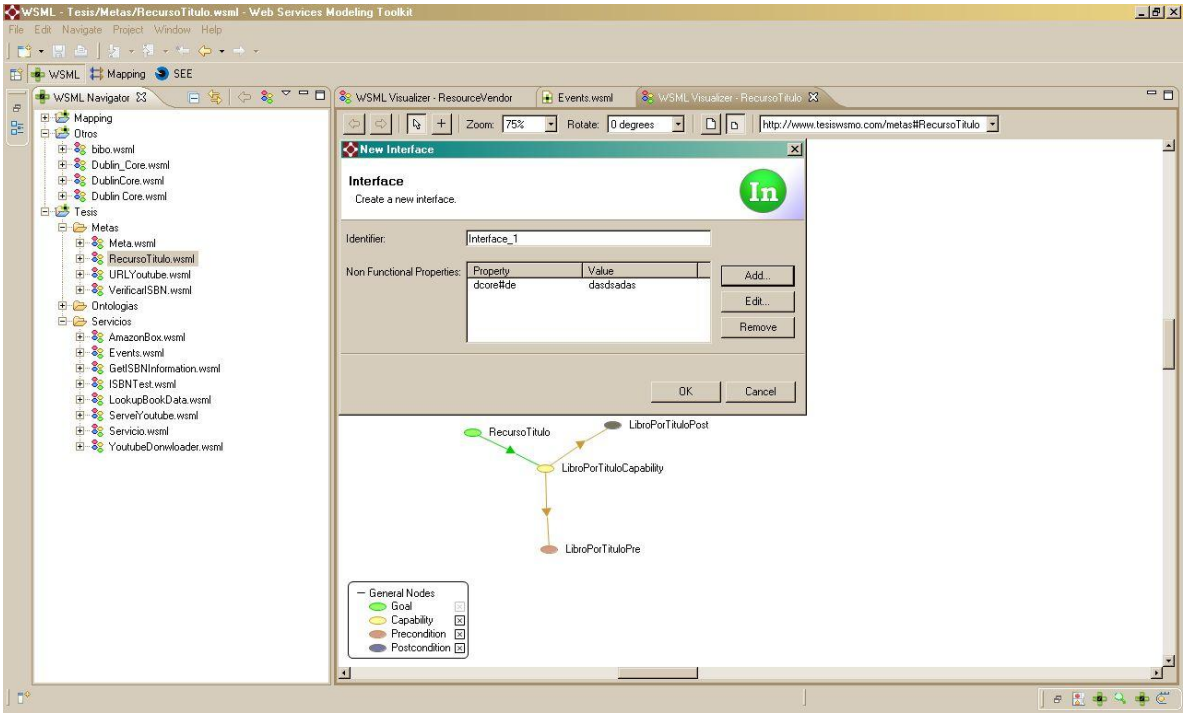


Figura 45. Metas implementadas

5.4.5. DESCUBRIMIENTO DE SERVICIOS WEB

Una vez implementada la Ontología que contiene los conceptos, propiedades, atributos y axiomas referentes a los Recursos de aprendizaje del caso de estudio de esta tesis (basado en Dublin Core y BIBO), se procedió a implementar los Servicios Web encontrados en repositorios UDDI y a la implementación de las Metas necesarias para la comprobación del descubrimiento semántico de Servicios.

Este descubrimiento semántico se realiza con la utilización de la herramienta *WSML Discovery* contenida dentro del conjunto de herramientas WSMT, esta herramienta trabaja internamente con el razonador WSML2Reasnoer.

El proceso de descubrimiento de los Servicios Web Semánticos registradas en la herramienta WSMT basándose en una Meta, ambos conceptos descritos semánticamente mediante el lenguaje WSML, se describe a continuación:

1. **Seleccionar la Meta:** La Vista WMSL Discovery nos muestra todas las Metas que tenemos almacenados en la herramienta WSMT, estas metas están asociadas a las diferentes ontologías mediante la palabra reservada ***importsOntology.***

2. **Variante WSML:** De similar forma se asocia la Meta a la variante WSML utilizada para realizar el descubrimiento mediante la definición de ***wsmlVariant***. Esta variante WSML será la encargada de realizar el razonamiento y calificación sobre los Servicios Web Semánticos que trabajen con la misma variante WSML.

3. **¿Similitud WSML?:** En esta condición la herramienta WSML Discovery mediante su razonador WSML2Reasoner identificará los Servicios Web que trabajen sobre la misma variante WSML y seleccionando el razonador interno que permitirá concluir con el descubrimiento.

4. **NO coincide:** Todos aquellos Servicios Web Semánticos que trabajen sobre una diferente variante WSML (WSML-Core, -DL, -Rule, -Flight ó WML-Full) que la utilizada en la Meta no serán tomados en consideración para el resultado del descubrimiento.

5. **Analizar descripciones:** Una vez determinados los Servicios Web Semánticos que coincidan con la variante WSML utilizada por la Meta se procede al análisis de las descripciones generales del Servicios, incluida en la descripción semántica WSML a través de la palabra reservada ***nonFunctionalProperties***, dentro de este bloque de código se describen las características generales del Servicio Web como son, Proveedor, ingresos, salidas y descripciones generales. Esta es la primera fase de calificación del Servicio Web Semántico.

6. **Análisis de las precondiciones:** Una vez analizadas las descripciones NFP del Servicio Web Semántico, se procede a analizar las precondiciones existentes en el Servicio y compararlas con las precondiciones requeridas en la Meta, cumpliéndose de esta manera la segunda fase de razonamiento para el resultado del descubrimiento.

7. **Análisis de las postcondiciones:** De manera similar al paso anterior se realiza un análisis comparativo de las postcondiciones del Servicio Web Semántico con las postcondiciones requeridas en la Meta seleccionada.

8. **Obtención de resultados:** Luego de terminados los procesos anteriores, la herramienta WSML Discovery desplegará una cantidad de resultados obtenidos del análisis de los Servicios Web Semánticos coincidentes tanto en la ontología utilizada como en la

variante WSML para resolver el razonamiento. Cabe destacar que el resultado despliega una columna con los tipos de coincidencia asociados a cada Servicio, estos tipos se tratarán posteriormente.[70]

La herramienta WSML Discovery mostrada en la figura 47 expone varias características como: la Meta, el resultado de los Servicios descubiertos, la URL relacionada a dichos Servicios (punto de entrada) y el grado de coincidencia obtenida para cada Servicio, estos grados de coincidencia varían dependiendo de la variante WSML utilizada.

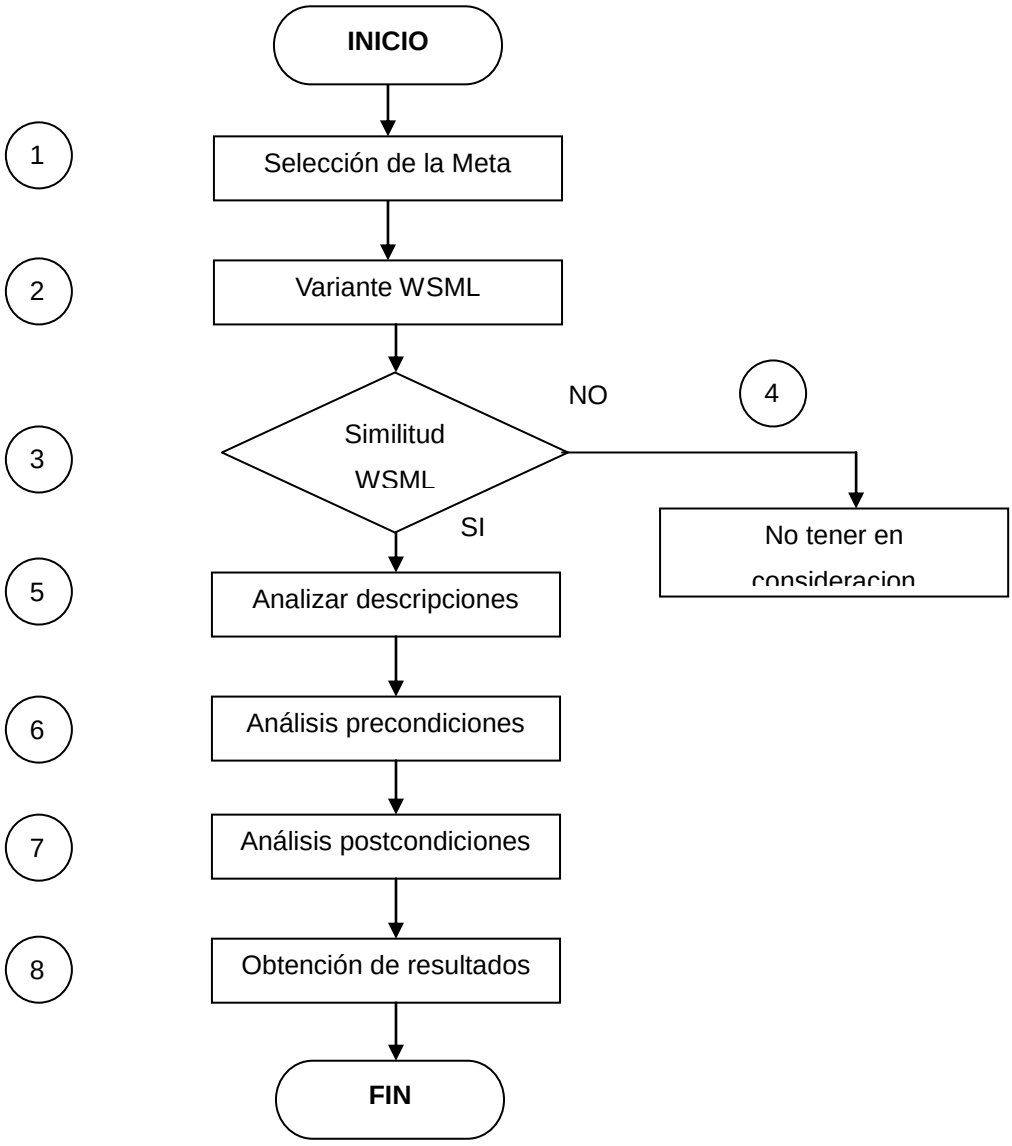


Figura 46. Diagrama de flujo descripción de WS

[70] Este proceso de descubrimiento de los Servicios Web Semánticos tiene su base teórica en **[49] [50]** y **[68],** a demás de los resultados obtenidos con varias implementaciones realizadas durante la fase de aprendizaje de la herramienta WSMT 1.4.1. y WSMT 2.0. y sus complementos WSML2Reasoner, WSML Discovery

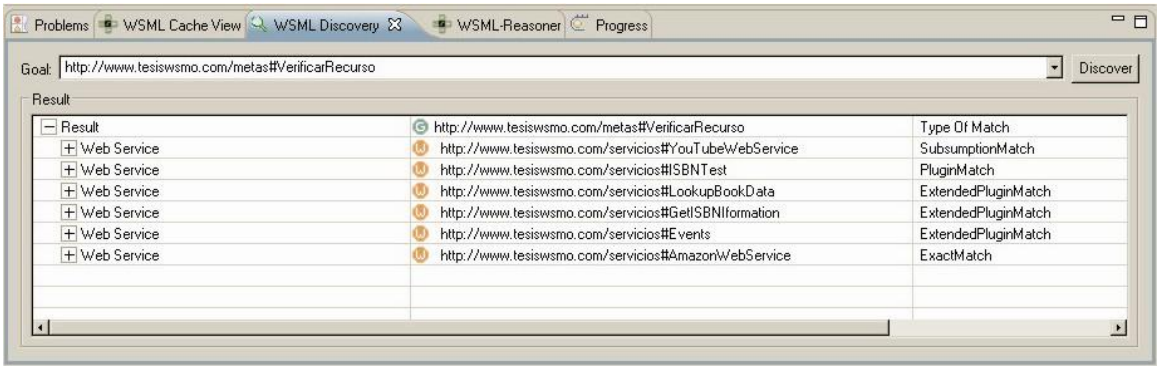


Figura 47. Resultado del descubrimiento

Los grados de coincidencia mostrados por la herramienta otorgan un alto nivel de confianza para que el usuario o agente software tome la decisión de utilizar un determinado Servicio, estos grados de coincidencia son: **[18][19]**

- **No Match:** Son aquellos Servicios que no tiene relación con la Meta, por tanto son excluidos de la lista de resultados.
- **SubsumptionMatch:** El resultado es un subconjunto del concepto esperado.
- **PluginMatch:** Los conceptos coinciden, pero no ciertos atributos.
- **ExtendedPluginMatch:** Existe similitud de conceptos y ciertos atributos esperados.
- **ExactMatch:** Coincide en los conceptos y atributos solicitados.

Los grados de coincidencia obtenidos mediante el descubrimiento semántico de la herramienta WSMT juegan un papel importante en la selección del Servicio y la posterior invocación del mismo (Véase la sección 3.2), pero no prevén una absoluta certeza de la existencia del Servicio, por lo cual es imprescindible tomar en cuenta otros aspectos como son:

- Confiabilidad del Servicio
- Similitud de las Ontologías utilizadas para cada Servicio Web
- Proveedor del Servicio
- Contrato general del Servicio

Finalmente, una correcta selección e invocación del Servicio es aún un objetivo en estudio en la teoría relacionada al framework WSMO, pues requieren la intervención del usuario humano para garantizar la completa satisfacción de sus necesidades, las proyecciones establecidas para el cumplimiento de este objetivo incluyen la Orquestación y Coreografía de Servicios, así como, la calificación de los Servicios obtenida a partir de la utilización por otros.

FASE 3

CAPITULO 6:

COMPARACIÓN DE LA METODOLOGÍA WSMO CON METODOLOGÍAS TRADICIONALES

6.1. METODOLOGÍAS TRADICIONALES

Como se ha tratado en capítulos anteriores, el manejo de los Servicios Web requiere de la publicación de su información general, así como la descripción sintáctica de métodos e interfaces dentro de un repositorio UDDI local o público mediante el lenguaje WSDL. Los repositorios UDDI trabajan en un entorno web que permite la comunicación mediante el uso de mensajes SOAP, así como otras alternativas de comunicación con el repositorio.

La utilización del lenguaje WSDL nos permite describir los métodos, interfaces, puertos de comunicación y parámetros de entrada/salida del Servicio Web, definiendo de esta manera las capacidades y limitaciones que posee el servicio, así como la forma de interactuar con dicho servicio. [71]

6.1.1. REPOSITORIOS UDDI

6.1.1.1. Repositorios UDDI locales

Un repositorio UDDI puede ser manejado localmente para la publicación de los Servicios Web propios de una misma empresa, este catálogo para servicios trabaja vía web mediante el protocolo HTTP y conformado por mensajes SOAP.

El repositorio UDDI local que se expone en la presente Tesis es jUDDI, desarrollado en Java para trabajar con el Servidor de aplicaciones Apache Tomcat, jUDDI puede ser configurado para trabajar con bases de datos relacionales como MySQL, Oracle, SQLServer, etc.

El proceso de descarga e instalación del repositorio jUDDI se expone detalladamente en el Anexo 5 del presente documento.

Para el trabajo con el repositorio jUDDI es necesario levantar los Servicios de MySQL y Apache para luego ubicarnos en la dirección http://localhost:8080/juddi en nuestro navegador; jUDDI

71 Este análisis se basa en el manejo tradicional de Servicios Web SOAP, su forma de descripción, publicación y búsqueda (descubrimiento sintáctico) de los servicios.

permite la interacción web con sus diferentes elementos como son la publicación de Negocios, Servicios, tModels a través de los elementos de consola instalados con jUDDI en nuestro Servidor Apache.

Cada uno de estos métodos es un mensaje SOAP que permite la interacción con jUDDI y devuelve otro mensaje SOAP de confirmación por cada tarea realizada, la consola web jUDDI se muestra en la figura 48. **[64]**

Figura 48. Consola web repositorio jUDDI

El trabajo con jUDDI a través de la consola de comandos es una tarea engorrosa y complicada de llevar a cabo, pues requiere del `Token` de autenticación correspondiente al Negocio que publica los servicios, `Bussiness_Key` para la publicación de Servicios y `tModelKey` para la creación de descripciones y enlazar el documento WSDL. Como alternativa al trabajo de consola se ha utilizado la herramienta UDDI Browser que maneja la comunicación SOAP con el repositorio jUDDI de modo transparente al usuario.

UDDI Browser es una herramienta desarrollada en Java y de libre distribución que se muestra en la figura 49, esta herramienta ofrece ventajas como:

- Entorno visual de trabajo
- Búsqueda básica y avanzada para Negocios, Servicios y tModels

- Interacción con repositorios UDDI locales y remotos
- Métodos de agregación para Negocios, Servicios y tModels
- Clasificación taxonómica de Servicios

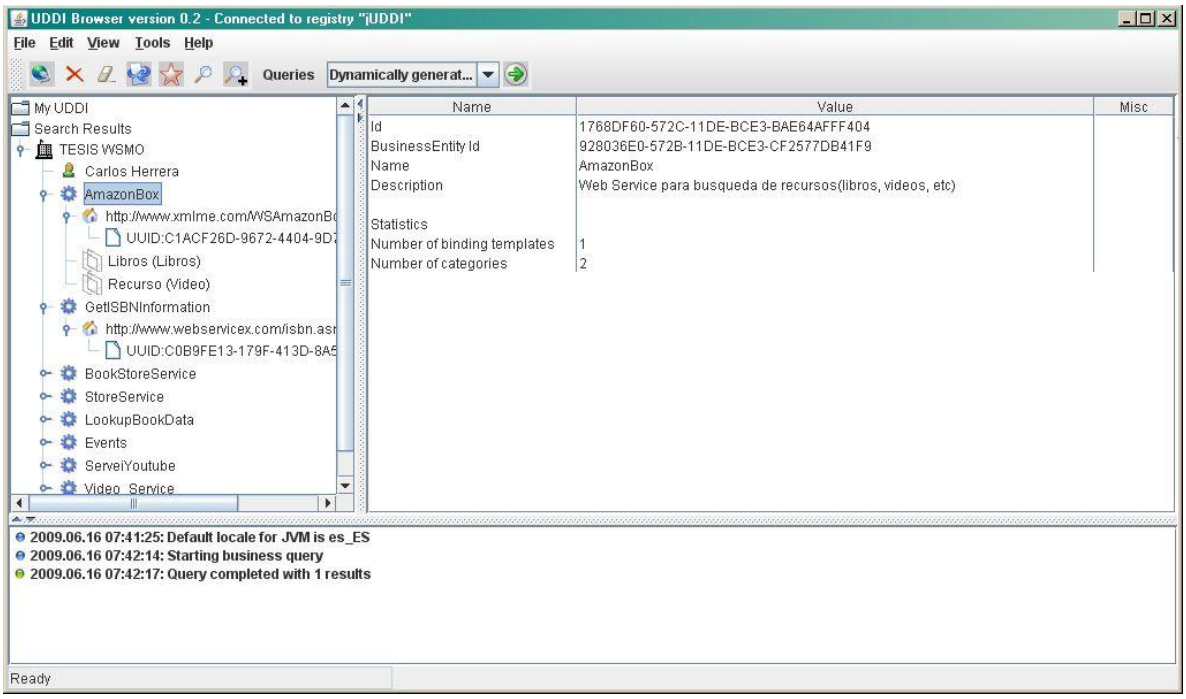


Figura 49. Entorno de trabajo UDDI Browser

Para objetos de comprobación de la teoría investigada en la presente Tesis se utilizó la herramienta UDDI Browser para la publicación de cada uno de los Servicios Web descritos en la sección 5.3.3. **[65]**

6.1.1.2. Repositorios UDDI públicos

Los repositorios UDDI públicos trabajan de manera similar a lo señalado en el punto anterior, son directorios generales de Servicios que permiten realizar búsquedas sencillas y avanzadas; principalmente su modo de trabajo es mediante la búsqueda de coincidencia de palabras dentro de las descripciones y métodos publicados por la Empresa o Servicios.

Existe gran cantidad de Servicios publicados en dichos repositorios, sin embargo, el lograr una búsqueda exitosa en un determinado repositorio UDDI no es garantía de que el Servicio encontrado siga en funcionamiento, no obstante, algunos repositorios UDDI públicos ofrecen la capacidad de probar el servicio a través de su consola web.

Para la búsqueda y descripción de los Servicios Web descritos en la sección 5.3.3. se ha utilizado el repositorio SEEKDA[72] que ofrece características de búsqueda avanzada y una consola de pruebas; dicha interacción muchas veces no pudo completarse pues las descripciones de los parámetros de entrada no son claros y no devuelven resultado alguno. La siguiente figura 50 muestra el entorno de trabajo del repositorio SEEKDA.

Figura 50. Repositorio UDDI público SEEKDA

6.1.2. DESCRIPCIÓN SINTÁCTICA DE SERVICIOS WEB[73]

Para determinar el funcionamiento de los métodos que constituyen un Servicio Web es necesario realizar la descripción sintáctica de dicho Servicio, donde se exponen los parámetros de entrada, tipos de dato requeridos, puertos de comunicación, etc.

Durante el desarrollo de la presente Tesis se ha expuesto las diferentes metodologías de desarrollo de Servicios Web, y, cada una de ellas ha generado su propio lenguaje para la descripción de su funcionamiento.

Cabe resaltar que el documento de descripción no es suficiente para conocer a profundidad el manejo e interacción del Servicio, es por ello que se requiere obtener una API del Servicio donde se detalle el funcionamiento de cada Método; generalmente estas API's son distribuidas luego de la contratación del servicio.

72 Repositorio público para Servicios Web SEEKDA disponible en [http://seekda.com]

73 Tomado de las Fuentes **[8]** y **[66]** donde se podrá ampliar los procesos realizados

6.1.2.1. Descripción WADL

El lenguaje de Descripción de Aplicaciones Web (WADL) se desarrolló para permitir la descripción de los Servicios Web basados en la tecnología REST, es un metalenguaje basado en XML que describe las funcionalidades del Servicios mediante etiquetas como `type`, `message`, `operation`, `port`.

Con el surgimiento de este lenguaje el W3C trabajó en la modificación del lenguaje WSDL 1.0 y 1.1 que originalmente solo permitían la descripción de Servicios Web basados en SOAP, con el lanzamiento de WSDL 2.0 se pretende acortar la brecha entre la descripción de los Servicios Web independientemente de la metodología que utilicen.

6.1.2.2. Descripción WSDL

El lenguaje WSDL 2.0 expuesto como recomendación W3C en Junio de 2007 para la descripción de Servicios Web sin diferenciar la metodología de desarrollo que se utilice, ha mejorado la generación de publicaciones de servicios, pues para los desarrolladores ya no se requiere el aprendizaje de otros lenguajes de publicación.

WSDL está basado en XML, como lenguaje de marcas permite realizar la descripción general de los métodos e interfaces utilizadas en un Servicios Web SOAP o REST, mediante esta descripción se puede determinar la similitud entre el Servicio encontrado y los parámetros que podemos ofrecer como entrada y la información que requerimos como salida.

Entornos de desarrollo de Software como NetBeans o Visual Studio .Net generan el documento WSDL automáticamente conforme se realiza la programación del Servicio, es por ello que los documentos WSDL encontrados en los repositorios UDDI no poseen mayor riqueza descriptiva lo que dificulta el entendimiento de los parámetros de entrada y métodos que el Servicio Web ofrece.

Una visión general de un documento WSDL examinado desde la plataforma Eclipse se visualiza en la figura 51 mostrada a continuación.

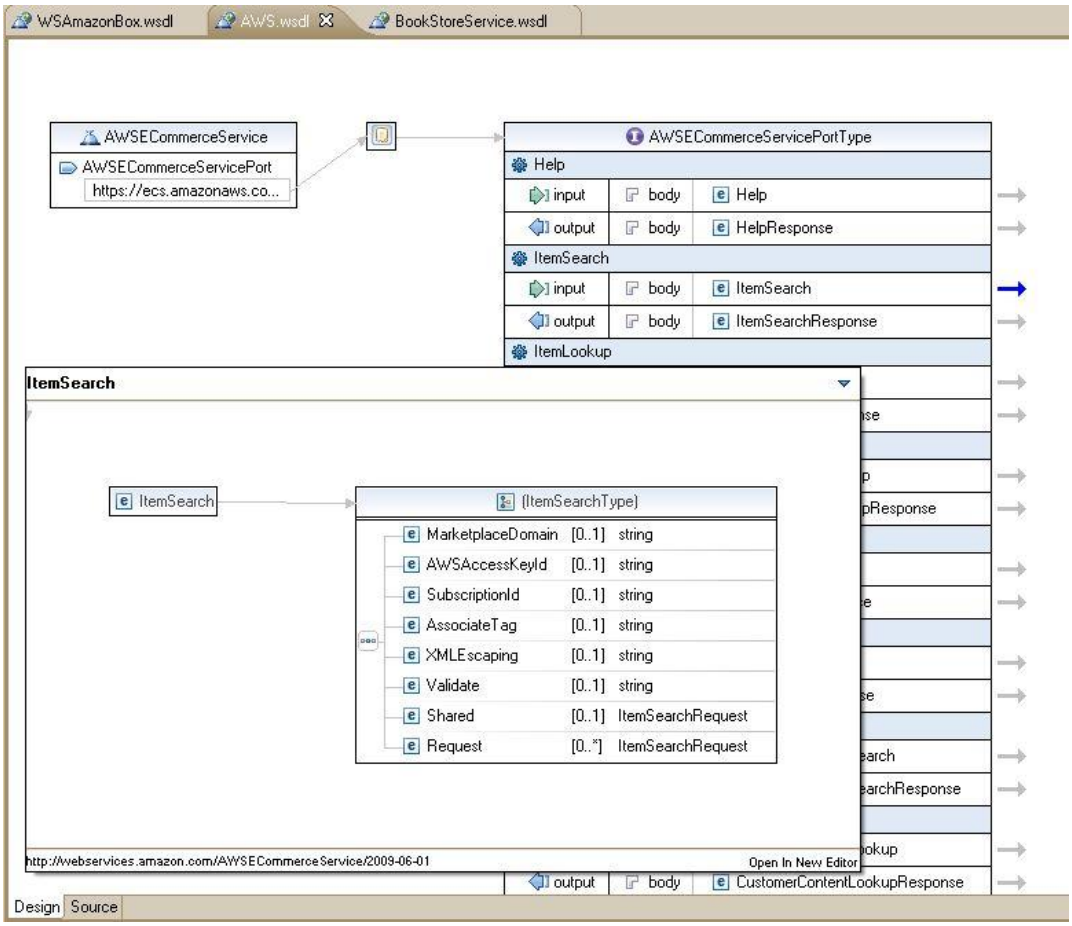


Figura 51. Documento WSDL[74]

6.1.3. DESCUBRIMIENTO DE SERVICIOS WEB

Como objetivo general del presente proyecto se ha planteado la implementación de un caso de estudio que permita realizar el descubrimiento de Servicios Web, en la primera parte se describe el proceso de descubrimiento sintáctico de un Servicio Web; para ello, se tomarán criterios descritos anteriormente.

Para el descubrimiento sintáctico (búsqueda) de un Servicio Web se requiere la publicación del Servicio en un repositorio UDDI público, estos repositorios permiten agregar una variada cantidad de información referente a la Empresa administradora del Servicio y características descriptivas del Servicio en sí.

Se utilizará nuestro repositorio jUDDI local para realizar la publicación de un mínimo conjunto de Servicios que permitan demostrar un descubrimiento sintáctico, cabe resaltar que, en un

74 **Documento WSDL:** Visión general de los documentos WSDL autogenerados en la plataforma Eclipse

repositorio público como SEEKDA existe una enorme cantidad de Servicios Web, lo cual influye negativamente en la obtención de resultados.

La búsqueda sintáctica de un Servicio Web inicia por la declaración de necesidades del usuario, estas necesidades serán incluidas en los parámetros de búsqueda incluyendo categorización taxonómica, palabras clave y nombre del servicio como se muestra en la figura 52.

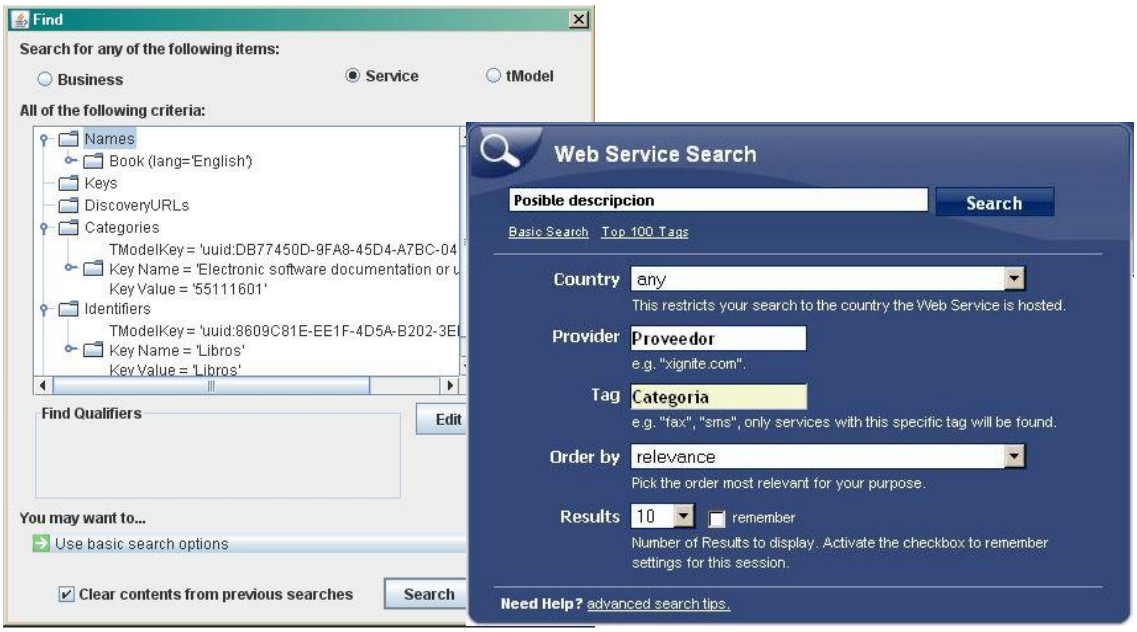


Figura 52. Opciones de búsqueda avanzada UDDI Browser / SEEKDA

Una vez obtenidos los resultados de la búsqueda se procede a analizar cada uno de los servicios, basándose primeramente en las descripciones adicionales para comprender su funcionamiento técnico; seguidamente, se procede al análisis de los documentos WSDL direccionados a través del repositorio.

Los criterios de selección del Servicio Web más adecuado a las necesidades expuestas son directamente responsabilidad del usuario final, pues, en la mayoría de los casos se requiere un análisis de los documentos WSDL para determinar las coincidencias con la petición, a continuación la figura 53 muestra un listado de los Servicios resultantes a nuestra petición.

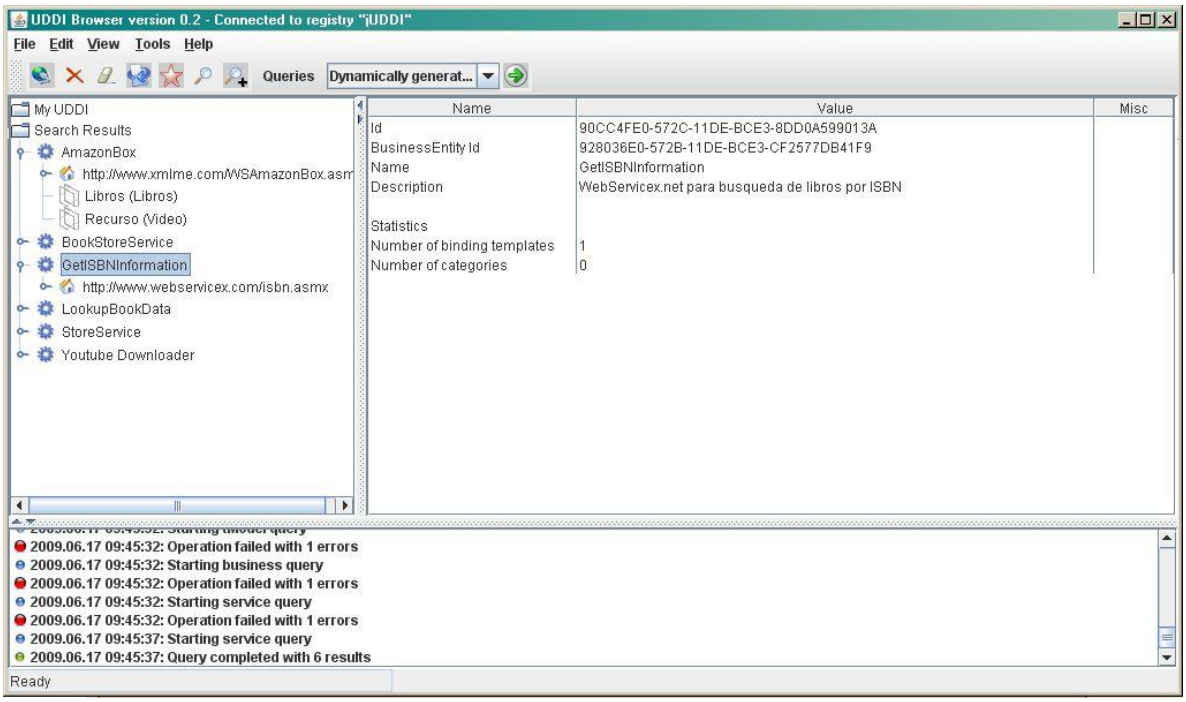


Figura 53. Listado de servicios descubiertos en UDDI Browser

Es posible el mejoramiento de los resultados obtenidos mediante diferentes acciones:

- Ofrecer una descripción textual detallada de los Servicios Web, sus métodos e interfaces, pues el documento WSDL generado automáticamente por los entornos de desarrollo no incluye estas descripciones.
- Denominar a los Servicios con nombres relacionados al trabajo que realiza el Servicio
- Utilizar categorías taxonómicas estándar para clasificar los Servicios previa su publicación (criterios semánticos)

Adicionalmente a las acciones que se puede realizar por parte del desarrollador del Servicio Web para mejorar las búsquedas sintácticas, es posible incrementar las capacidades de los repositorios UDDI mediante la utilización de software de análisis de los documentos WSDL para incluir estas características dentro de los parámetros generales de descubrimiento de Servicios. [75]

75 El proceso de descubrimiento es una búsqueda manual basada en similitudes sintácticas entre los servicios expuestos y los parámetros ingresados en los repositorios **[64]** y **[65]**

6.2. METODOLOGÍA WSMO

Mediante la utilización de la metodología WSMO se realizará el descubrimiento semántico de los Servicios Web publicados, realizando la creación de varias Metas que nos permitan comparar los resultados obtenidos sintácticamente y demostrar el mejoramiento obtenido al utilizar descripciones semánticas.

Los puntos esenciales para el descubrimiento de un Servicio Web que hemos tratado en secciones anteriores son:

- Crear una ontología que contenga los conceptos relacionados a los Recursos de aprendizaje.
- Describir semánticamente los Servicios Web utilizando la ontología anterior.
- Describir semánticamente las Metas que requerimos encontrar basada en la ontología.
- Utilizar la herramienta WSMT para el desarrollo de todos estos conceptos.

En los siguientes puntos se expone en mayor detalle los puntos anteriores relacionados con la creación de nuestro caso de estudio.

6.2.1. ENTORNO DE EJECUCIÓN WSMX

La metodología WSMO utiliza principalmente la herramienta WSMT para realizar la creación y modelado de todos los conceptos (Ontologías, Servicios, Metas y Mediadores); estos conceptos permanecen incluidos en el espacio de trabajo de la herramienta y nos facilita el razonamiento, interacción y descubrimiento mediante las herramientas adicionales como WSML Discovery, WSML Reasoner, etc.

Adicionalmente en el capitulo 4.3.2. se ha descrito el ambiente de ejecución WSMX, el cual puede ser utilizado a manera de repositorio UDDI para las descripciones de los Servicios Web Semánticos, estas descripciones se almacenan en el Servidor Apache Tomcat que sirve de base para WSMX.

WSMX posee una consola de comandos que permiten la incorporación de nuevos conceptos WSMO a través de código WSML o direcciones URL, de esta manera se puede realizar una consulta vía web. La descarga e instalación del ambiente de ejecución WSMX se detalla en el Anexo 6 y su consola de comandos se muestra en la figura 54. **[67]**

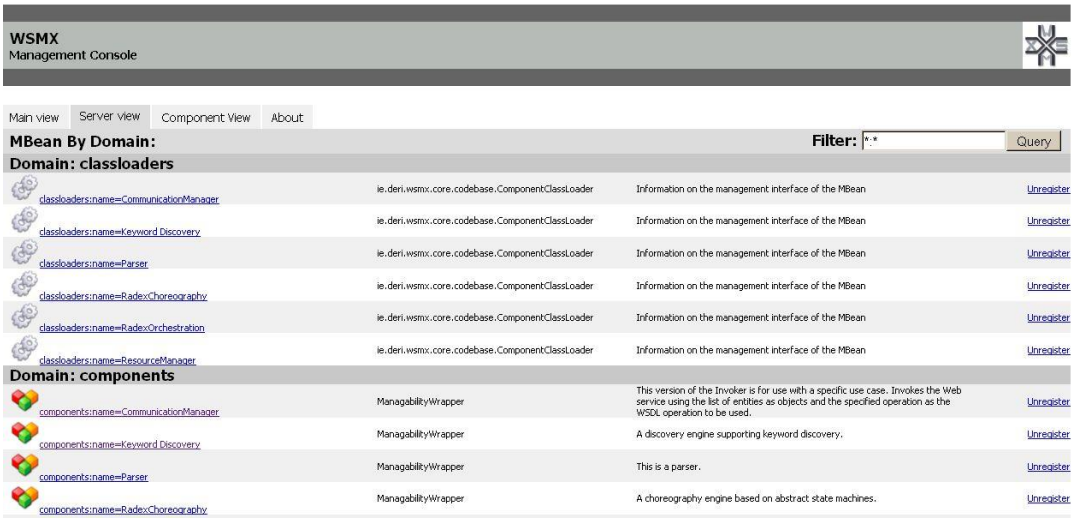


Figura 54. Consola WSMX[76]

6.2.2. DESCRIPCIÓN SEMÁNTICA WSML

Para el descubrimiento semántico de los Servicios Web es necesario crear su descripción semántica en el lenguaje WSML, tanto de Ontologías, Servicios y Metas que se relacionen con un tema específico, en nuestro caso, los Recursos de Aprendizaje.

Como se ha descrito en la sección 5.2. las Metas y Servicios se relacionan a través de la especificación de precondiciones y postcondiciones descritas dentro de una interface, para realizar el descubrimiento semántico de los servicios debemos elaborar estas descripciones.

La figura 55 muestra un ejemplo sencillo del código WSML para declaración y descripción de una Meta o un Servicio general y los elementos que la componen.

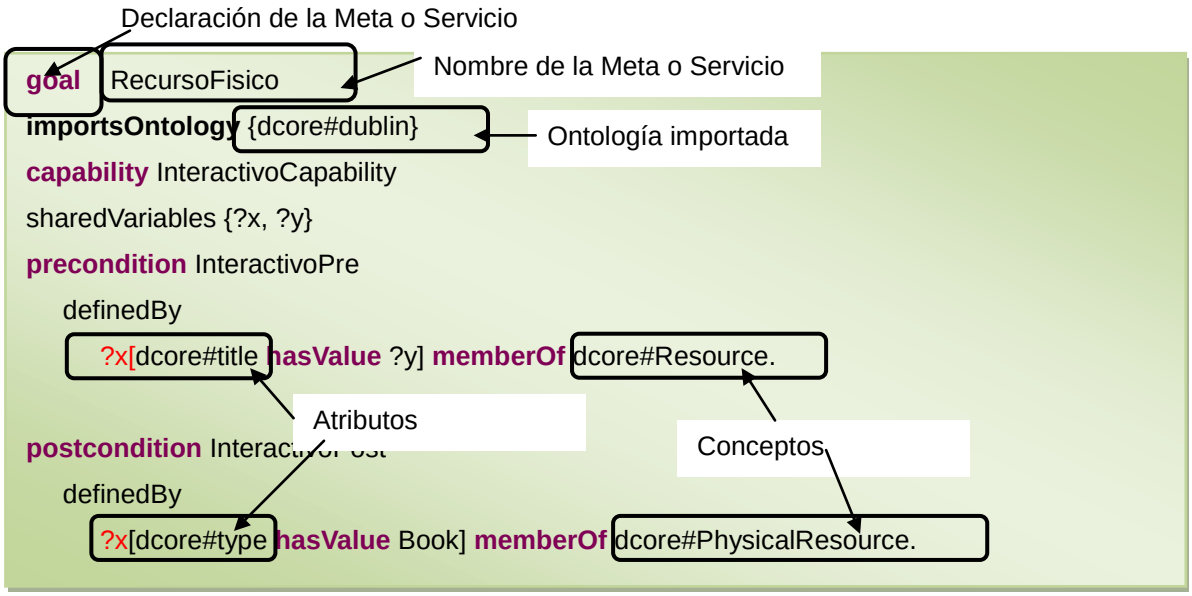


Figura 55. Código WSML de una meta[77]

76 **Consola WSMX:** Entorno de aplicación web obtenido de la instalación del paquete WSMX. Anexo 6

6.2.3. DESCUBRIMIENTO SEMÁNTICO DE SERVICIOS WEB

Una vez realizadas las descripciones de los diferentes conceptos (Ontología, Servicios y Metas) se procede a invocar la Meta para la cual requerimos ejecutar un descubrimiento semántico, la invocación de la Meta se puede realizar en la herramienta de modelado WSMT o en el entorno de ejecución WSMX.

El procedimiento que involucra el descubrimiento semántico hace la comparación de las descripciones entre Metas y Servicios (Véase la sección 5.4.5.), tomando en cuenta los elementos antes descritos, es así que los Servicios resultantes deben coincidir de manera semántica con las precondiciones y postcondiciones declaradas en la Meta.

La comparación entre precondiciones y postcondiciones de una Meta y los Servicios se realizan tomando en cuenta los siguientes criterios:

- Propiedades no funcionales (*No Functional Properties)*
- Conceptos ontológicos que conforman las postcondiciones.
- Atributos esperados/devueltos entre las postcondiciones.
- Conceptos ontológicos que conforman las precondiciones.
- Atributos entregados/recibidos en las precondiciones.

De acuerdo al grado de coincidencia de estos criterios la herramienta WSMT nos devolverá un listado de Servicios, en el cual consta una columna con la información del tipo de coincidencia encontrada. La definición de los tipos de coincidencias se detalla en la siguiente tabla, la figura 56 muestra el resultado devuelto en WSMT.

TIPO DE COINCIDENCIA	PRECONDICIONES		POSTCONDICONES	
	Conceptos	Atributos	Conceptos	Atributos
Nula	- - - - -	- - - - -	- - - - -	- - - - -
Subsumption	x x x x	- - - - -	- - - - -	- - - - -
Plugin	x x x x	x x x x	- - - - -	- - - - -
Extended Plugin	x x x x	x x x x	x x x x	- - - - -
Exact	x x x x	x x x x	x x x x	x x x x

CLAVES:

- - - - -	No existe coincidencia alguna
x x x x	Son nociones exactas o relacionadas

77 **Código WSML de una meta:** Se resaltan los puntos esenciales que permitirán el descubrimiento automático de Servicios Web Semánticos

Figura 56. Resultado del descubrimiento[78]

Cabe destacar que los diferentes tipos de coincidencias varían de acuerdo a la variante WSML que se haya seleccionado para trabajar los conceptos WSMO, así como la correcta descripción de dichos conceptos.[79]

6.3. COMPARACIÓN DE RESULTADOS

Una vez realizados los procesos de descubrimiento sintáctico y semántico utilizando las tecnologías tradicionales (SOAP) y la metodología WSMO en los diferentes ambientes de publicación (WSMT, WSMX) podemos realizar una comparación de resultados.

Esta comparación de resultados se realizará enfocando un análisis de los pasos a seguir para obtener el descubrimiento de un Servicio Web requerido, los principales objetivos de la semántica aplicada a los Servicios Web son la reducción en tiempos de descubrimiento de Servicios Web claramente utilizables, y la correcta interpretación de requerimientos del usuario, pudiendo ser este otro Servicio Web

6.3.1. GENERACIÓN DE DOCUMENTOS DESCRIPTIVOS

El tiempo de publicación de los Servicios Web (SOAP) depende directamente de la generación del documento descriptivo WSDL, y por tanto podrá ser reducido utilizando las plataformas de desarrollo como Eclipse o .NET, teniendo una incidencia directa en la interpretación de parámetros de entrada/salida que requiera el usuario; pues, estas plataformas de desarrollo generan documentos WSDL básicos con carencia de descripciones y significados.

---

[78] Adaptada de los resultados obtenidos con las diferentes variantes WSML bajo pruebas con la herramienta WSMT

79 La explicación detallada en este punto se basa en el manejo de la utilización de la herramienta WSMT y su comparación con los conceptos expuesto en **[49] [50] [67] [68]** y **[69]**

Por otra parte, para la publicación de un Servicio Web Semántico según la metodología WSMO, es necesario crear un documento descriptivo WSML que relacione cada uno de los parámetros de entrada y salida del Servicio con los conceptos descritos en la ontología que se vaya a utilizar, generando un documento que define claramente los conceptos utilizados por el Servicio.

6.3.2. PUBLICACIÓN DE SERVICIOS WEB

Para lograr la publicación de los Servicios Web sintácticos es necesario vincular la dirección del documento WSDL al Servicio Web y publicarla mediante una plataforma UDDI, para una mejor organización del Servicio se requerirá:

- Realizar una descripción textual de las funciones generales que realiza el Servicio.
- Publicar el Servicio Web dentro de la categoría correspondiente.
- Realizar un continuo mantenimiento del repositorio UDDI en búsqueda de servicios no disponibles.
- Relacionar el documento WSDL con la API del manejo de las funciones del Servicio.
- Por otra parte, para realizar la publicación de un Servicio Web Semántico acorde a los conceptos WSMO, es necesaria la inserción del documento WSML en un repositorio WSMX; la categorización y descripción de los parámetros de Entrada/Salida ya se encuentran ubicados en el mismo documento por su relación con una determinada ontología.

6.3.3. REQUERIMIENTO DE BÚSQUEDA

Los requerimientos de búsqueda de un Servicio Web sintáctico se plantean por parte de los desarrolladores en forma manual, para posteriormente insertarlos en un repositorio UDDI público o privado donde se depende directamente de las opciones de búsqueda presentadas por el repositorio, en el caso de existir búsquedas avanzadas facilitará notablemente esta tarea, pudiendo categorizar los Servicios que se requiere.

En cambio, los requerimientos de búsqueda en los Servicios Web Semánticos se los plantea en base a un documento WSML de la ***Meta*** que se desea conseguir, donde se describirá exactamente los parámetros de entrada que se posee y los parámetros de salida que se requiere que posea el Servicio Web solicitado.

6.3.4. PROCESO DE BÚSQUEDA

El proceso de búsqueda de un Servicio Web sintáctico realizado a través de un repositorio UDDI, es una búsqueda de coincidencia de palabras, las mismas que se compararán con las descripciones de los Servicios Web publicados; en caso de existir búsquedas avanzadas se podrá categorizar o incrementar los parámetros de búsqueda como son, nombre de la empresa proveedora, categoría del Servicio Web requerido, valores que admite como entra y valores que entrega como salida.

En contraste, el proceso de búsqueda de los Servicios Web Semánticos, descrito en la sección 5.4.5. de esta investigación, se basa directamente en las definiciones, precondiciones y postcondiciones establecidas en la ***Meta***, otorgando un alto grado de confiabilidad en los resultados obtenidos.

6.3.5. RESULTADOS OBTENIDOS

Luego de realizar varias búsquedas en el repositorio jUDDI local, se obtiene un listado de los servicios que coinciden con los requerimientos ingresados como se trata en la sección 6.1.3., sin embargo, el listado obtenido se refiere a la coincidencia de palabras, mas no a la validez de los datos de entrada/salida con los que trabaje el Servicio listado.

Por otra parte, los resultados obtenidos de la búsqueda mediante WSMT y la herramienta WSML Discovery nos devuelve un listado de Servicios que coinciden semánticamente con los requerimientos ingresados, adicionalmente, el resultado de la búsqueda entrega una calificación lógica del resultado según las diferentes coincidencias de la Meta con el Servicio Web, véase la sección 5.4.5.

6.3.6. CRITERIO PARA LA SELECCIÓN

Finalmente, para realizar la selección – y posible utilización – de los Servicios Web se requerirá, en el caso de los Servicios Web sintácticos, verificar la existencia del Servicios, analizar los métodos y parámetros descritos en la API relacionada al servicio y verificar la calificación del Servicio por otros usuarios y/o empresas.

Para los Servicios Web Semánticos, es necesario apoyar la calificación obtenida por la herramienta con los comentarios y calificaciones expuestas por otros usuarios o empresas; pues, la concordancia de métodos ya viene dada por la incorporación de la semántica a través de la ontología.

6.4. ANÁLISIS CUANTITATIVO

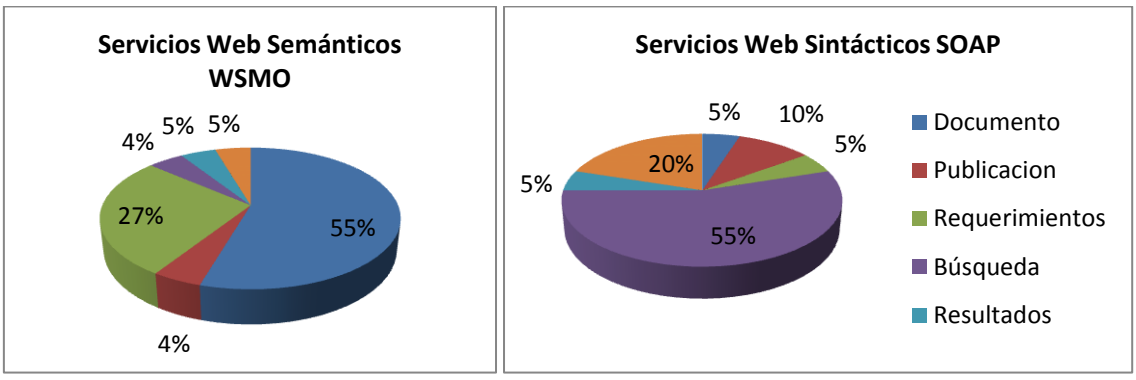
Una vez expuesta una comparación detallada de las diferencias entre las metodologías durante los procesos que conllevan al descubrimiento de servicios se realiza un análisis cuantitativo tomando como factores primordiales el tiempo transcurrido en cada fase y la claridad en la definición de conceptos.

TIEMPO EMPLEADO EN CADA FASE

Los tiempos utilizados para la generación de los documentos descriptivos, publicación, planteamiento de requerimientos, proceso de búsqueda, obtención de resultados satisfactorios y el proceso de selección se han medido asignando valores porcentuales de la siguiente manera:

- 5% para procesos automáticos y donde se ha requerido poca intervención del usuario.
- 10% para la publicación sintáctica debido al mejoramiento y categorización de servicios.
- 20% en la selección sintáctica pues requiere mayor verificación de campos externos.
- 25% a los requerimientos semánticos pues se requiere la generación del documento WSML relacionado a una ontología específica.
- 55% en el proceso de búsqueda sintáctica puesto que devuelven gran cantidad de servicios no utilizables lo que implica una gran pérdida de tiempo.
- 55% en la implementación de una ontología de conceptos y la descripción semántica de servicios, este tiempo invertido reduce notablemente el tiempo en las otras fases.

PASOS PARA EL DESCUBRIMIENTO	TIEMPO EMPLEADO (%)	
	Servicios Web SOAP	Servicios Web WSMO
Generación del documento descriptivo	5 %	55 %
Publicación del Servicio Web	10 %	5 %
Requerimientos de búsqueda	5 %	25 %
Proceso de búsqueda	55 %	5 %
Resultados Obtenidos	5 %	5 %
Criterios para la selección	20 %	5 %

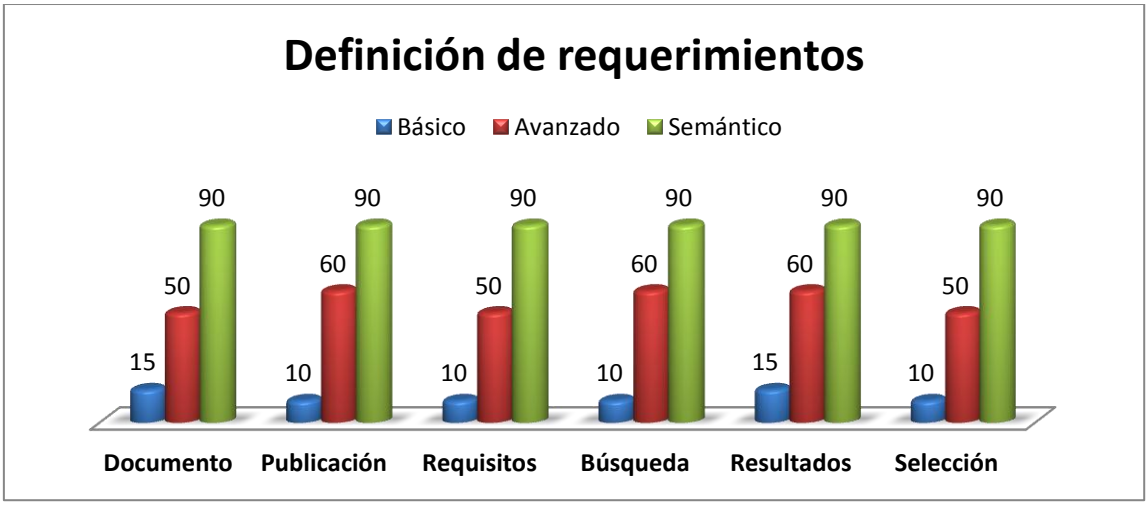


DEFINICIÓN DE CONCEPTOS

Otro factor esencial es la definición clara de los requerimientos de entrada y salida, métodos y parámetros tanto en el servicio como en las peticiones del usuario, así como la coincidencia con los resultados obtenidos, los que influencian en la futura selección de un servicio idóneo. Para este propósito se han comparado ambas metodologías y se asignaron valores porcentuales dependiendo del grado de definición de significados, quedando de la siguiente forma:

- 10% a la publicación, requisitos, búsqueda y selección sintáctica básica puesto que son procesos puramente manuales carentes de significado.
- 15% para la generación del documento descriptivo y obtención básica de resultados sintácticos, pues poseen definiciones básicas de conceptos.
- 50% del significado puede ser expresado mediante la descripción manual detallada de documentos WSDL y la obtención de resultados mediante búsquedas avanzadas que precisan mejor la información requerida.
- 60% en la exactitud de conceptos puede lograrse con el uso de opciones avanzadas en la publicación y búsqueda, logrando resultados con mucho más exactos que con un trabajo a nivel básico.
- En la metodología semántica WSMO se logra una definición clara de conceptos al ser relacionados con una ontología bien definida, por lo cual su exactitud será de un 90% o superior dependiendo de las destrezas del equipo de desarrollo.

PASOS PARA EL DESCUBRIMIENTO	SOAP Básico	SOAP Avanzado	WSMO
Generación del documento descriptivo	15 %	50 %	+ 90 %
Publicación del Servicio Web	10 %	60 %	+ 90 %
Requerimientos de búsqueda	10 %	50 %	+ 90 %
Proceso de búsqueda	10 %	60 %	+ 90 %
Resultados Obtenidos	15 %	60 %	+ 90 %
Criterios para la selección	10 %	50 %	+ 90 %



CAPITULO 7:

PROYECCIONES FUTURAS BASADAS EN WSMO

7.1. PAQUETES JAVA

Actualmente el trabajo e investigación sobre la metodología y framework WSMO se encuentra aún en estado de desarrollo bajo una orientación Open Source, es por ello que se involucran varias herramientas externas como es el entorno de ejecución WSMX y repositorios como SESAME; estas implementaciones y distribuciones trabajan de manera conjunta y se encuentran en un estado de beta permanente ofreciéndose los paquetes de desarrollo para Java.

7.1.1. PROYECTO WSMO4J[80]

El núcleo del trabajo WSMO se basa en los objetos principales de la metodología como son Ontologías, Servicios, Metas y Mediadores; el proyecto WSMO4J es un proyecto Java desarrollado en el entorno Eclipse para la implementación de Servicios Web basándose en el lenguaje WSML con el propósito de permitir a los usuarios crear aplicaciones Java que hagan uso del modelado de Servicios Web Semánticos basados en los conceptos WSMO.

Este proyecto consta de tres librerías principales:

- wsmo4j-api.jar
- wsmo4j.jar
- WSML-grammar.jar

7.1.1.1. wsmo4j-api.jar

El módulo wsmo4j-api contiene varias clases e interfaces de gran utilidad para los desarrolladores, estas clases e interfaces están divididas en paquetes que conforman grupos de conceptos principales.

En el paquete org.omwg.ontoly.* se manejan las bases para el modelado de ontologías comunes en lenguaje OWL.

El paquete org.omwg.logicalexpression.* contempla interfaces de razonamiento lógico para trabajar sobre ontologías con expresiones lógicas atómicas o compuestas, permitiendo tratar conceptos como conjunción, disyunción, equivalencia, implicación, implicación inversa, etc.

80 El proyecto Open Source WSMO4J puede ser orientado a varios tipos de proyectos por lo cual se ha tomado como fuentes primordiales esenciales los trabajos más evolucionados **[70]** y **[71]**

Dentro del paquete org.wsmo.common.* se maneja las clases e interfaces centrales para el modelado de objetos WSMO (Ontologías, Servicios, Metas y Mediadores) mediante el manejo de entidades, identificadores, namespace, no functional properties.

El paquete org.wsmo.service.* contiene las interfaces relacionadas al trabajo con objetos WSMO tratadas de manera indiferente, dentro de este paquete podemos encontrar `ServiceDescription, Capability, Interface`.

7.1.1.2. wsmo4j.jar

Contiene los paquetes esenciales para el trabajo con el resto de clases e interfaces como son Factories, Parsers y Serializers, Datastore, Repositories y Validadores que involucran el trabajo de organización y manejo de los diferentes conceptos WSMO.

El paquete wsmo4j contiene la implementación de varias interfaces descritas en wsmo4j-api además de otros componentes relacionados que no son de gran interés para la mayoría de usuarios, sin embargo, podrían ser utilizados por los usuarios que intención de modificar el código central para trabajos de campo específicos.
Adicionalmente, el paquete wsmo4j contiene varios sub paquetes que hacen referencia a patrones de aprendizaje, importación de archivos .wsml, métodos de creación de ontologías mediante programación y validadores del código WSML, los cuales orientan a los usuarios a la utilización de una variante del lenguaje WSML; finalmente, se utiliza test.wsmo4j.* para el agrupamiento de escenarios de prueba que pueden servir de ayuda a los programadores.

El funcionamiento de estas interfaces y clases se centra en la utilización de wsmo.factory el cual controla la creación de otras interfaces como son DataFactory y LogicalExpressionFactory; la clase WSMOParser está compuesta de analizadores sintácticos que pueden ser adicionados mediante otros paquetes, entre estos podemos nombrar:

- Pellet
- Mins
- Kaon2
- IRIS

7.1.2. OTROS PAQUETES

7.1.2.1. WSML2reasoner

Otro proyecto relacionado a las teorías WSMO es el razonador del lenguaje WSML que permite trabajar con Ontologías escritas en WSML, el propósito de este proyecto es la implementación de una librería Java que permita el razonamiento acorde a los lineamientos WSMO.

Este proyecto incorpora varios razonadores como Mins y Pellet que se ofrece bajo licencia LGPL, sin embargo admite la incorporación de razonadores externos como IRIS y Kaon2, éste último ofrece una versión no comercial bajo licencia GPL y una versión comercial llamada OntoBroker OWL para trabajos con WSMO y OWL.

La intención de generar un razonador como WSML2Reasoner es mejorar las búsquedas en las Ontologías WSML, mediante este razonador –incorporado en la herramienta WSMT- se tiene una sintaxis similar a SQL y permite trabajar con cualquiera de las variantes de WSML (WSML-Core, WSML-Rule, WSML-Flight, WSML-Full). La figura 54 muestra la importación del proyecto wsml2reasoner a la plataforma Eclipse, pues está conformado de varias librerías Java que sirven para utilizar el razonador WSML en la programación Java.[81]

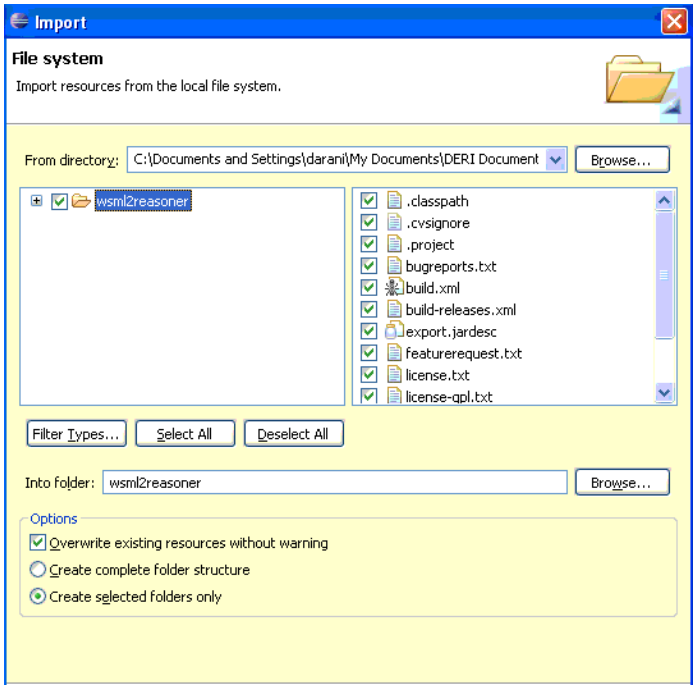


Figura 54. Importación a Eclipse de wsml2reasoner[82]

7.1.2.2. WSMX

El entorno de ejecución para objetos WSMO también es una herramienta Open Source desarrollada en Java que ofrece el proyecto a través de varias opciones como son:

- Subversion Eclipse
- Distribución Maven
- Paquete wsmx.jar
- Ambiente Apache

81 Mayor información sobre la implementación de la herramienta en Eclipse puede ser consultada en **[50]** los conceptos relacionados a la herramientas se describen con mayor detalle en **[72]** y **[73]**

82 Paquetes que conforman WSML2Reasoner en vista de importación eclipse **[50]**

Para el presente trabajo se ha descargado la versión para el ambiente Apache, que nos permite interactuar con WSMX mediante un navegador web en la dirección por default localhost:8081, sin embargo, las otras opciones permiten interactuar con WSMX sin depender del uso del navegador, el detalle de las descargas e instalación de estas opciones se precisa mejor en el Anexo 6.

Tanto la Subversion como el paquete wsmx.jar ofrecen el código para la implementación de este ambiente de ejecución desde la plataforma Eclipse, sin embargo, presentan varias incongruencias al momento de ser incorporadas con errores redundantes ocasionados por la inexistencia de librerías necesarias para que el proyecto funcione adecuadamente.

Otra alternativa para este ambiente de ejecución es la distribución MAVEN que permite la ejecución de WSMX en un ambiente fuera del navegador, cabe destacar, que dentro del sitio oficial WSMX se encuentra la documentación de referencia indicando claramente que es un proyecto en continuo desarrollo, que sin embargo, no ha llegado a un punto de madurez satisfactorio como para ser utilizado en un ambiente de producción.[83]

El ambiente de ejecución de la distribución MAVEN del WSMX se muestra la figura 55.

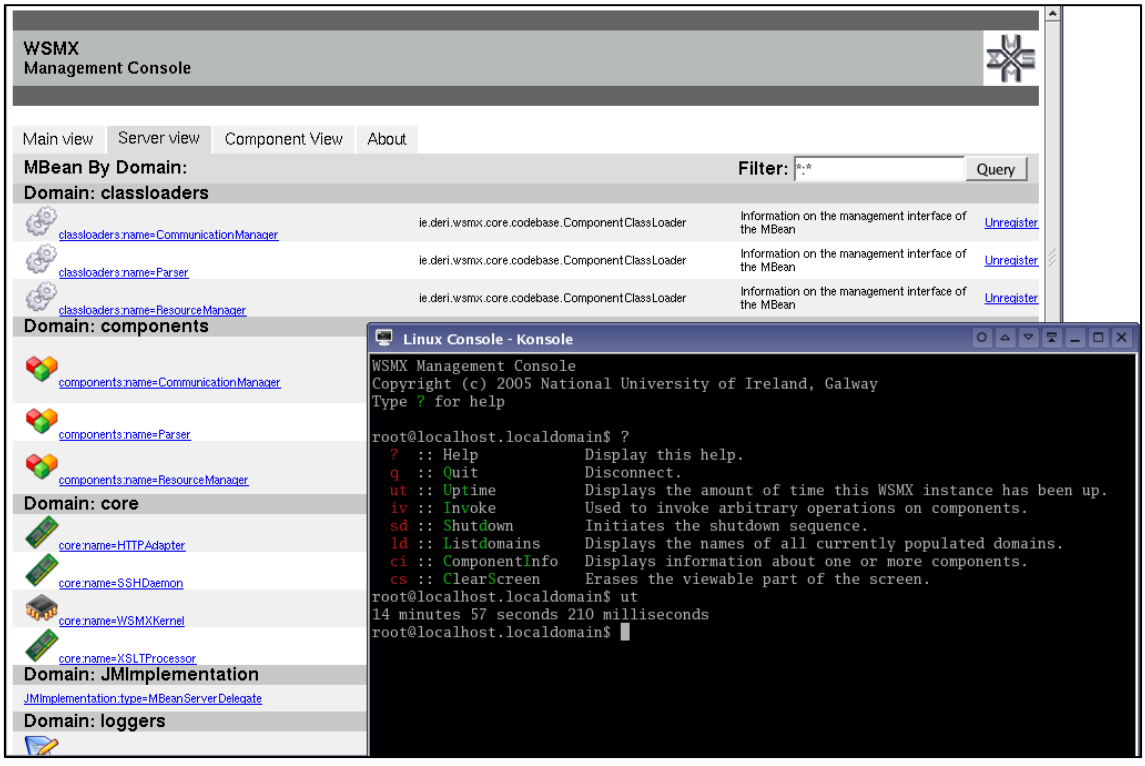


Figura 55. Ambiente de ejecución WSMX (Ambiente MAVEN)

83 Las diferentes versiones de la herramienta WSMX pueden ser encontradas en [www.wsmx.org] así como los conceptos y forma de utilización

7.2. TRABAJOS BASADOS EN WSMO

En el Capítulo 4 se trataron diferentes frameworks de trabajo con Servicios Web Semánticos, dentro de los cuales se analizaron Infrawebs e IRS-III, los mismos que se basaban en los fundamentos WSMO para incrementar su estudio.

En las secciones posteriores se tratará sobre estos proyectos y las expectativas a futuro con la herramienta WSMO.

7.2.1. INFRAWEBS

El proyecto Infrawebs consiste en diseñar, mantener y ejecutar Servicios Web Semánticos (SWS) basados en Servicios Web existentes diseñados acorde la WSMO framework. El INFRAWEBS SWS Designer es una herramienta para convertir de forma semiautomática los Servicios Web no semánticos a semánticos.

El INFRAWEBS DESIGNER es una aplicación Java de escritorio basada en la plataforma cliente de Eclipse que provee una herramienta para buscar y seleccionar Servicios Web desde un almacén remoto (SIR) mediante un formulario de consulta ubicado en el comunicador de la aplicación.

Esta herramienta además tiene en cuenta las ontologías por demanda, es decir, aquellas ontologías que poseen conceptos que son utilizados por la ontología cargada, y en ese caso se actualizará automáticamente los campos similares cargando los nuevos atributos o reglas, la figura 56 muestra la herramienta INFRAWEBS DESIGNER[84]

Figura 56. INFRAWEBS DESIGNER[85]

84 Introducción y teoría de la herramienta en **[48]** y **[61]** donde se podrá encontrar información más detallada

85 Vista obtenida del documento **[61]** donde se describe el manejo e instalación de la INFRAWEBS

7.2.2. IRS III

El proyecto IRS-III como se había tratado en la sección 4.4. consta de varios componentes como son IRS Server, IRIS Publisher e IRIS Client donde cada uno de ellos cumple una función específica, de manera general IRS-III permite crear y editar conceptos WSMO, aplicaciones Web y descripciones WSDL de los Servicios creados, además de la invocación de Servicios a través de una Meta alcanzada.

La herramienta de trabajo IRS-III es el IRS-III Browser/Editor que hace uso de los componentes antes mencionados, esta herramienta ha sido desarrollada en Java con la utilización del proyecto de paquetes wsmo4j mencionado en secciones anteriores.

El objetivo principal de esta herramienta es que el usuario controle los diferentes pasos en la composición de Servicios, de forma similar a WSMO el trabajo de los Servicios y las Ontologías se realizan con la utilización de Metas y sus descripciones en WSML la figura 57 muestra la herramienta.[86]

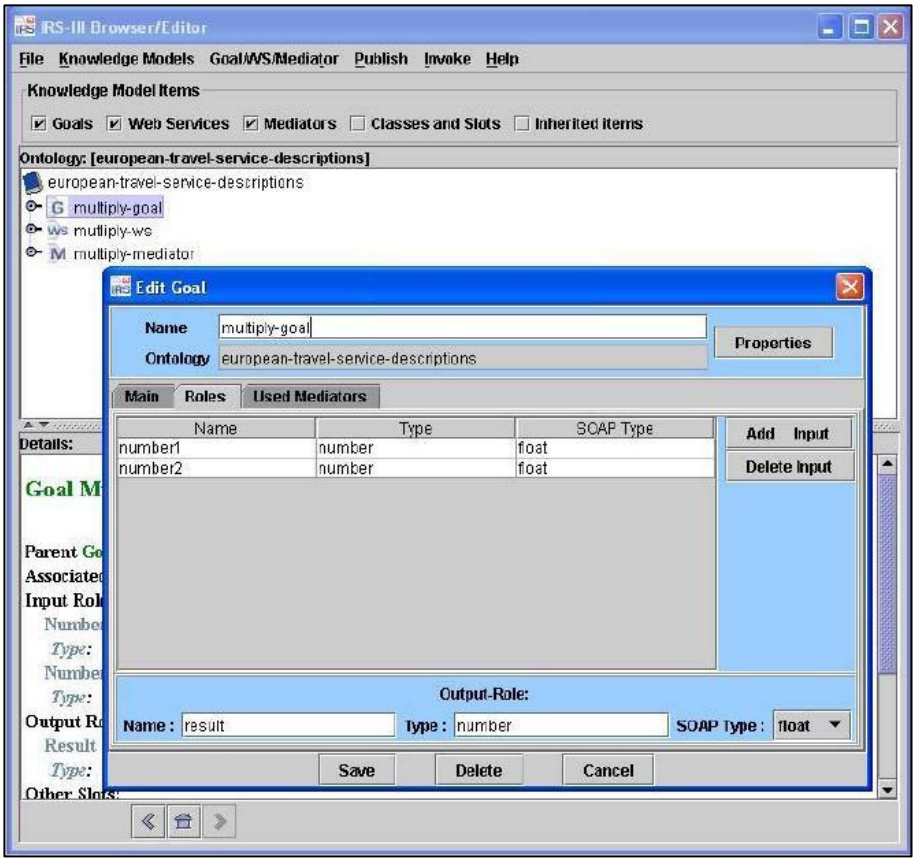


Figura 57. IRS-III Browser/Editor[87]

86 Visión general de la herramienta se ha extraído de **[47]** y **[62]**

87 Tomada del tutorial introductorio **[62]**

7.2.3. WSMO STUDIO

Es una herramienta desarrollada por las empresas DERI y OntoTextLab para el trabajo con Servicios Web Semánticos bajo los conceptos WSMO, permite el modelado de los procesos de negocios basados en Ontologías. Esta herramienta está integrada por varios elementos como son:

- **WSMO Editor Launcher:** Este editor permite la creación de las descripciones WSML tanto de los Servicios Web como de los Mediadores de una manera más legible que los editores textuales. Adicionando la utilización de Mediadores para la interpretación y descubrimiento del Servicio Web.

- **SAWSDL Editor:** Es un editor para documentos de descripción de Servicios WSDL, que permite la agregación de anotaciones semánticas a las descripciones WSDL y XML.

De manera similar a los casos anteriores, esta herramienta ha sido diseñada para trabajar con los conceptos WSMO, adicionando interfaces mejoradas que permiten añadir contenido semántico a los Servicios Web importados, la figura 58 muestra la herramienta WSMO Studio.

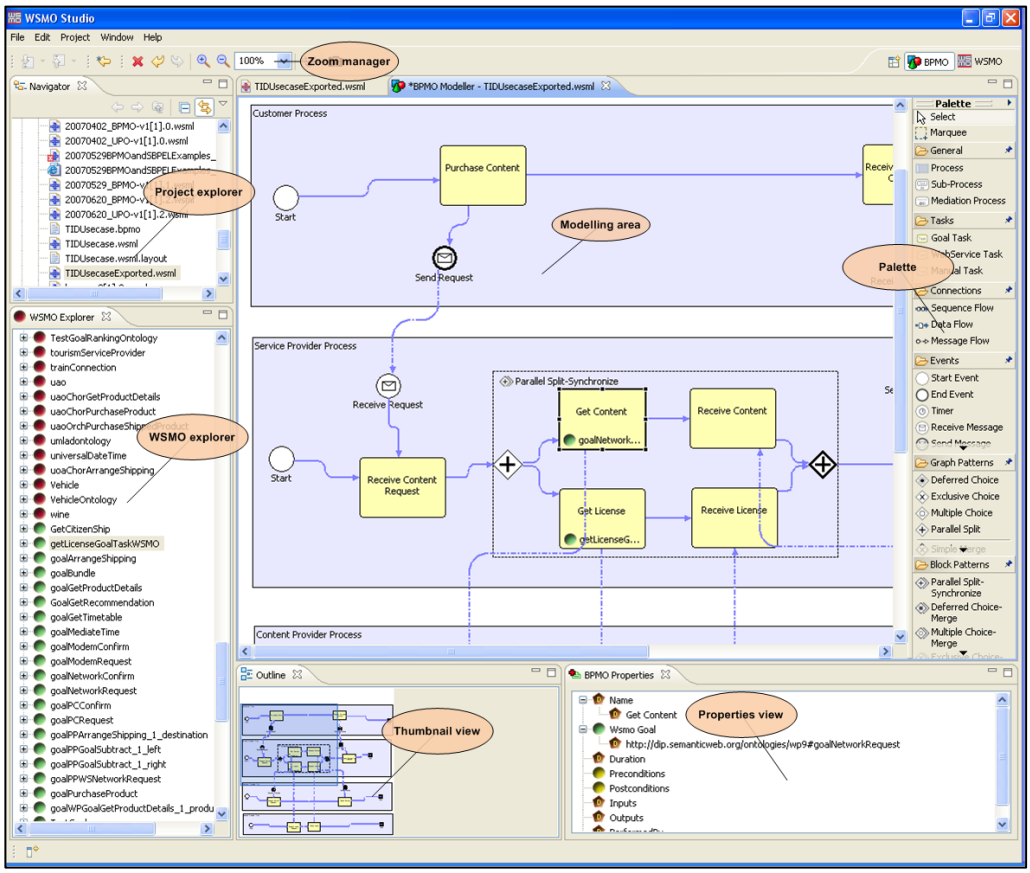


Figura 58. Herramienta WSMO Studio[88]

88 **Herramienta WSMO Studio:** Vista del editor BPMO provisto en la herramienta, disponible en [http://www.wsmostudio.org/doc/screenshots/]

7.3. ESPECTATIVAS FUTURAS

Como se ha tratado en el presente Capítulo existen varias líneas de investigación y ejecución de los conceptos desarrollados con WSMO y la utilización de su herramienta de modelado WSMT, el entorno de ejecución WSMX y el repositorio de Servicios IRS; sin embargo, cabe destacar que la presente investigación se basa en proyectos piloto que no son utilizados para ambientes de producción.

Durante la ejecución de la investigación se han logrado conocer las características de algunos de los proyectos mencionados y la intención de los equipos de investigación en seguir mejorando el framework WSMO como tal.

Los avances con esta herramienta y el lenguaje WSML permiten una mejor estructuración de los conceptos semánticos en Ontologías y Servicios Web, y es por tal motivo que han sido tomados en cuenta por el W3C como una esperanza de estandarización para el trabajo con Servicios Web Semánticos.

Las librerías Open Source existentes wsmo4j, wsmx, wsml2reasoner permiten medianamente el trabajo de programación de los Servicios Web Semánticos en la plataforma de desarrollo Eclipse, no obstante, con el mejoramiento de las herramientas y la publicación de documentación adecuada (escasa hasta el momento de terminada esta Tesis) se podría pensar en lograr el ansiado objetivo del descubrimiento automático de Servicios Web.

CONCLUSIONES

A continuación se presentan las principales conclusiones obtenidas en el desarrollo de esta investigación:

- La utilización de un único lenguaje (WSML) para la generación de descripciones tanto de Servicios Web como Metas y Ontologías mejora sustancialmente la interacción entre estos elementos.

- El uso de una herramienta (WSMT) que unifique el modelado tanto de Ontologías y descripciones semánticas de Servicios Web facilita la comprensión del comportamiento de los Servicios Web Semánticos en ambiente de producción

- Los procesos involucrados en el ciclo de vida de los Servicios Web Semánticos son comprendidos y ejecutados con mayor facilidad mediante la inclusión de los conceptos WSMO (Ontologías, Servicios Web, Metas y Mediadores) en cada uno de los pasos que lo componen.

- La utilización de una misma Ontología (o un adecuado mapeo de Ontologías) para la definición de Servicios Web, Metas y Mediadores facilita la comprensión y el éxito en los procesos involucrados en el Ciclo de Vida de los Servicios Web Semánticos.

- El tiempo invertido en la correcta definición semántica de los Servicios Web en WSML se ve notablemente recuperado en las subsiguientes fases que conforman el Ciclo de Vida de los Servicios Web.

- Una publicación correctamente definida de Servicios Web sintácticos no asegura un descubrimiento idóneo de los Servicios Web, pues, no se garantiza la correspondencia entre los parámetros requeridos y obtenidos del Servicio.

- El trabajo con la metodología WSMO ofrece un ambiente completo para el Modelado y futura aplicación de los Servicios Web Semánticos otorgando grados de coincidencia claramente utilizables por los desarrolladores.

- El descubrimiento automático de Servicios Web es un objetivo evidentemente alcanzable mediante la utilización de una misma metodología en el transcurso del Ciclo de Vida de los Servicios Web

- La inclusión de contenido semántico en las descripciones de los Servicios promueve la creación de nuevas herramientas y metodologías orientadas al descubrimiento y composición automático de Servicios Web.

- La invocación automática de Servicios Web es aún un objetivo difícil de alcanzar con la utilización de las herramientas WSMO, inclusive mediante la ejecución del código provisto con wsmo4j, no así el razonamiento y descubrimiento de Servicios.

RECOMENDACIONES

- Durante la publicación de Servicios en repositorios UDDI es recomendable realizar una categorización taxonómica de los servicios, con el objetivo de mejorar las búsquedas.
- Trabajar con software para el análisis de los documentos WSDL obtenidos como resultado de una búsqueda en repositorios UDDI para mejorar la confiabilidad de los resultados.
- Como pilar fundamental para la comprensión e incorporación de Semántica en el contenido web es necesario enfocar el conocimiento académico en la creación de documentos XHTML y XML semánticamente bien estructurados, que permitan la agregación de información semántica en el contenido web.
- Incentivar la inclusión de información semántica en los contenidos web de la UTPL a través de gestores de contenido especializados con el uso de microformatos y metodologías afines.
- Incursionar en el manejo de la herramienta WSMT y los conceptos WSMO con fines académicos, pues, muestran un panorama más amplio para el manejo de los procesos involucrados en el Ciclo de Vida de los Servicios Web.
- Continuar con el manejo de líneas de investigación relacionadas a la Web Semántica y Servicios Web Semánticos con la finalidad de orientar mejor a los futuros estudiantes.
- Incentivar la investigación enfocada en los recursos de código libre y paquetes Java orientados al manejo de los Servicios Web Semánticos mediante los conceptos WSMO tratados en esta investigación.
- Promover la utilización del lenguaje WSML y el conjunto de herramientas WSMT a los estudiantes de la Escuela de Ciencias de la Computación de la UTPL para la creación de ontología y semántica aplicada.

GLOSARIO DE TÉRMINOS

ADUNA: El equipo central de la compañia esta formada por profesionales IT con una larga trayectoria en negocios trabajando juntos por más de una década. [http://www.aduna-software.com]. Sección 2.1.2.1.

AGENTES SOFTWARE: Programas de computación que interactúan con la información y conceptos disponibles en la World Wide Web a través de estándares como el lenaguaje XML y protocolos como HTTP. Secciones 1.3.1., 1.3.5., 2.2., 2.3., 3., 3.4.

API: Interfaz de programación de aplicaciones o **API** (del inglés *application programming interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Usados generalmente en las bibliotecas. Sección 1.2.2., 2.1.2.2., 4.4.2., 6.1.2., 6.3.2., 6.3.6., 7.

AXIOMA. AXIOM: Se refiere a los conceptos de axiomas lógicos para el razonamiento sobre los conceptos definidos en una ontología WSMO, las mismas que se identifican por la palabra reservada ***axiom*** en el lenguaje WSML. Secciones 5.1.1., 5.2., 5.2.2.

BIBO: La Ontología Bibliográfica describe tipos bibliográficos sobre la Web Semántica en RDF. Esta ontología puede ser usada como una citación ontológica, como un documento de clasificación de ontologías, o simplemente como una forma de descripción de cualquier tipo de documento en RDF. Esta siendo inspirada por varios formatos de metadatos existentes de metadatos, y pueden ser utilizados como una base común para convertir otros recursos de datos bibliográficos. **[**http://bibliontology.com/]. Secciones 5.4.

BPEL: Business Process Execution Language ó Lenguaje de Ejecución de Procesos de Negocios, WS-BPEL, *Web Services Business Process Execution Language ó Lenguaje de Ejecuciónes un Lenguaje Estándar de Ejecución para las interacciones específicas con Servicios Web.La información de los proceso en BPEL son* exportados and importados usando interfaces de Servicios Web exclusivamente. [http://www.sti-innsbruck.at/]. Sección 4.1.2.

BPEL4WS: Business Process Execution Language for Web Services ó Lenguaje de Ejecución de Procesos de Negocio, provee un significado formalmente específico para procesos de negocio y la interacción de protocolos. Define un modelo de integración interoperable que debería facilitar la expansión de procesos automáticos de integración en ambos espacios intra-corporativos y B2B Herramienta desarrollada por IBM para el trabajo con negocios complejos que involucran Servicios Web. Sección 4.1.

COMPOSICIÓN: Es una fase del Ciclo de Vida de los Servicios Web, donde se trata de la colaboración manual o automática de uno o más Servicios Web, la composición sintáctica de Servicios Web se realiza de forma manual identificando en primera instancia los Servicios que componen la colaboración; la composición automática se encuentra aún en fase de estudio y hacen uso de los conceptos de coreografía y orquestación. Secciones 3.2., 3.2.4.

COREOGRAFÍA: Establece el comportamiento de los Servicios Web frente a las acciones solicitadas por un usuario, pudiendo ser humano u otro servicio. Las tareas de coreografía se complementan con el trabajo de orquestación. Sección 3.3.1.

DCMI[i] Dublin Core Metadata Initiative, "DCMI", es una organización abierta, ocupada en el desarrollo de la interoperatividad de estándares de metadatos que soportan un amplio rango de propositos y modelados de negocios. Las actividades de DCMI incluye trabajos sobre arquitecturas y modelados, discusiones y trabajo colaborativo entre comunidades y grupos de trabajo DCMI. [http://dublincore.org/]. Sección 2.1.1.4.

DERI: Instituto de Tecnología Semántica (STI)) Innsbruck, formalmente que **DERI Innsbruck**, fue fundada por la Univ.-Prof. Dr. Dieter Fensel en el año 2002 y fue desarrollado en una desafiante y dinámica investigación de aproximadamente 60 personas. STI Innsbruck colabora con una Red Internacional de Institutos en Asia, Europa y USA. Sección 7.3.2.

DESCUBRIMIENTO: Se refiere al proceso de búsqueda y selección de un Servicio Web, ya sea por metodología sintáctica o semántica. Sección 3.2.2.

DOCUMENTO DESCRIPTIVO: Un documento descriptivo de un Servicio Web está definido en un lenguaje estándar para la descripción de las Entradas/Salidas y métodos que conforman el Servicio, estos lenguajes pueden ser WSDL, WADL, WSDL-S, SAWSDL, SAREST o WSML. Sección 1.2.1.

DUBLIN CORE[i] Organización dedicada a la estandarización de conceptos ontológicos a ser manejados en ambiente web, refiérase a DCMI. Capitulo 5.

eRDF: Microformato basado en el lenguaje RDF que puede generar conceptos básicos de un solo nivel y puede ser incluido dentro de un documento HTML. Sección 2.1.1.2.

FRAMEWORK: Conjunto de librerías que conforman una plataforma para el desarrollo de aplicaciones, puede estar unido a un IDE. Secciones 2.1.2., 4.

GOAL: Es uno de los conceptos WSMO mediante el cual se definen los requerimientos de un usuario para realizar la búsqueda de un Servicio Web Semántico definido en WSML. Sección 4.5.1., 5.1.3.

GRDDL: Es un mecanismo para obtener una descripción semántica de recursos a partir de documentos XML, Secciones 1.1.7., 2.1.1.3.

HTML: Lenguaje de marcado de hipertexto, es la base para el desarrollo de los documentos de hipertexto en la World Wide Web, en las cuales se generan vínculos interactivos.

HTTP: Protocolo de transferencia de hipertexto. Protocolo utilizado en cada transacción de la World Wide Web fue desarrollado por W3C

hREST: Microformato llamado HTML RESTful que es utilizado para agregar información semántica en los Servicios Web basados en REST. Sección 5.3.1.

jUDDI: Repositorio UDDI basado en Java y que utiliza como plataforma Apache Tomcat, puede ser utilizado tanto como repositorio público o privado de documentos WSDL. Sección 6.1.1.

JENA: Repositorio Open Source de estructuras semánticas RDF elaborado en Java, permite la recopilación de información semántica, extracción, administración e inserción a través de un Servicio Web. Sección 2.1.2.2.

KAON2: Es una infraestructura para administración de ontologías como OWL-DL, SWRL, y F-Logic. Creada por el esfuerzo conjunto de las siguientes instituciones:

- Information Process Engineering (IPE) at the Research Center for Information Technologies (FZI)
- Institute of Applied Informatics and Formal Description Methods (AIFB) at the University of Karlsruhe
- Information Management Group (IMG) at the University of Manchester. Sección 7.1.1.2.

MAPPING: Se refiere al mapeo (búsqueda equivalente) de información semántica entre dos estructuras ontológicas diferentes, es la base para el trabajo de Servicios Web con conceptos similares expresados en conceptos y ontologías distintas. Sección 5.1.5.

MEDIADOR. MEDIATOR: Concepto WSMO referente al trabajo de razonamiento entre los demás conceptos WSMO (Ontologías, Servicios y Metas) existen varios tipos de mediadores: OO-Mediator, WW-Mediator, WG-Mediator y GG-Mediator. Sección 5.1.4.

MICROWSMO. WSMO-Lite: Tecnología complementaria a los conceptos WSMO, permite el trabajo semántico con Servicios Web basados en REST, mediante el uso del microformato hREST y el mecanismo GRDDL. Sección 5.3.1.

MIME: Extensiones de Correo de Internet Multipropósito. Especificaciones dirigidas para intercambiar a través de Internet todo tipo de archivos (texto, audio, video, etc.)

NPF.NonFunctionalProperties: Referente a la descripción de propiedades no funcionales dentro de la definición de los diferentes conceptos WSMO en lenguaje WSML, es también una palabra reservada ***nonFunctionalProperties*** para la declaración de información textual de los conceptos WSMO. Sección 6.2.3.

OASIS: (Organization for the Advancement of Structured Information Standards) es un consorcio sin fines de lucro que conduce el desarrollo, convergencia y adopción de estándares abiertos para la sociedad global de información. Este consorcio produce más estándares de Servicios Web que cualquier otra organización. [http://www.oasis.org]. Sección 1.2.2.2.

ORQUESTACIÓN: Concepto complementario a la Coreografía de Servicios Web. Establece y controla un mecanismo para el intercambio de mensajes entre Servicios.

OWL: Lenguaje de Ontologías Web. Lenguaje estandarizado por W3C para la generación de ontologías [http://www.w3.org/TR/owl-semantics]

PELLET: es un OWL2Reasoner provee estándares y razonares de servicios para las ontologías OWL. La versión Pellet 2.0 está ya liberada para su descarga. [http://clarkparsia.com]. Seccion 7.1.1.2.

POSTCONDICIÓN: Parámetros de salida de información de un Servicio Web, refiriéndose al tipo de dato, longitud de caracteres, concepto relacionado, etc. Es un parámetro principal para la descripción semántica de Servicios Web y Metas. Sección 5.2.3.

PRECONDICIÓN: Parámetros requeridos para el ingreso de la información a un Servicio Web, refiriéndose al tipo de dato, longitud de caracteres, concepto relacionado, etc. Es un parámetro principal para la descripción semántica de Servicios Web y Metas. Sección 5.2.3.

PUBLICACIÓN: Se refiere a la exposición pública de un Servicio Web sintáctico a través del documento descriptivo WSDL en un repositorio UDDI público o privado, refiriéndose también a la publicación de Servicios Web Semánticos en un repositorio WSMX. Sección 3.2.1.

REST: Técnica moderna de arquitectura de software principalmente orientado a la WWW basado en intercambio de mensajes XML sobre el protocolo HTTP. Sección 1.2.1.2.

RDF Schema: es una extensión semántica de RDF. Un lenguaje primitivo de ontologías que proporciona los elementos básicos para la descripción de vocabularios. La primera versión fue publicada en abril de 1998 por la W3C, la versión actual de la recomendación fue publicada en febrero de 2004 también por la W3C. Existen actualmente otros lenguajes de ontologías más potentes, como puede ser OWL. Sección 1.3.1.

RDF: Es un direccionador de etiquetas gráficas de formato de datos para representar información en la Web. RDF es a menudo usado para representar, entre otras cosas, información personal, redes sociales, metadatos acerca de artifactos digitales, que proveen un significado de integración sobre recursos dispersos de información. Esta especificación define la sintaxis y semántica del lenguaje de consulta SPARQL para RDF. [http://www.w3.org/TR/rdf-syntax-grammar/]. Sección 1.3.3.

RFC: **R**equest **F**or **C**omments (*"Petición De Comentarios"* en español) son una serie de notas sobre Internet que comenzaron a publicarse en 1969. Cada una de ellas individualmente es un documento cuyo contenido es una propuesta oficial para un nuevo protocolo de la red Internet (originalmente de ARPANET), Cualquiera puede enviar una propuesta de RFC a la IETF, pero es ésta la que decide finalmente si el documento se convierte en una RFC o no. Si luego resulta lo suficientemente interesante, puede llegar a convertirse en un estándar de Internet. Cada RFC tiene un título y un número asignado, que no puede repetirse ni eliminarse aunque el documento se quede obsoleto. Ejemplo: el protocolo IP se detalla en el RFC 791, el FTP en el RFC 959, y el HTTP (escrito por Tim Berners-Lee, entre otros) el RFC 2616. Sección 2.1.1.1.

RPC: lamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. Algunos tipos son: el RPC de Sun denominado ONC RPC (RFC 1057), el RPC de OSF denominado DCE/RPC y el Modelo de Objetos de Componentes Distribuidos de Microsoft DCOM, aunque ninguno de estos es compatible entre sí. La mayoría de ellos utilizan un lenguaje de descripción de interfaz (IDL) que define los métodos exportados por el servidor. Sección 1.2.

SA-REST: Metodología de agregación semántica al contenido web desarrollado en REST se basa en el uso del lenguaje RDF de manera similar a WSDL-S. Sección 2.2.4.

SAWSDL: Metodología de agregación semántica al contenido web desarrollado en SOAP con la existencia de un documento WSDL, la agregación de semántica es un enlace hacia contenido semántico externo en lenguaje RDF. Sección 2.2.3.

SERQL: "Sesame RDF Query Language", pronounced "circle" es un nuevo lenguaje de consulta RDF/RDFS que está actualmente siendo desarrollado por Aduna como parte de Sesame. Combina las mejores características de otros lenguajes de consulta (RQL, RDQL, N-Triples, N3) y adiciona algunos propios. [http://www.openrdf.org/doc/sesame/users/ch06.html]. Sección 2.1.2.1.

SMTP: Simple Mail Transfer Protocol (SMTP) ó Protocolo Simple de Transferencia de Correo, es un protocolo de la capa de aplicación. Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos (PDA's, teléfonos móviles, etc.). Está definido en el RFC 2821 y es un estándar oficial de Internet. Sección 1.2.1.1.

SOAP: Protocolo Simple de Acceso a Objetos. Define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de mensajes XML. Sección 1.2.1.1.

SPARQL: Es un lenguaje de consulta para RDF esta diseñado para encontrar los casos de uso u los requerimientos identificados por el Grupo de Trabajo de RDF de Acceso a Datos en RDF Data Access Use Cases and Requirements [UCNR]. [http://www.w3.org/TR/rdf-sparql-query/#introduction]. . Sección 2.1.2.1.

SWSF: Semantic Web Services Framework que significa Framework para Servicios Web Semánticos, que incluye lenguajes de Servicios Web Semántica ó Semantic Web Services Language (SWSL) y Ontologías de Servicios Web Semántica ó Semantic Web Services Ontology (SWSO). [http://www.w3.org/Submission/SWSF/]. Sección 4.2.

TOMCAT: (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems. APACHE [http://tomcat.apache.org/]. Sección 6.1.1.1.

UDDI : son las siglas del catálogo de negocios de Internet denominado *Universal* ***Description, Discovery and Integration***. Es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el contexto de los servicios Web. Sección 6.1.1.

URI: Identificador Uniforme de Recursos. Es una cadena de caracteres que identifica inequívocamente un recurso (servicio, página, documento, e-mail, etc.) . Sección 1.3.

URL: son las siglas de **Localizador Uniforme de Recurso** (en inglés *Uniform Resource Locator*), la dirección global de documentos y de otros recursos en la World Wide Web. La primera parte de la dirección indica qué protocolo utilizar, la segunda parte especifica la dirección IP o nombre de dominio donde se localiza el recurso. Sección 1.3.

URN: es un acrónimo inglés de ***Uniform Resource Name***, al español "Nombre de recurso uniforme". Éste trabaja de manera similar a un URL. Éstos identifican recursos en la web, pero a diferencia de un URL, no indican exactamente dónde se encuentra ese objeto. Básicamente un URI(Uniform Resource Identifier) = URL + URN. Sección 1.3.

W3C: World Wide Web Consortium encargado del desarrollo de tecnologías y estándares para potenciar Internet a su máxima capacidad. Mayor información en www.w3c.org

WADL: Lenguaje para Descripción de Aplicaciones Web, tomado como base para la descripción de servicios web REST, actualmente se utiliza el estándar WSDL 2.0. Sección 1.2.2.

WSDL: Lenguaje de descripción de servicios, más información en [http://www.w3.org/TR/wsdl] . Sección 1.2.2.

WSFL: Web Services Flow Language es un leguaje para la descripción de composición de servicios Web. Considera dos tipos de composiciones de servicios web: El uso apropiado de patrones de procesos de negocio y la interacción de dichos patrones con el conjunto de negocios. Sección 4.1.1.

WSMF: Framework para el Modelado de Servicios Web ó Web Service Modeling Framework que provee el modelo conceptual apropiado para el desarrollo y descripción de los Servicios Web y su composición. Sección 4.3.1.

WSMO: Ontología para el Modelado de Servicios Web. Es un framework que permite el completo modelado de Servicios Web Semánticos con sus conceptos principales (Ontologías, Servicios, Metas y Mediadores), hace uso de la herramienta WSMT y razonadores de lenguaje WSML.

WSMOSTUDIO: WSMO Studio es una herramienta Open Source para el trabajo con Servicios Web Semánticos, utilizado como plataforma de Modelado de Procesos de Negocios, así como también para el Modelado de Ontologías de Servicios Web. WSMO Studio e incluye razonadores

WSML Reasoner (MINS, KAON2, Pellet, IRIS), editores de código (WSML, OWL-DL, RDF) y trabaja con repositorios ontológicos IRS-III, WSMX y ORDI. [http://www.wsmostudio.org]. Sección 7.2.3.

WSML: Lenguaje para el Modelado de Servicios Web. Lenguaje base para el trabajo con el framework WSMO. Sección 5.2.

WSML2Reasoner: Es un arquitectura altamente modular que combina varias validaciones, normalizaciones y funciones esenciales de transformación para la traducción de descripciones de ontologías en WSML. Sección 7.1.2.1.

WSMT: Conjunto de herramientas para el Modelado de Servicios Web. Herramienta complementaria para el trabajo con los conceptos WSMO y lenguaje WSML. Sección 5.4.1.

WSMX: Ambiente de Ejecución para Servicios Web. Repositorio para la descripción semántica de los Servicios Web, se incluye en el conjunto de herramientas WSMT y es liberado como código libre independiente. Además sirve para la ejecución de Aplicaciones de Negocios donde los servicios Web mejorados son integrados para varias aplicaciones de negocios. [http://www.wsmx.org]. Sección 7.1.2.2.

XHTML: Acrónimo en inglés de e**X**tensible **H**yper**t**ext **M**arkup **L**anguage (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para páginas web. En su versión 1.0, XHTML es solamente la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo es avanzar en el proyecto del W3C de lograr una web semántica, donde la información, y la forma de presentarla estén claramente separadas.

XLANG: Lenguaje Extensible e**X**tensible **LAN**guage, creado por IBM y Microsoft junto con WSFL conforman el nucleo del sistema BPEL posteriormente BPEL4WS dirigido y auspiciado por IBM

XML: Lenguaje de Marcas Extensible. Metalenguaje de etiquetas desarrollado por el W3C para facilitar el marcado de información en WWW, además permite el intercambio de mensajes estructurados entre agentes software sobre varios protocolos.

BIBLIOGRAFÍA

[1] YOUSIF, Mazin(Colaborador). W3C Oficina Española. "Guía Breve de Servicios Web", Marzo 2008, [http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb], consultado 17 de Julio de 2008.

[2] BOOTH, David; HAAS, Hugo. W3C. "Web Services Architecture", Febrero 2004, [http://www.w3.org/TR/ws-arch/], consultado 2 de Agosto de 2008

[3] PEREZ, Juan Ignacio. "Web Services: XML-RPC, SOAP, sobre PHP, Perl, y otros conceptos", Agosto 2005, [http://www.programacion.com/tutorial/xmlrpcsoap/], consultado 28 de Julio de 2008.

[4] MITRA, Nilo; LAFON, Yves. W3C. "SOAP Version 1.2", Abril 2007, [http://www.w3.org/TR/2007/REC-soap12-part0-20070427/], consultada 16 de Julio de 2008.

[5] COWAN, John. "RESTful Web Services", 2005, [http://home.ccil.org/~cowan/restws.pdf*],* consultado 10 de Agosto de 2008.

[6] RUBY, Sam; RICHARDSON, Leonard. "RESTful Web Service". Mayo 2007

[7] NAVARRO, Rafael. "REST VS Web Services", Julio 2006, [http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf], consultado 8 de Septiembre de 2008.

[8] CHRISTENSEN, Erik; CRUBERA, Francisco; MEREDITH, Greg; WEERAWARANA, Sanjiva. W3C. "Web Service Description Language (WSDL) 1.1", Marzo 2001, [http://www.w3.org/TR/wsdl], consultado 20 de Julio de 2008.

[9] CLÉMENT, Luc (Editor); BRENDLE, Rainer; GUINAN, Dan. UDDI ORG (OASIS). "UDDI Version 2.03 Replication Specification", Julio 2002, [http://uddi.org/pubs/Replication-V2.03-Published-20020719.htm], consultado 17 de Agosto de 2008.

[10] GARCIA, Franciso. Universidad de Salamanca. "Web semántica y Ontologías", Noviembre 2005, [http://zarza.usal.es/~fgarcia/doctorado/iuce/WSemantica.pdf], consultada 19 de Julio de 2008.

[11] YOUSIF, Mazin(Colaborador). W3C Oficina Española. "Guía Breve de Web Semántica", Febrero 2008, [http://www.w3c.es/Divulgacion/Guiasbreves/WebSemantica], consultado 10 de Julio de 2008

[12] PEREZ, Damian (Colaborador). Maestros del web. "La Web Semántica y sus principales características", Junio 2007, [http://www.maestrosdelweb.com/editorial/web-semantica-y-sus-principales-caracteristicas/], consultado 7 de Octubre de 2008.

[13] ABIAN, Miguel Ángel. Web Semántica Hoy.**"**¿De dónde vienen los agentes software?**",** Marzo 2008, [http://www.wshoy.sidar.org/}, consultado 29 de Septiembre 2008.

[14] DIAZ, Elena. Universidad Carlos III (Madrid). "Web Semántica para recuperar información", Marzo 2006, [http://usuarios.lycos.es/websemanticas/], consultada 20 de Octubre de 2008.

[15] ALONSO, José Manuel. W3C Oficina Española. "La web semántica: Metadatos", Julio 2004, [http://www.w3c.es/Presentaciones/2005/0202-BPMS-JA/19.html], consultada 3 de Septiembre de 2008.

[16] SWICK, Ralph. W3C. "Metadata Activity Statement", Agosto 2002, [http://www.w3.org/Metadata/Activity.html], consultado 20 de Julio de 2008.

[17] WEB TALLER. "Metadatos en HTML", 2008, [http://www.webtaller.com/construccion/lenguajes/html/lecciones/metadatos-html.php], consultado 13 de Agosto de 2008.

[18] BECKETT, Dave; MCBRIDE, Brian. W3C. "RDF/XML Syntaxis Specification", Febrero 2004, [http://www.w3.org/TR/rdf-syntax-grammar/], consultada 15 de Agosto de 2008

[19] MICROFORMATOS ORG. "About Microformats", [http://microformats.org/about/], consultada 23 de Agosto de 2008.

[20] Microformats ORG. "Microformats, Code & Tools", Mayo 2008, [http://microformats.org/code-tools], consultada 12 Agosto de 2008.

[21] MCGUINNESS, Deborah; HARMELEN, Frank van. "Lenguaje de Ontologías Web (OWL) Vista General", Febrero 2004, [http://www.w3.org/2007/09/OWL-Overview-es.html], consultada en 25 de Agosto de 2008.

[22] PATEL-SCHNEIDER , Peter F.; HAYES, Patrick; HORROCKS, Ian. W3C. "OWL Web Ontology Language Semantics and Abstract Syntax", Febrero 2004, [http://www.w3.org/TR/owl-semantics], consultada 18 de Agosto de 2008.

[23] WIKIPEDIA, "iCalendar", http://es.wikipedia.org/wiki/ICalendar, consultada 16 de Septiembre de 2008.

[24] WorldLINGO. "iCALENDAR", 2009, [http://www.worldlingo.com/ma/enwiki/es/iCalendar], consultado 8 de Octubre de 2008

[25] ADIDA, BEN; BIRBECK, Mark. W3C. "RDFa Primer", Octubre 2008, [http://www.w3.org/TR/xhtml-rdfa-primer/], consultado 12 de Enero 2009.

[26] BIRBECK, Mark. "Introduction to RDFa", Junio 2009, [http://www.alistapart.com/articles/introduction-to-rdfa/], consultado 23 de Agosto de 2009

[27] NOWACK, Benjamin. "Posts tagged with: erdf", Mayo de 2009, [http://bnode.org/blog/tag/erdf], consultado 19 de Junio de 2009

[28] CONOLLY, Dan; MALONEY, Murray. W3C. "Gleaning Resource Description from Dialectc of Languages (GRDDL)", Septiembre 2007, [http://www.w3.org/TR/grddl/], consultada 18 de Septiembre de 2008.

[29] DALY, Janet; FORGUE; Marie-Clarie; HIRAKAWA, Yasuyuki. "W3C Completes Bridge Between HTML/Microformats and Semantic Web", Septiembre 2007, [http://www.w3.org/2007/07/grddl-pressrelease], consultada 28 de Agosto de 2008.

[30] CONOLLY, Dan. WEC. "Gleaning Resource Descriptions from Dialects of Languages (GRDDL)", Septiembre 2008, [http://www.w3.org/2004/01/rdxh/spec], consultada 8 de Noviembre de 2008.

[31] NOWACK, Benjamin. "ARC Embedded RDF (eRDF) Parser for PHP", Mayo 2006, [http://bnode.org/blog/2006/05/29/arc-embedded-rdf-erdf-parser-for-php], consultado 15 de Junio de 2009.

[32] ADUNA B.V.; SIRMA AI Ltda. "User Guide for SESAME", Octubre 2006, [http://www.openrdf.org/doc/sesame/users/index.html], consultada 15 de Diciembre de 2008.

[33] BROEKSTRA, Jeen; KAMPMAN, Arjohn; HARMELEN, Frank van. "Sesame: A generic Architecture for Storing and Querying RDF and RDF Schema", [http://www.dcc.uchile.cl/~churtado/TECNICAS/hcondori.pdf], consultado 8 de Enero de 2009.

[34] MCCARTHY, Philip. IBM. "Introductuin to Jena", Junio 2004, [http://www.ibm.com/developerworks/xml/library/j-jena/], consultado 18 de Octubre de 2008.

[35] ATELLA, Santiago; GENTA, Nicolas; GONZÁLEZ, Diego. "Tutorial de Jena", [http://www.slideshare.net/dragonx/tutorial-de-jena], consultado 23 de Enero de 2009.

[36] MARTIN , David (Editor); BURSTEIN, Mark; HOBBS, Jerry. W3C. "OWL-S: Semantic Markup for Web Services", Noviembre 2004,[http://www.w3.org/Submission/OWL-S/], consultado 23 de Octubre de 2008.

[37] AKKIRAJU, Rama; FARRELL, Joel. W3C. "Web Service Semantics - WSDL-S", Noviembre 2005, [http://www.w3.org/Submission/WSDL-S/], consultado 15 de Septiembre de 2008.

[39] KOURTESIS, Dimitrios; PARASKAKIS, Iraklis. "Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery", Mayo 2008, [http://www.eswc2008.org/final-pdfs-for-web-site/swsI-1.pdf], consultado 19 de Septiembre de 2008.

[38] FARRELL, Joel; LAUSEN, Holger. W3C. "Semantic Annotations for WSDL Working Group, Semantic Annotations for WSDL and XML Schema", Agosto 2007, [http://www.w3.org/TR/sawsdl/], consultado 28 de Agosto de 2008.

[41] GOMADAM, Karthik(Editor); RANABAHU, Ajith, SHETH, Amit. Kno.e.sis Services Science Lab (Wright State University), "SA-REST: Semantically enhancing REST services", Abril 2008, [http://knoesis.org/research/srl/standards/sa-rest/], consultado 19 de Octubre de 2008.

[40] DOUGLAS, Jonathan. "SA-REST: ADDING SEMANTICS TO REST-BASED WEB SERVICES", Noviembre 2005, [http://www.jonlathem.com/sarest/Thesis_final.pdf], consultado 9 de Septiembre 2008.

[41] Cardoso Jorge. "Semantic Web Services: Theory, Tools and Applications". 2007

[42] ZAREMBA, Michal; ZAREMBA, Maciej. "Semantically-enabled Service Oriented Architecture: Concepts, Technology and Application", en Journal of Service Oriented Computing and Applications, Agosto 2007

[43] Rudi Studer, Stephan Grimm. "Semantic Web Services: Concepts, Technologies and Applications" 2007

[44] Dieter Fensel, Holger Lausen. "Enabling Semantic Web Services: The Web Service Modeling Ontology" 2006

[45] WEERAWARANA, Sanjiva; CURBERA, Francisco. IBM. "Business Process with BPEL4WS", Agosto 2002, [http://www.ibm.com/developerworks/webservices/library/ws-bpelcol1/], consultado 23 de Noviembre de 2009.

[46] BATTLE, Steve; BERNSTEIN, Abraham; GRUNINGER, Michael. W3C. "Semantic Web Services Framework (SWSF) Overview", Septiembre 2005, [http://www.w3.org/Submission/SWSF/], consultado 7 de Febrero 2009.

[47] CABRAL, Liliana; DOMINGUE, John; GALIZIA, Stefania. "IRS III: A Broker for Semantic Web Services Aplications", Abril 2006, [http://projects.kmi.open.ac.uk/dip/resources/ISWC2006/ISWC06CabralFinal.pdf], consultado 18 de Febrero de 2009.

[48] PAUTZKE, Friedbert; MÖLLER Joachim. "INFRAWEBS System Design & Architecture", 2007, [http://www.infrawebs.eu/dissemination-system_design_and_architecture-.html?site=system_design_and_architecture], consultado 12 de Agosto de 2009.

[61] Lû, Huang. "Infrawebs Designer", Febreo 2008, [http://wdyw.files.wordpress.com/2008/03/infrawebs-designer.pdf], consultado 19 de Agosto 2009

[49] Mick Kerrigan, Michael Zaremba. "Implementing Semantic Web Services: The SESA Framework". 2008

[50] STOLLBERG, Michael; SHAFIQ, Omair; ANICIC, Darko. STI INNSBRUCK. "Semantically Enabled Service-Oriented Architectures (SESA)", Marzo 2006, [http://www.wsmo.org/TR/d17/sesa-tutorial/], consultado 12 de Marzo de 2009.

[51] LAUSEN, Holger; BRUIJN, Jos de; POLLERES, Axel; FENSEL, Dieter. "WSML - a Language Framework for Semantic Web Services ", Abril 2005, [http://www.w3.org/2004/12/rules-ws/paper/44/], consultado 15 de Marzo de 2009

[52] BRUIJN, Jos de (Editor); LAUSEN, Holger(Editor); KELLER, Uwe. W3C. "Web Service Modeling Language (WSML)", Junio 2005, [http://www.w3.org/Submission/WSML/], consultado 22 de Marzo de 2009.

[53] Jacek Kopecký, Tomas Vitvar. "Hrest & MicroWSMO CMS WG Deliverable.pdf " 10 de Marzo de 2009.

[54] STEINMETZ, Nathalie. STI Innsbruck. "Web Service Crawling And Annotation", Septiembre 2009,

[http://events.seekda.com/FIS2009/fis2009_t2_serviceOntologies.pdf], consultado Octubre 2009

[55] GIASSON, Frédérick (Editor). Google Code. "API BIBO Ontology", Noviembre 2008, [http://bibotools.googlecode.com/svn/bibo-ontology/trunk/doc/index.html], consultada

[56] GIASSON, Frédérick (Editor); D'ARCUS, Bruce. Google Code. "API BIBO Ontology", Noviembre 2008, [http://bibliontology.com/], consultada

[57] DCMI, Usage Board. "DCMI Metadata Terms", Enero 2008, [http://dublincore.org/documents/dcmi-terms/], consultado

[58] HILLMANN, Diane. "Using Dublin Core", Noviembre 2005, [http://dublincore.org/documents/usageguide/], consultado

[60] ENGEL, Robert A. van "Pushing the SOAP Envelope With Web Services for Scientific Computing", 2005, http://www.cs.fsu.edu/~engelen/icws03.pdf, consultada 20 de Julio 2008

[61] Lû, Huang. "Infrawebs Designer", Febreo 2008, [http://wdyw.files.wordpress.com/2008/03/infrawebs-designer.pdf], consultado 20 de Septiembre de 2009.

[62] Lû, Huang. "IRS-III Browser User Guide", [http://wdyw.files.wordpress.com/2008/02/irs-iii_user_guide.pdf], consultado 29 de Septiembre de 2009.

[63] SOM, Guillermo, "Web Service Paso a Paso", 2004, [http://www.elguille.info/NEt/universidad/WebServices1/WebServices1.htm], consultado

[64] APACHE GROUP. "jUDDI Developer Guide", Febrero 2009, [http://ws.apache.org/juddi/usersguide.html], consultado 14 de Mayo de 2009.

[65] HOLMES, Alex. "UDDI Browser 1.0-beta1 - User Guide", Septiembre 2005, [http://uddibrowser.org/docs/0.2/userguide.html], consultado 12 de Mayo de 2009.

[66] GONZÁLEZ, Benjamin(Colaborador). Desarrollo web. "WSDL para la documentación de Servicios Web", Julio 2004, http://www.desarrolloweb.com/articulos/1581.php, consultada 12 Julio de 2008.

[67] ZAREMBA, Maciej; "Web Service Execution Environment", Octubre 2008, [http://www.wsmx.org/], consultado 10 de Julio, 12 de Agosto de 2009.

[68] LAUSEN, Holger; BRUIJN, Jos de; POLLERES, Axel; FENSEL, Dieter. "WSML - a Language Framework for Semantic Web Services ", Abril 2005, [http://www.w3.org/2004/12/rules-ws/paper/44/], consultado

[69] BRUIJN, Jos de (Editor); LAUSEN, Holger(Editor); KELLER, Uwe. W3C. "Web Service Modeling Language (WSML)", Junio 2005, [http://www.w3.org/Submission/WSML/], consultado

[70] DIMITROV, Marin; MOMTCHEV, Vassill; SIMOV, Alex; OGNYANOFF, Damian. WSMO ORG, "WSMO4J Programmers Guide", Noviembre 2006, [http://wsmo4j.sourceforge.net/doc/wsmo4j-prog-

guide.pdf], consultado 23 de Agosto de 2009.

[71] STEINMETZ, Nathalie. WSMX ORG. "WSMO4J", Mayo 2008, [http://wiki.wsmx.org/index.php?title=DERI_WSMO4J_Working_Group], consultado 12 de Agosto de 2009.

[72] BISHOP, Barry; FISCHER, Florian; KELLER, Uwe. "WSML2Reasoner", Noviembre 2008, [http://tools.sti-innsbruck.at/wsml2reasoner/], consultado 18 de Agosto de 2009.

[73] BISHOP, Barry; FISCHER, Florian; KELLER, Uwe "IRIS - Integrated Rule Inference System – API and User Guide", Abril 2008, [http://www.iris-reasoner.org/pages/user_guide.pdf], consultado 3 de Septiembre de 2009.

ANEXOS

ANEXO 1

ONTOLOGÍA DUBLIN

La ontología dublin implementada en la presente Tesis toma como núcleo la ontología RDF Dublin Core, adicionando elementos de la ontología BIBO desarrollada por Google.

Primeramente se expondrá la declaración de conceptos, subconceptos, atributos y axiomas, para mostrar por separada la declaración de instancias.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-rule"
namespace { _"http://www.tesiswsmo.com/ontologias/dc#"
,
     dc _"http://purl.org/dc/elements/1.1/" }

ontology dublin

/***********************************/
/****** DECLARACION DE AXIOMAS ******/
/***********************************/

axiom Agent_ResourceAuthor
     definedBy ?x[name hasValue ?y] memberOf Agent
             and ?z[authorInstance hasValue ?x] memberOf Resource
       implies ?z[author hasValue ?y] memberOf Resource.

axiom Agent_ResourceDirector
     definedBy ?x[name hasValue ?y] memberOf Agent
             and ?z[directorInstance hasValue ?x] memberOf Resource
       implies
       ?z[author hasValue ?y, director hasValue ?y] memberOf Resource.

axiom Resource_AuthorAgent
     definedBy ?x[authorInstance hasValue ?y] memberOf Resource
       implies ?y[works hasValue ?x] memberOf Agent.

axiom Resource_DirectorAgent
     definedBy ?x[directorInstance hasValue ?y] memberOf Resource
       implies ?y[works hasValue ?x] memberOf Agent.

axiom Resource_rightsHolder
     definedBy  ?x[member hasValue ?y] memberOf Agent
             and ?z[authorInstance hasValue ?x] memberOf Resource
             and ?y memberOf Organization
       implies ?z[rightsHolder hasValue ?y] memberOf Resource.

axiom Resource_relations
     definedBy ?x[subject hasValue ?y] memberOf Resource
             and ?z[subject hasValue ?y] memberOf Resource
       implies ?z[related hasValue ?x] memberOf Resource
             and ?x[related hasValue ?z] memberOf Resource.

axiom Research_Theory
     definedBy ?x[level hasValue Research] memberOf Resource
       implies ?x[level hasValue Theory] memberOf Resource.
```

```
axiom Tutorial_Developmetn
      definedBy ?x[level hasValue Tutorial] memberOf Resource
        implies ?x[level hasValue Development] memberOf Resource.

axiom Phd_Implies
      definedBy ?x[degree hasValue Phd] memberOf Agent
        implies   ?x[degree  hasValue  {Master,  Engineering,  Graduate,  Student}]
memberOf Agent.

axiom Master_Implies
      definedBy ?x[degree hasValue Master] memberOf Agent
        implies  ?x[degree  hasValue  {Engineering,  Graduate,  Student}]  memberOf
Agent.

axiom Engineering_Implies
      definedBy ?x[degree hasValue Engineering] memberOf Agent
        implies ?x[degree hasValue {Graduate, Student}] memberOf Agent.

axiom Graduate_Implies
      definedBy ?x[degree hasValue Graduate] memberOf Agent
        implies ?x[degree hasValue Student] memberOf Agent.

axiom Interactive
      definedBy
            ?x memberOf Resource and ?x[needUser hasValue _boolean("true")]
        implies ?x memberOf InteractiveResource.

/*************************************/
/****** DECLARACION DE CONCEPTOS ******/
/*************************************/

concept DataType

concept Collection subConceptOf DataType

concept Image subConceptOf DataType

concept Text subConceptOf DataType

concept PhysicalResource subConceptOf Resource

concept DigitalResource subConceptOf Resource

concept Identifier

concept ISBN subConceptOf Identifier
      ISBN ofType _string
      valid ofType _boolean

concept URI subConceptOf Identifier
      URI ofType _string

concept AgentClass

concept Agent subConceptOf AgentClass
      name ofType _string
      member ofType Organization
      works ofType Resource
      degree ofType Degrees
      phone ofType _string
      email ofType _string
      homePage ofType _string
      paper ofType Resource

concept Organization subConceptOf AgentClass
      description ofType _string
      location ofType _string
```

```
concept Resource
     title ofType _string
     authorInstance ofType Agent
     author ofType _string
     directorInstance ofType Agent
     director ofType _string
     subject ofType _string
     rightsHolder ofType Organization
     identifier ofType Identifier
     type ofType DataType
     description ofType _string
     date ofType _date
     location ofType _string
     version ofType _string
     related ofType Resource
     license ofType _string
     language ofType _string
     format ofType _string
     physicalFormat ofType PhysicalFormat
     level ofType Publications
     needUser ofType _boolean

concept PhysicalFormat subConceptOf DataType

concept Level

concept Degrees subConceptOf Level

concept Publications subConceptOf Level

concept InteractiveResource subConceptOf { DigitalResource, PhysicalResource}

/****************************************/
/****** DECLARACION DE INSTACNCIAS ******/
/****************************************/

instance Rudi_Studer memberOf Agent
     degree hasValue Engineering
     name hasValue "Rudi Studer"

instance Stephan_Grimm memberOf Agent
     degree hasValue Engineering
     name hasValue "Stephan Grimm"

instance Andreas_Abecker memberOf Agent
     degree hasValue Engineering
     name hasValue "Andreas Abecker"

instance Charles_Petrie memberOf Agent
     degree hasValue Engineering
     name hasValue "Charles Petrie"

instance Tiziana_Margaria memberOf Agent
     degree hasValue Engineering
     name hasValue "Tiziana Margaria"

instance Holger_Lausen memberOf Agent
     degree hasValue Engineering
     name hasValue "Holger Lausen"

instance Michael_Zaremba memberOf Agent
     degree hasValue Master
     name hasValue "Michael Zaremba"

instance Dieter_Fensel memberOf Agent
     degree hasValue Master
     name hasValue "Dieter Fensel"
```

```
instance Mick_Kerrigan memberOf Agent
     degree hasValue Master
     member hasValue STI
     name hasValue "Mick Kerrigan"

instance Liyang_Yu memberOf Agent
     degree hasValue Engineering
     name hasValue "Liyang Yu"

instance Martin_Hepp memberOf Agent
     degree hasValue Engineering
     name hasValue "Martin Hepp"

instance Pieter_De_Leenheer memberOf Agent
     degree hasValue Master
     name hasValue "Pieter De Leenheer"

instance Aldo_de_Moor memberOf Agent
     degree hasValue Engineering
     name hasValue "Aldo de Moor"

instance York_Sure memberOf Agent
     degree hasValue Master
     name hasValue "York Sure"

instance Jos_de_Bruijn memberOf Agent
     degree hasValue Phd
     name hasValue "Jos de Bruijn"

instance Uwe_Keller memberOf Agent
     degree hasValue Phd
     name hasValue "Uwe Keller"

instance James_Scicluna memberOf Agent
     degree hasValue Master
     name hasValue "James Scicluna"

instance Axel_Polleres memberOf Agent
     degree hasValue Phd
     name hasValue "Axel Polleres"

instance Michel_Stollberg memberOf Agent
     degree hasValue Phd
     name hasValue "Michel Stollberg"

instance Dumitru_Roman memberOf Agent
     degree hasValue Master
     name hasValue "Dumitru Roman"

instance John_Domingue memberOf Agent
     degree hasValue Phd
     name hasValue "John Domingue"

instance Software memberOf DataType

instance Dataset memberOf DataType

instance InteractiveResource memberOf DataType

instance Sound memberOf DataType

instance MovingImage memberOf Image

instance Book memberOf Text

instance WebPage memberOf Text

instance WebSite memberOf Collection
```

```
instance Encyclopedia memberOf Collection

instance StaticImage memberOf Image

instance Carlos_Herrera memberOf Agent
     member hasValue UTPL
     degree hasValue Graduate
     name hasValue "Carlos Herrera"

instance Tesis1 memberOf PhysicalResource
     authorInstance hasValue Carlos_Herrera
     title hasValue "Estudio de Servicios Web Semánticos"
     subject hasValue {"Semantic Web", "Web Service", "Semantic Web Service" }
     type hasValue Book
     directorInstance hasValue {Jorge_Lopez, Nelson_Piedra }
     location hasValue "Loja, Ecuador"
     date hasValue _date(2009,7,2,9,0)
     level hasValue Research
     needUser hasValue _boolean("false")

instance Jorge_Lopez memberOf Agent
     member hasValue UTPL
     degree hasValue Phd
     name hasValue "Jorge López"

instance Event memberOf DataType

instance UTPL memberOf Organization
     description hasValue "Universidad Técnica Particular de Loja"
     location hasValue "Loja, Ecuador"

instance Nelson_Piedra memberOf Agent
     member hasValue UTPL
     degree hasValue Phd
     name hasValue "Nelson Piedra"

instance Event_1 memberOf InteractiveResource
     language hasValue "ENG"
     title hasValue "Future Internet Symposium 2009"
     authorInstance hasValue STI
     subject hasValue {"Internet", "Web Service", "Semantic Web" }
     date hasValue _date(2009,11,3,8,0)
     type hasValue Event
     location hasValue "Berlin, Germany"
     needUser hasValue _boolean("true")

instance STI memberOf Organization
     description hasValue "STI Innsbruck"
     location hasValue "Innsbruck, Austria"

instance ISBN_1 memberOf ISBN
     ISBN hasValue "978-3-540-70893-3"
     valid hasValue _boolean("true")

instance ISBN_2 memberOf ISBN
     ISBN hasValue "978-0-387-72495-9"
     valid hasValue _boolean("true")

instance ISBN_3 memberOf ISBN
     ISBN hasValue "978-3-540-77019-0"
     valid hasValue _boolean("true")

instance ISBN_4 memberOf ISBN
     ISBN hasValue "978-1-58488-933-5"
     valid hasValue _boolean("true")

instance ISBN_5 memberOf ISBN
```

```
     ISBN hasValue "978-0-387-69899-1"
     valid hasValue _boolean("true")

instance ISBN_6 memberOf ISBN
     ISBN hasValue "978-3-540-68169-4"
     valid hasValue _boolean("true")

instance ISBN_7 memberOf ISBN
     ISBN hasValue "978-3-540-34519-0"
     valid hasValue _boolean("true")

instance Libro_1 memberOf PhysicalResource
     authorInstance hasValue {Rudi_Studer, Stephan_Grimm, Andreas_Abecker }
     title hasValue "Semantic Web Services: Concepts, Technologies, and
Applications"
     language hasValue "ENG"
     date hasValue _date(2007,5,5,0,0)
     identifier hasValue ISBN_1
     subject hasValue "Semantic Web Service"
     type hasValue Book
     level hasValue Theory
     needUser hasValue _boolean("false")

instance Libro_2 memberOf PhysicalResource
     authorInstance hasValue {Charles_Petrie, Tiziana_Margaria, Holger_Lausen,
Michael_Zaremba }
     language hasValue "ENG"
     date hasValue _date(2008,5,5,0,0)
     type hasValue Book
     subject hasValue {"Semantic Web Service", "Semantic Web" }
     identifier hasValue ISBN_2
     title hasValue "Semantic Web Services Challenge: Results from the First
Year (Semantic Web and Beyond)"
     level hasValue Theory
     needUser hasValue _boolean("false")

instance Libro_3 memberOf PhysicalResource
     authorInstance hasValue {Dieter_Fensel, Mick_Kerrigan, Michael_Zaremba }
     language hasValue "ENG"
     date hasValue _date(2008,5,5,0,0)
     type hasValue Book
     subject hasValue {"Semantic Web Service", "SESA" }
     identifier hasValue ISBN_3
     title hasValue "Implementing Semantic Web Services: The SESA Framework"
     level hasValue Development
     needUser hasValue _boolean("false")

instance Libro_4 memberOf PhysicalResource
     authorInstance hasValue Liyang_Yu
     language hasValue "ENG"
     date hasValue _date(2007,5,5,0,0)
     type hasValue Book
     identifier hasValue ISBN_4
     subject hasValue {"Semantic Web", "Semantic Web Service" }
     title hasValue "Introduction to the Semantic Web and Semantic Web
Services"
     level hasValue Theory
     needUser hasValue _boolean("false")

instance Libro_5 memberOf PhysicalResource
     authorInstance hasValue {Martin_Hepp, Pieter_De_Leenheer, Aldo_de_Moor,
York_Sure }
     language hasValue "ENG"
     date hasValue _date(2007,5,5,0,0)
     type hasValue Book
     identifier hasValue ISBN_5
     subject hasValue {"Ontologies", "Semantic Web", "Semantic Web Service",
"Bussiness Application" }
```

```
     title hasValue "Ontology Management: Semantic Web, Semantic Web Services,
and Business Applications (Semantic Web and Beyond)"
     level hasValue Theory
     needUser hasValue _boolean("false")

instance Libro_6 memberOf PhysicalResource
     authorInstance  hasValue  {Jos_de_Bruijn,  Dieter_Fensel,  Mick_Kerrigan,
Uwe_Keller, Holger_Lausen, James_Scicluna }
     language hasValue "ENG"
     date hasValue _date(2008,5,5,0,0)
     type hasValue Book
     identifier hasValue ISBN_6
     subject hasValue {"Semantic Web Service", "Web Service", "WSMT" }
     title hasValue "Modeling Semantic Web Services: The Web Service Modeling
Language"
     level hasValue Tutorial
     needUser hasValue _boolean("false")

instance Libro_7 memberOf PhysicalResource
     authorInstance  hasValue  {Dieter_Fensel,  Holger_Lausen,  Axel_Polleres,
Jos_de_Bruijn, Michel_Stollberg, Dumitru_Roman, John_Domingue }
     language hasValue "ENG"
     date hasValue _date(2006,5,5,0,0)
     type hasValue Book
     identifier hasValue ISBN_7
     subject hasValue {"Semantic Web Service", "WSMO" }
     title hasValue "Enabling Semantic Web Services: The Web Service Modeling
Ontology"
     level hasValue Development
     needUser hasValue _boolean("false")

instance Paper memberOf Text

instance Movie_1 memberOf PhysicalResource
     title hasValue "The Godfather"
     authorInstance hasValue Al_Pacino
     type hasValue MovingImage
     physicalFormat hasValue DVD

instance Al_Pacino memberOf Agent
     name hasValue "Al Pacino"

instance DVD memberOf PhysicalFormat

instance CD memberOf PhysicalFormat

instance VHS memberOf PhysicalFormat

instance Julia_Roberts memberOf Agent
     name hasValue "Julia Roberts"

instance Movie_2 memberOf PhysicalResource
     title hasValue "The Monalisa smile"
     authorInstance hasValue Julia_Roberts
     physicalFormat hasValue VHS
     type hasValue MovingImage

instance Graduate memberOf Degrees

instance Engineering memberOf Degrees

instance Master memberOf Degrees

instance Phd memberOf Degrees

instance Student memberOf Degrees

instance Research memberOf Publications
```

```
instance Theory memberOf Publications

instance Development memberOf Publications

instance Tutorial memberOf Publications

instance Video_1 memberOf DigitalResource
     authorInstance hasValue Mick_Kerrigan
     level hasValue Tutorial
     physicalFormat hasValue DVD
     date hasValue _date(2008,5,5,0,0)
     title hasValue "Web Service Modelling Toolkit (WSMT)"
     subject hasValue {"WSMO", "WSMT", "Semantic Web Service" }
     description hasValue "Tutorial WSMT"
     type hasValue MovingImage
     language hasValue "ENG"

instance URL_1 memberOf URI
     URI                                                                    hasValue
"http://rease.semanticweb.org/ubp/PUSH/search@srchDetailsLR%3Bjsessionid=C3421"

instance Web_1 memberOf DigitalResource

instance Animacion_1 memberOf InteractiveResource
     needUser hasValue _boolean("true")
```

ANEXO 2

SERVICIOS WEB PUBLICADOS

AMAZONBOX

Proveedor: Amazon

URL: http://www.xmlme.com/WSAmazonBox.asmx

WSDL: http://www.xmlme.com/WSAmazonBox.asmx?wsdl

ISBN INFORMATION

Proveedor: WebServicex.net

URL: http://www.webservicex.com/isbn.asmx

WSDL: http://www.webservicex.com/isbn.asmx?WSDL

BOOKSTORESERVICE

Proveedor: jku.at

URL: http://dotnet.jku.at/csbook/solutions/19/BookStoreService.asmx

WSDL: http://dotnet.jku.at/csbook/solutions/19/BookStoreService.asmx?WSDL

STORESERVICE

Proveedor: jku.at

URL: http://soatest.parasoft.com/services/#Store-04

WSDL: http://soatest.parasoft.com/store-04.wsdl

LOOKUPBOOKDATA

Proveedor: pickabook.co.uk

URL: http://services.pickabook.co.uk/service.asmx

WSDL: http://services.pickabook.co.uk/service.asmx?WSDL

EVENTS

Proveedor: aspalliance.com

URL: http://aspalliance.com/webservices/events.asmx

WSDL: http://aspalliance.com/webservices/events.asmx?WSDL

VIDEOSERVICE

Proveedor: vertigo.com

URL: http://pp2g.tv/home/videoservice.asmx

WSDL: http://pp2g.tv/home/videoservice.asmx?WSDL

VIDEO_SERVICE

Proveedor: ecocoma.com

URL: http://service.ecocoma.com/video/video.asmx

WSDL: http://service.ecocoma.com/video/video.asmx?WSDL

YOUTUBEDOWNLOAD_SERVICE

Proveedor: ecubicle

URL: http://www.ecubicle.net/youtubedownloader.asmx

WSDL: http://www.ecubicle.net/youtubedownloader.asmx?WSDL

ANEXO 3

PREREQUISITOS

La presente Tesis utiliza como única plataforma de desarrollo el lenguaje Java debido a los avances y herramientas propuestas por los diferentes grupos de investigación[89] dedicados a los Servicios Web Semánticos y tecnologías complementarias, la mayor parte de estas herramientas y paquetes son liberados como código abierto para ser utilizados con el IDE[90] Eclipse.

JAVA

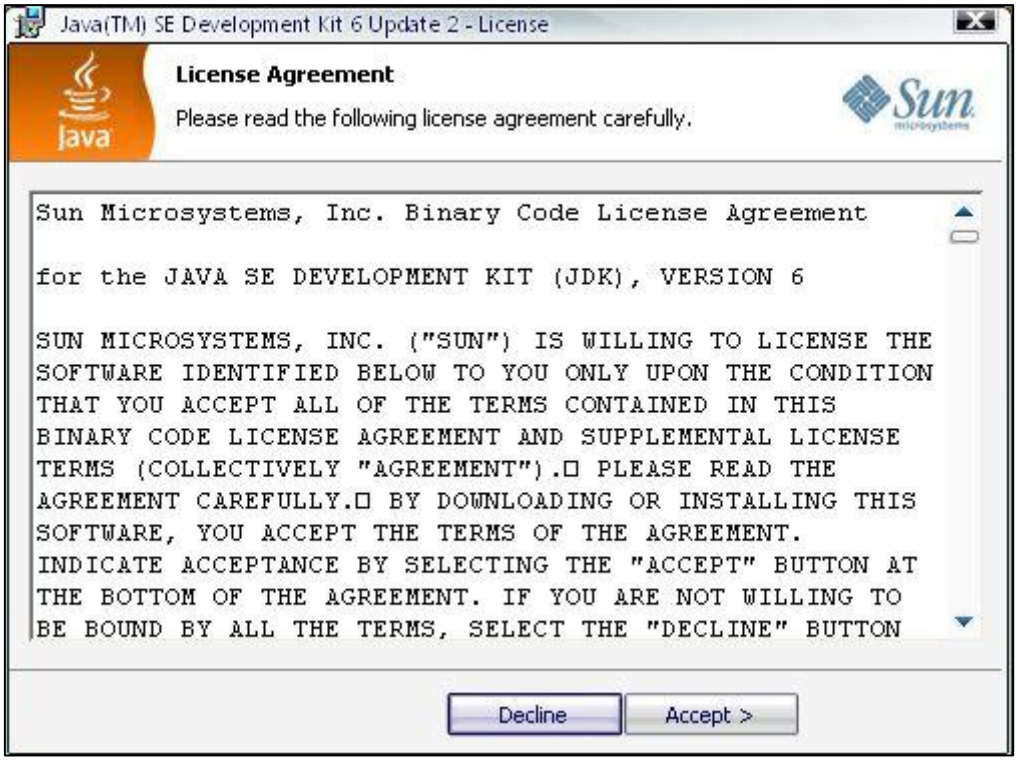
Durante el desarrollo de este proyecto se ha utilizado el JDK 6 Update 14 tomado de http://java.sun.com/javase/downloads/index.jps, una vez realizada la descarga del archivo empaquetado (70 MB aproximadamente) se procede a descomprimirlo, obteniendo un archivo con nombre jdk-6u2-windows-i586-p.exe, el cual se procede a ejecutarlo y seguir los pasos solicitados en el Wizard.

Paso 1. Obtención de información requerida por el instalador.

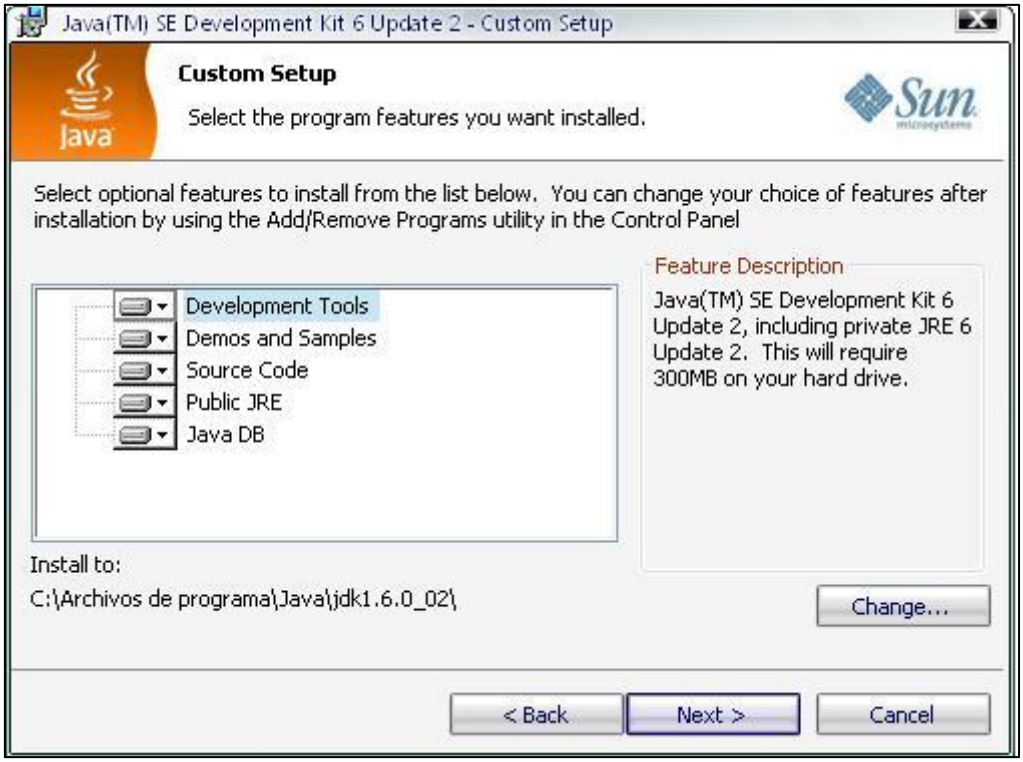
Paso 2. Aceptación de la licencia

89 **D**igital **E**nterprise **R**esearch **I**nstitute [http://www.deri.ie/]
The Semantic Technology Institute Innsbruck [http://www.sti-innsbruck.at/]
Service Execution Environment [http://www.wsmx.org/]

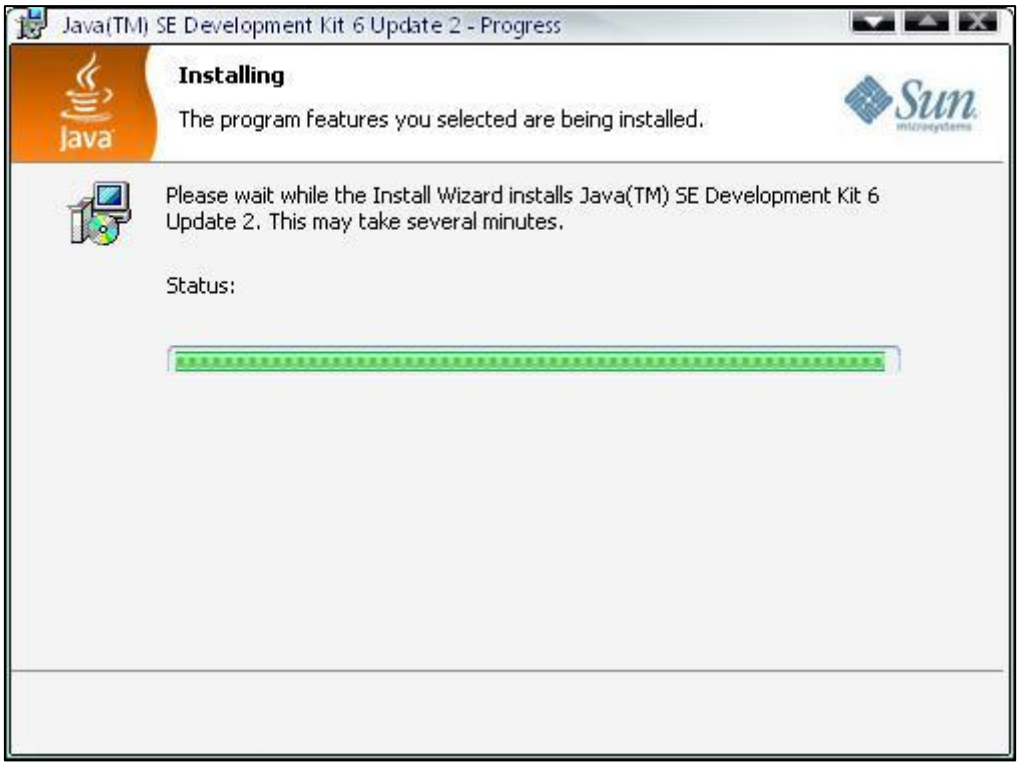
90 **I**ntegrated **D**evelopment **E**nvironment (Entorno de desarrollo integrado)



Paso 3. Personalización de la herramienta

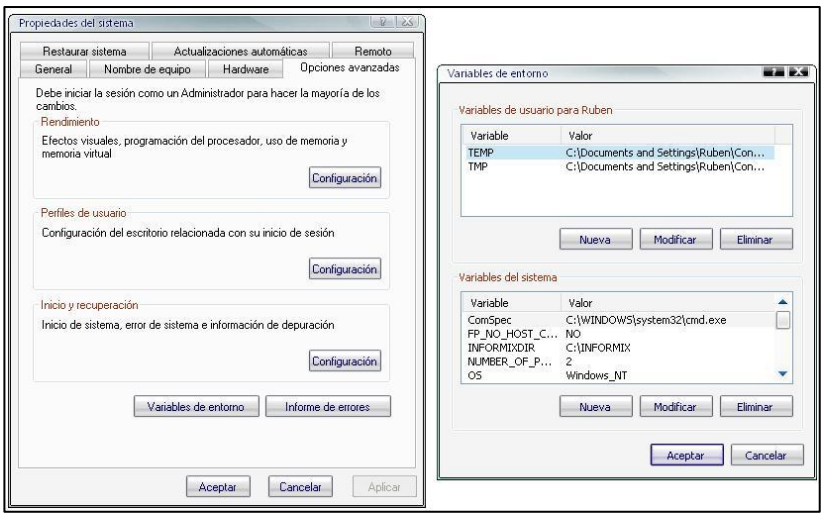


Paso 4. Proceso de instalación



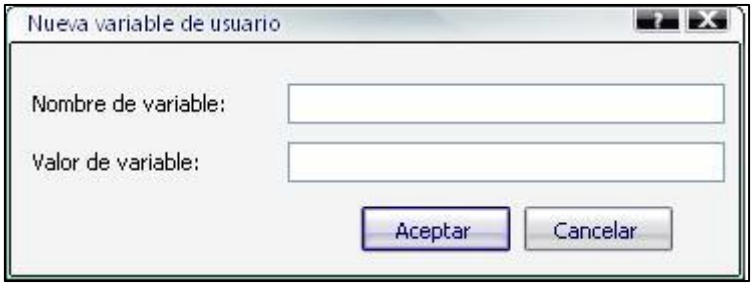
Paso 5. Variables de entorno

Una vez finalizado satisfactoriamente el proceso de instalación es necesaria la configuración adecuada de las variables de entorno `JAVA_HOME` y `PATH`, este paso se realiza ubicándonos en las **Propiedades del sistema** (Clic derecho sobre Mi PC), nos ubicamos en la pestaña **Opciones Avanzadas** y luego pulsamos en el botón **Variables de entorno** mostrado en la siguiente figura.



Ahora realizaremos la configuración de las variables `JAVA_HOME` y `PATH`, en caso que las variables mencionadas existan será necesario re direccionar hacia la instalación Java versión 6.

En la opción de **Variables de usuario para xxx** pulsamos sobre el botón **Nueva** para crear la variable de entorno `JAVA_HOME`, luego se realizará el mismo proceso para crear la variable `PATH`, debemos completar la información de la ventana mostrada en la figura con los valores expuestos posteriormente



Para la variable JAVA_HOME los valores son:

Nombre de variables: JAVA_HOME
Valor de la variable: C:\Archivos de programa\Java\jdk1.6.0_02

Con la instalación de JAVA se crean las carpetas **jdk1.6.0_02** y **jre1.6.0_02** (dependiendo de la versión instalada) bajo C:\Archivos de programa\Java\, para la configuración de la variable JAVA_HOME debemos apuntar la variable hacia el JDK, no hacia el JRE.

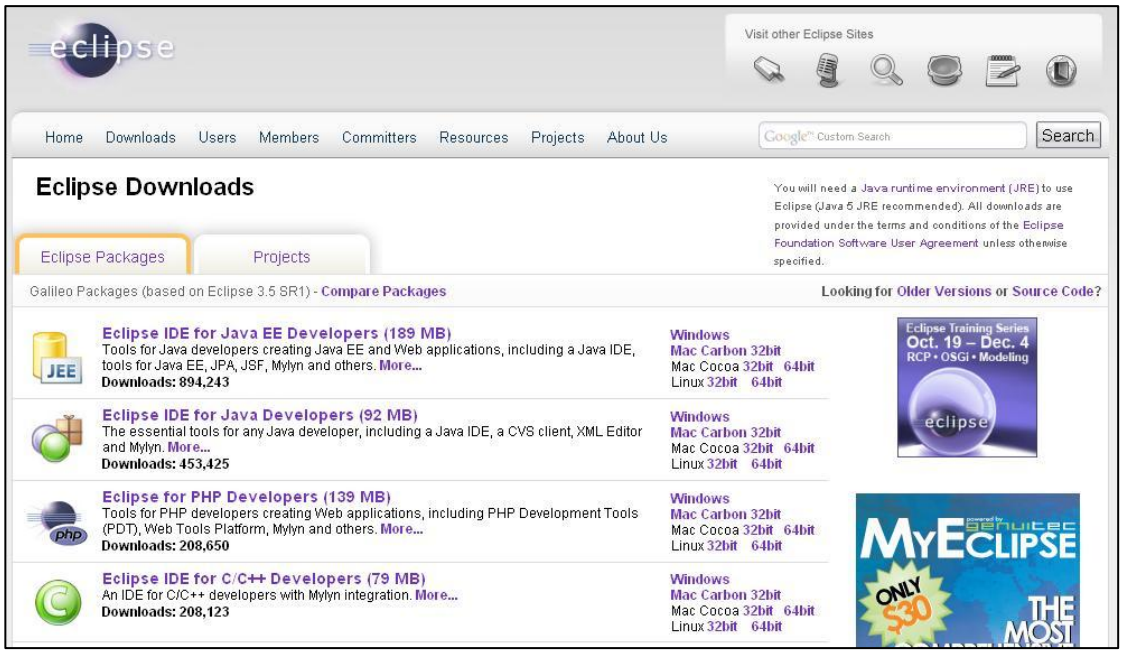
Para la variable PATH los valores son:

Nombre de variables: PATH
Valor de la variable: %JAVA_HOME%\bin

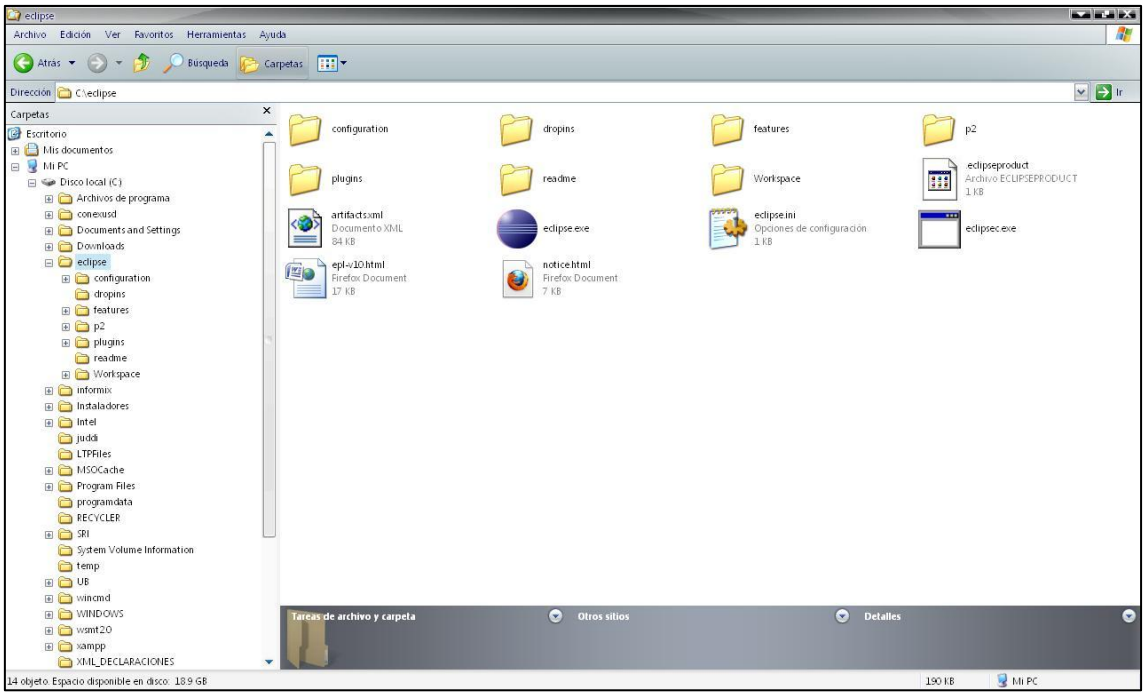
En caso de existir la variable `PATH` configurada, los valores antes mencionados se colocarán el inicio del valor de la variable seguido por un **punto y coma** ";" para no afectar a los demás valores en la variable.

ECLIPSE

En primer lugar realizaremos la descarga del IDE Eclipse desde el sitio oficial http://www.eclipse.org/downloads/, la siguiente figura muestra las diferentes opciones disponibles, pues Eclipse es una herramienta muy versátil que provee plug-ins para diferentes lenguajes, para el presente caso se utilizará la versión para desarrolladores Java (Eclipse IDE for Java Developers – 92 MB-).



El archivo resultante de la descarga “eclipse-java-europa-win32.zip” contiene todas las librerías y plug-ins (según el paquete de instalación que hayamos seleccionado) necesarios para ejecutar el IDE Eclipse, únicamente es necesario descomprimirlo colocándolo en cualquier ubicación en el disco local, preferentemente C:\Eclipse con lo cual obtendremos los archivos como se muestra en la siguiente figura.



Una vez extraídos los archivos tenemos el ejecutable **eclipse.exe** que inicia el IDE Eclipse que se ha tratado en las diferentes secciones de esta Tesis, antes de iniciar el entorno de trabajo siempre se realizará la carga de plug-ins y librerías.

ANEXO 4

HERRAMIENTA WSMT

La principal herramienta que se ha utilizado para la creación de casos de estudio y comprobación de las diferentes teorías y conceptos investigados es la herramienta WSMT (Web Service Modelling Toolkit), la misma que se ofrece en **SouceForge.net** como herramienta de código abierto.

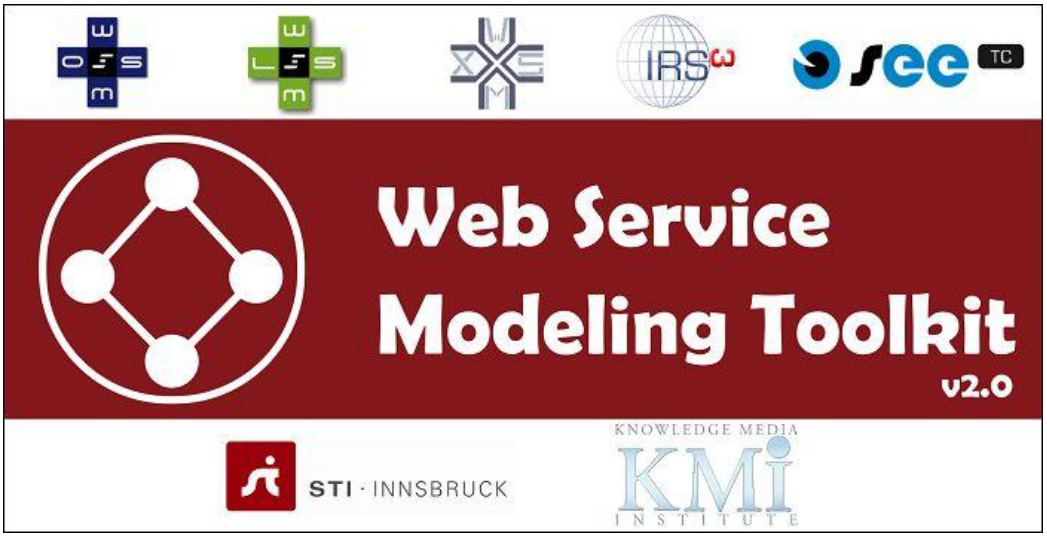
Durante el desarrollo de esta Investigación se ha tratado con dos versiones WSMT 1.4.1 y la versión 2.0 liberada el 26 de Marzo de 2009

Mediante la dirección http://sourceforge.net/projects/wsmt/ podemos acceder a las diferentes versiones de descarga, los diferentes casos de estudio han sido modificados para adaptarse a la nueva versión 2.0 por lo cual se recomienda hacer uso de la mencionada versión.

En la siguiente figura se muestra el enlace **View all files** mediante el cual se puede acceder a versiones anteriores y versiones específicas para los diferentes Sistemas Operativos.

La herramienta WSMT es elaborada como un producto Eclipse, por tanto, tiene grandes similitudes con este entorno y de la misma manera no requiere de una instalación, sino únicamente la extracción del paquete y una ubicación en el disco local, preferentemente en

C:\WSMT donde se ubicarán todos los plug-ins y paquetes necesarios, así como su ejecutable **Web Service Modelling Toolkit.exe.**



ANEXO 5

REPOSITORIO jUDDI

En el Capítulo 6 se ha utilizado jUDDI como repositorio local para la descripción sintáctica de Servicios Web, esta herramienta trabaja con Apache Tomcat para ofrecer una consola de administración a través del puerto http://localhost:8080/juddi/.
Adicionalmente, jUDDI permite almacenar la descripción sintáctica de los Servicios Web en una base de datos relacional, en este caso se ha utilizado MySQL; es necesario realizar varias configuraciones para un correcto funcionamiento de estas herramientas, a continuación se detalla a los sitios de descarga, instalación y configuración.

Apache jUDDI

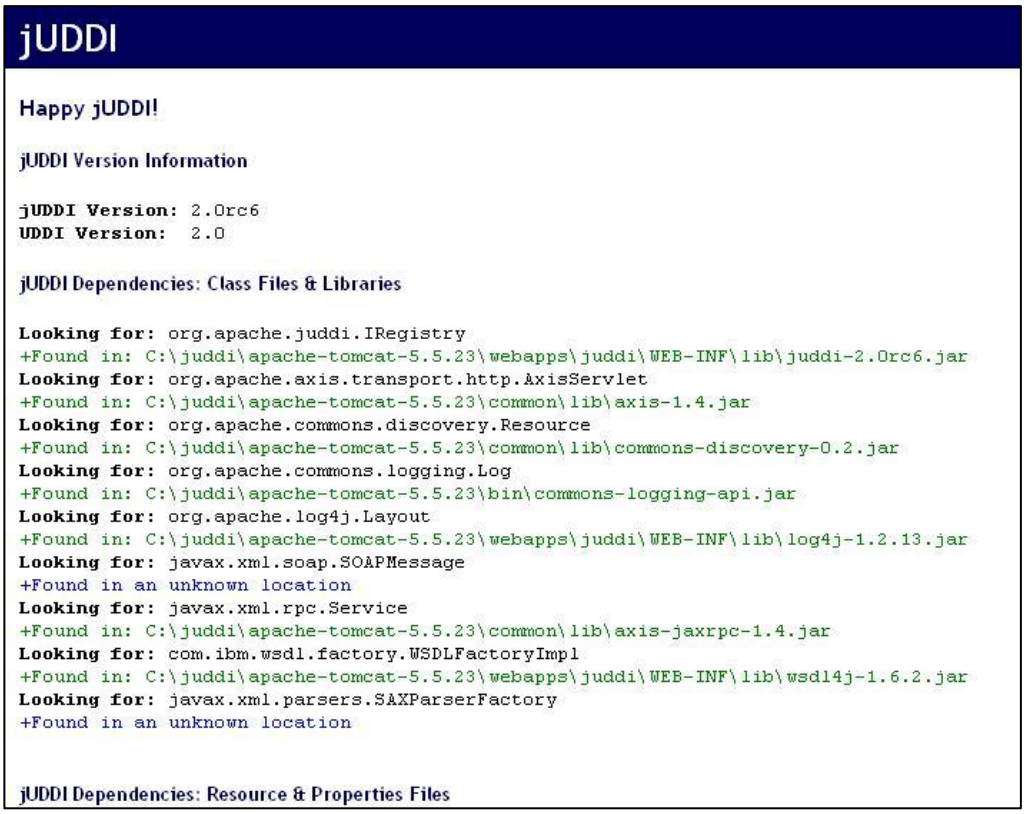
Para iniciar con el proceso de instalación de esta herramienta debemos acceder a http://ws.apache.org/juddi/releases.html de donde se escogerá la distribución ya integrada con Apache Tomcat 5.5:juddi-tomcat-2.0rc6.zip

Procedemos a descomprimir el fichero juddi-tomcat-2.0rc6.zip que hemos descargado, en un directorio a nuestra elección, preferentemente C:\juddi. Al ser Apache Tomcat el servidor sobre el que se ejecuta jUDDI es necesario realizar la configuración de la variable de sistema `CATALINA_HOME` de la misma manera como se ha detallado en el Anexo 2.

En el caso particular de este proyecto la variable `CATALINA_HOME` poseerá los valores:

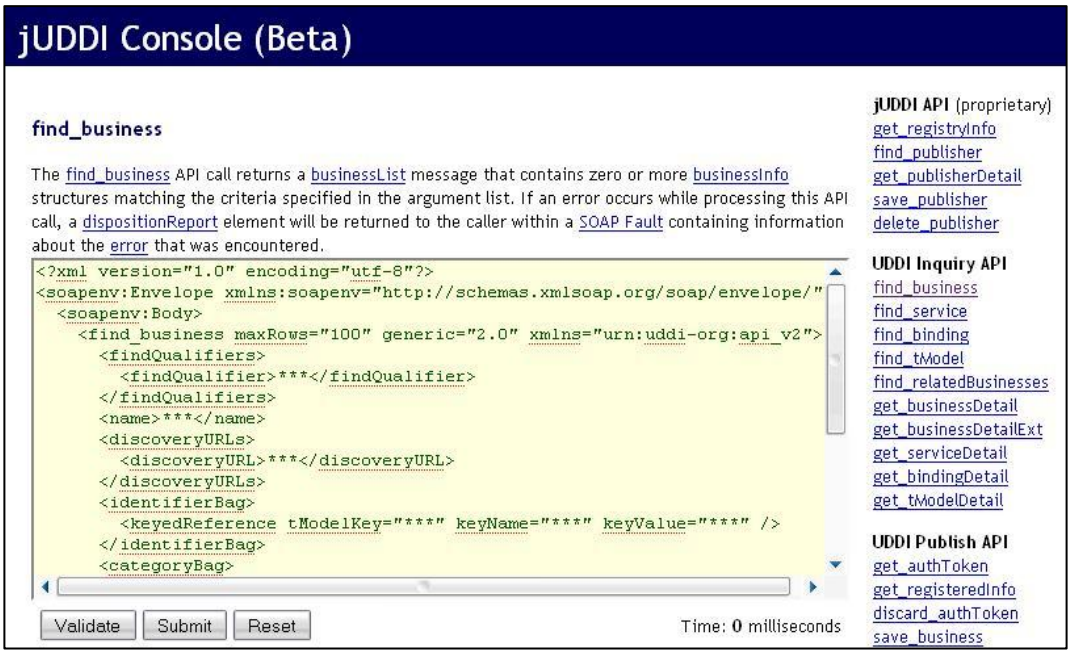
Nombre de variable: CATALINA_HOME
Valor de la variable: C:\juddi\apache-tomcat-5.5.23

Caso seguido se procede a levantar el servicio ejecutando **startup.bat** que se encuentra ubicado en C:\juddi\apache-tomcat-5.5.23\bin para finalmente colocar la dirección URL http://localhost:8080/juddi/ en nuestro navegador, el resultado se muestra en la siguiente figura.



Esto indica la correcta instalación del repositorio jUDDI, no obstante, aún es necesaria la instalación y configuración de MySQL donde se almacenará la información descriptiva de los Servicios Web y nos permitirá utilizar los métodos de jUDDI Console.

Luego de la instalación y configuraciones relacionadas al MySQL podemos utilizar los diferentes métodos de jUDDI Console, estos métodos trabajan mediante el intercambio de mensajes SOAP con el servidor jUDDI y en este caso, insertan la información en la base de datos MySQL configurada para el caso. La siguiente figura muestra el trabajo con los métodos jUDDI mediante la consola web.

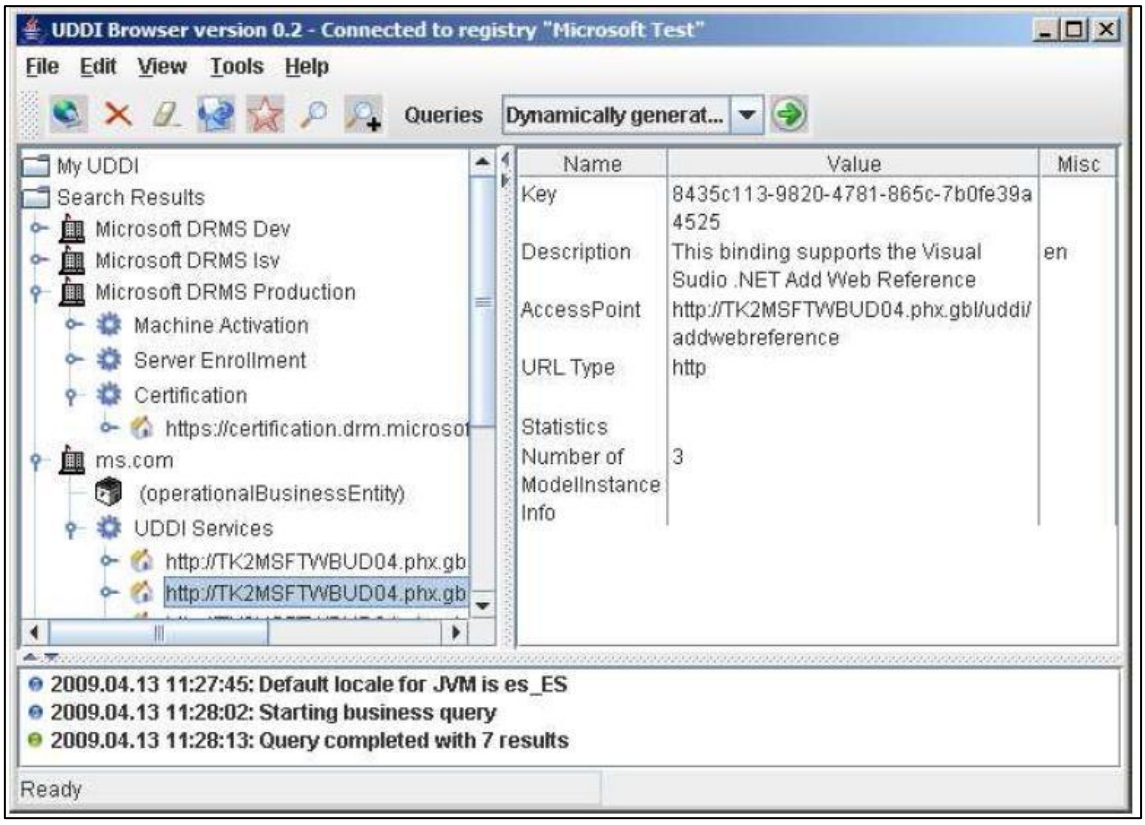


El manejo de mensajes SOAP para el uso de los métodos de jUDDI es un trabajo engorroso que dificulta las tareas de publicación y descripción de los Servicios, por tanto, en la presente investigación se ha utilizado UDDI Browser que es una aplicación cliente de jUDDI desarrollada en Java.

UDDI Browser

UDDI Browser es un desarrollo Open Source que soporta UDDI 2.0, permite trabajar con los registros públicos de IBM, Microsoft, SAP, HP, etc. (algunos de los cuales ya no están disponibles) y también con nuestro repositorio jUDDI local, para lo cual requerimos añadir un nuevo registro jUDDI.

Para descargar esta herramienta accedemos a http://uddibrowser.org del cual obtendremos el archivo ub-0.2-bin.zip, procedemos a descomprimirlo en cualquier ubicación en nuestro disco local, preferentemente en C:\UB, e iniciamos la aplicación con ub-0.2-bin\bin\ub.bat.



MySQL

La base de datos MySQL es también una herramienta Open Source, la misma que podemos encontrar en http://dev.mysql.com/downloads/mysql/5.1.html#win32 para nuestro caso, utilizaremos la versión 5.1. de la herramienta.

Una vez instalado MySQL es necesario realizar varios pasos adicionales que se detallan a continuación.

Configuración del conector

Para realizar la integración con jUDDI necesitamos del conector respectivo (Conector/J5.1) que podemos descargarlo de http://dev.mysql.com/downloads/connector/j/5.1.html, una vez obtenido el archivo mysql-connector-java-5.1.7-bin.jar, procedemos a extraer los archivos y colocarlos en `CATALINA_HOME`\common\lib, para este caso específico la variable `CATALINA_HOME` tiene el valor “C:\juddi\apache-tomcat-5.5.23”.

Preparación de la base de datos

Es necesario crear una base de datos MySQL que servirá de destino para nuestro catálogo de Servicios, esta base de datos la creamos mediante la siguiente sentencia

```
CREATE DATABASE 'juddidb' DEFAULT CHARACTER SET utf8;
GRANT ALL ON juddidb.* TO 'juddidb_user' IDENTIFIED BY 'juddidb_pwd';
```

A continuación ejecutamos sobre el esquema juddidb los siguientes scripts, que crean y pueblan las tablas de datos

- create_database.sql
- insert_publishers.sql

Estas sentencias se encuentran comprimidas dentro del paquete juddi-2.0rc6.jar, ubicado en CATALINA_HOME\webapps\mysql\create_database.sql y la segunda sentencia la encontramos bajo juddi-sql\insert_publishers.sql.
Adicionalmente debemos realizar las siguientes modificaciones dentro de la sentencia insert_publishers.sql antes de ejecutarla:

```
INSERT INTO PUBLISHER (PUBLISHER_ID,PUBLISHER_NAME,EMAIL_ADDRESS,IS_ENABLED,IS_ADMIN,
MAX_BUSINESSES,MAX_SERVICES_PER_BUSINESS,MAX_BINDINGS_PER_SERVICE,MAX_TMODELS)
VALUES ('autentia','Autentia SL','igpuebla@autentia.com','true','true',25,20,10,100);
```

Configuración de las propiedades jUDDI

En primer lugar es necesario editar el fichero de configuración, el mismo que se encuentra bajo CATALINA_HOME\webapps\juddi\WEB-INF\juddi.properties. Debemos realizar los cambios necesarios que se muestran a continuación:

```
# The UDDI Operator Name
juddi.operatorName = autentia

# The i18n locale default codes
juddi.i18n.languageCode = es
juddi.i18n.countryCode = ES

# The UDDI Operator Contact Email Address
juddi.operatorEmailAddress = igpuebla@autentia.com

# straight JDBC
juddi.jdbcDriver=com.mysql.jdbc.Driver
juddi.jdbcUrl=jdbc:mysql://localhost:3306/juddidb?autoReconnect=true
juddi.jdbcUsername=juddi_user
juddi.jdbcPassword=juddi_pwd

# jUDDI database creation
juddi.isCreateDatabase=false
#juddi.tablePrefix=JUDDI_
juddi.databaseExistsSql=select * from ${prefix}BUSINESS_ENTITY
juddi.sqlFiles=juddi-sql/derby/create_database.sql,juddi-sql/insert_publishers.sql
```

A continuación modificaremos el fichero CATALINA_HOME\conf\Catalina\localhost\juddi.xml y cambiamos la etiqueta Resource por el siguiente fragmento:

```
<Resource name="jdbc/juddiDB" auth="Container" type="javax.sql.DataSource"
  maxActive="100" maxIdle="30" maxWait="10000" username="juddi_user" password="juddi_pwd"
  driverClassName="com.mysql.jdbc.Driver"
  url="jdbc:mysql://localhost:3306/juddidb?autoReconnect=true">
</Resource>
```

Luego modificaremos el archivo CATALINA_HOME\conf\server.xml donde se agregará la siguiente información entre las etiquetas <host> </host>:

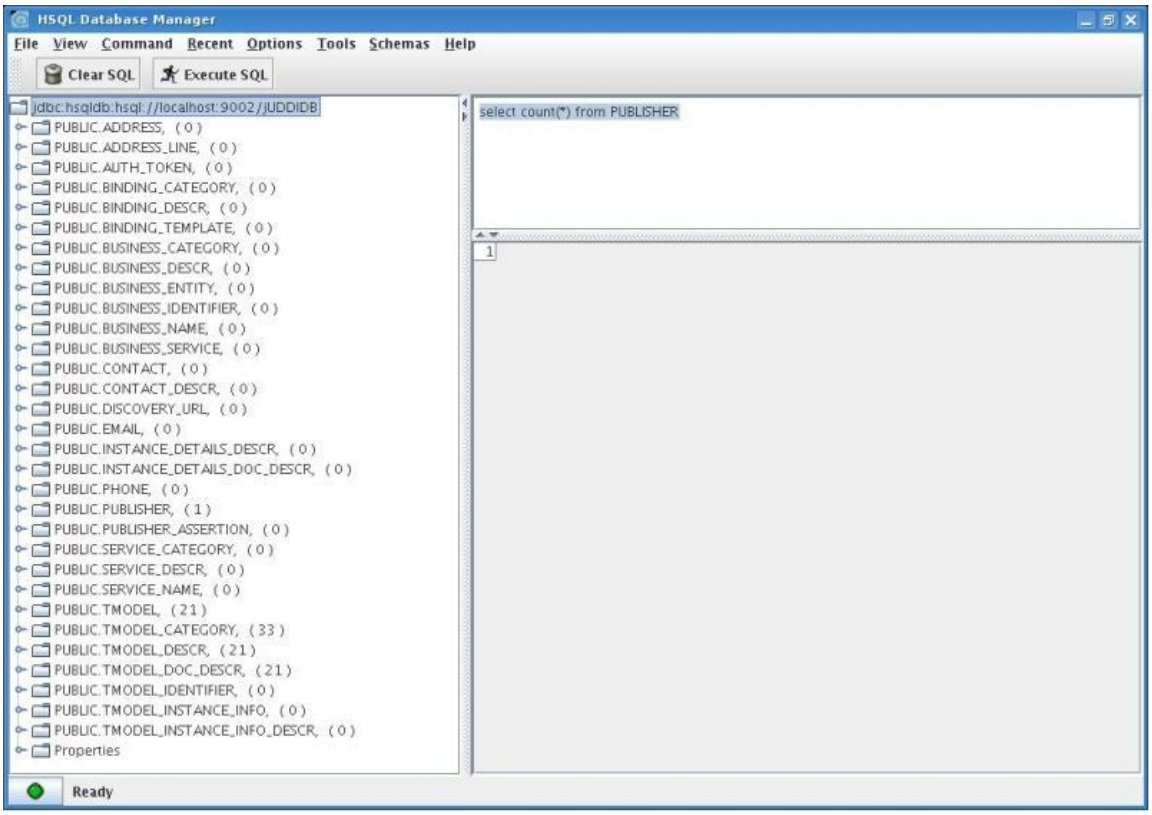
```
<Context path="/juddi" docBase="juddi" debug="5" reloadable="true" crossContext="true">

  <Logger className="org.apache.catalina.logger.FileLogger"
    prefix="localhost_juddiDB_log" suffix=".txt" timestamp="true"/>

  <Resource name="jdbc/juddiDB" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000" username="juddi_user" password="juddi_pwd"
    driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:3306/juddidb?autoReconnect=true">
  </Resource>

</Context>
```

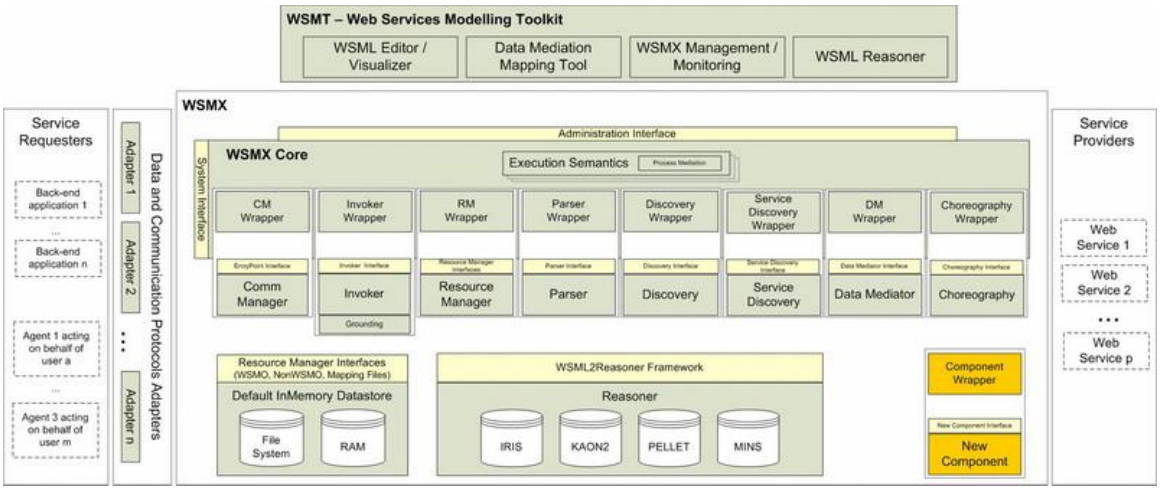
Una vez realizados todos estos cambios tenemos preparado jUDDI para ser utilizado con MySQL, las tablas generadas en la base de datos con la información correspondiente debe generarse como se muestra en la siguiente figura.



ANEXO 6

ENTORNO DE EJECUCIÓN WSMX

Este ambiente de ejecución se está desarrollando para servir como repositorio de Servicios Web Semánticos y Ontologías bajo los principios WSMO con la utilización del lenguaje WSML, la estructura en desarrollo de la herramienta se muestra en la siguiente figura.



Sin embargo, aún es una herramienta en desarrollo, es por este motivo que se ofrece varias alternativas de descarga orientadas a diferentes tipos de trabajo, estas alternativas se han tratado con mayor detalle en la sección 7.1.2.2., estas opciones de descarga pueden ser obtenidas a través de http://www.wsmx.org/ en la sección **Download** donde se muestran:

- Nightly-builds
- Stabel Binaries
- Sources
- SVN

Con propósitos investigativos se han utilizado las tres últimas opciones, pues la opción Nightly-builds son pequeñas betas liberadas a intervalos de tiempo indefinido.

Stable binaries

Esta opción permite obtener una versión estable de WSMX que ha sido utilizada y ha superado diferentes pruebas en su desempeño, para un análisis más profundo de esta herramienta se ha utilizado la versión 1.0 que se describe su uso en la sección 5.4.1.

Sources

Ofrece de manera similar diferentes distribuciones dividiendo la herramienta en los diferentes conjuntos de herramientas que lo conforman (esta distribución requiere adicionalmente la instalación de Apache MAVEN para su correcto funcionamiento) como son:

- WSMX-CORE
- WSMX-COMPONENTS
- WSMX-INTEGRATION-API

WSMX-CORE conforma el núcleo central de la herramienta, realiza el manejo de mensajes entre los diferentes componentes y controla los diferentes repositorios para Servicios Web, así como los diferentes razonadores que pueden ser integrados a esta herramienta.

WSMX-COMPONENTS contiene varios componentes que ayudan al razonamiento, descubrimiento, selección, invocación, mediación, coreografía y orquestación de los Servicios Web (varios componentes se encuentran aún en desarrollo).

WSMX-INTEGRATION-API permite el intercambio de mensajes entre cada uno de los componentes, así como la administración de los componentes a manera de plug-ins que pueden ser invocados o deshabilitados según se requiera.

SVN

Es el acrónimo de Subversión, es decir, es utilizado para el trabajo con herramientas en desarrollo que generalmente ocupan varios grupos de trabajo, en este caso, al ser una herramienta Open Source puede ser modificada, adaptada y mejorada por cualquier conjunto de usuarios.

Para descargar el código fuente de la herramienta en el IDE Eclipse se requiere de la descarga e instalación del plug-in subclipse como cliente para el manejo de versiones SVN.

APACHE MAVEN

Apache Maven es una herramienta para la gestión y construcción de proyectos Java, posee un modelo de configuración basado en XML, la descarga de Apache Maven puede realizarse a través del enlace http://maven.apache.org/download.html que nos permite escoger varias opciones, para la presente investigación se ha utilizado la versión 2.0 “apache-maven-2.2.0-bin.zip”

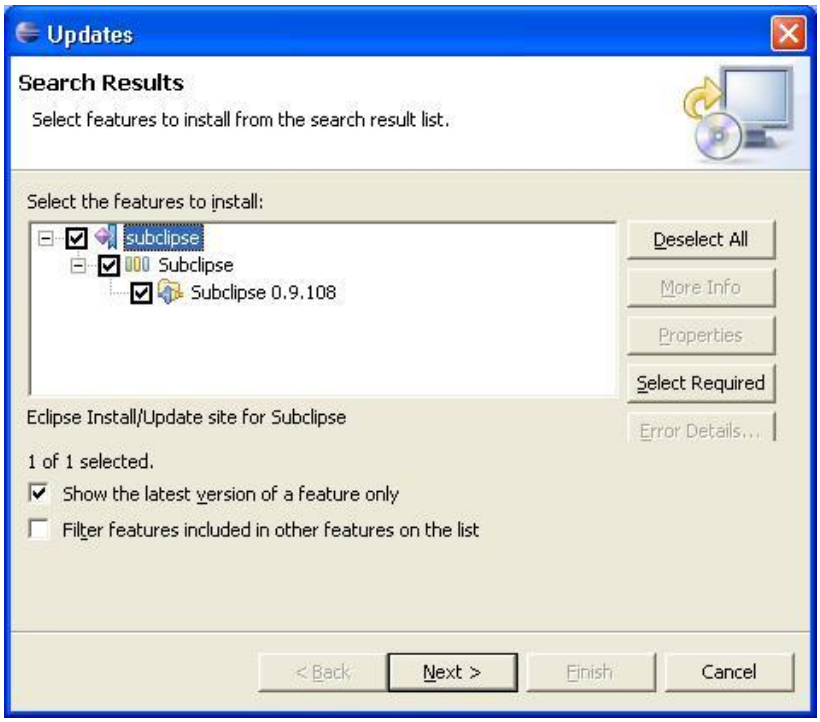
Una vez obtenido el archivo comprimido procedemos a extraerlo en cualquier ubicación en el disco local, preferentemente en C:\Maven2, Apache Maven requiere también la configuración de las variables especificadas en el Anexo 3 con su información particular.

Para la variable `M2_HOME` los valores son:
Nombre de variable: M2_HOME
Valor de la variable: C:\maven2

Para la variable `M2` los valores son:
Nombre de variable: M2
Valor de la variable: %M2_HOME%\bin

PLUG-IN SUBCLIPSE

La descarga e instalación del plug-in Subclipse se realiza mediante el IDE Eclipse en el menú Help con las opciones Help->Software Updates->Find and Install… donde se permite seleccionar la dirección donde se encuentra un plug-in específico mediante **Search for new feautures to install** mostrado en la siguiente figura

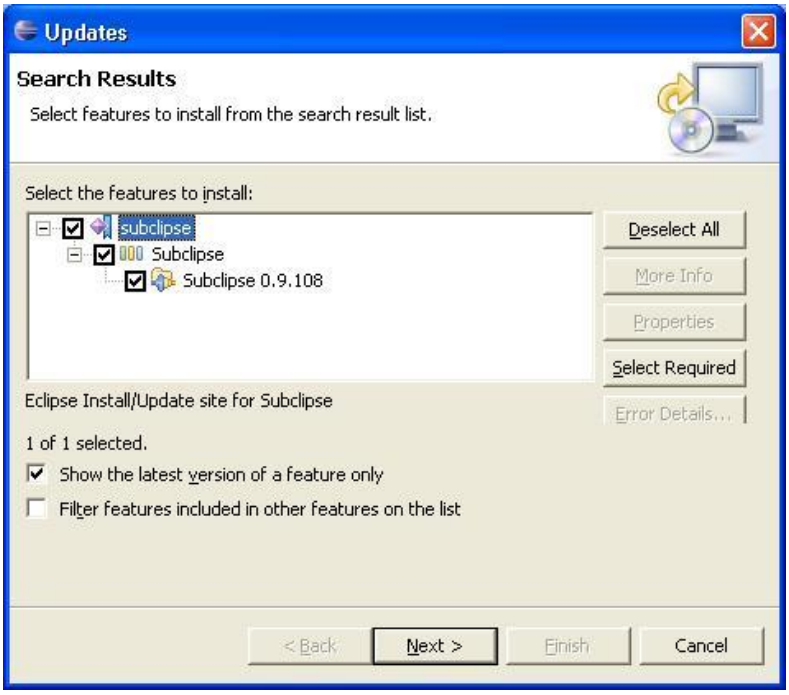


Finalmente pulsamos en el botón **New Remote Site** y colocamos la siguiente información

Name: Subclipse 1.4.x

URL: http://subclipse.tigris.org/update_1.4.x

Una vez descargada la información necesaria para instalar el plug-in Subclipse requerimos seleccionar todas las opciones mostradas en la siguiente figura, aceptar la declaración de licencia y esperar el proceso de instalación.



Una vez instalado el plug-in podemos hacer uso de la SubVersion de WSMX mediante el uso del cliente subclipse del IDE Eclipse, el cual lo podemos ubicar en las perspectivas del IDE Eclipse bajo Window->Open Perspective->Other como se muestra en la siguiente figura.