TRABAJO FIN DE CARRERA

WEB SEMÁNTICA Y SERVICIOS WEB SEMANTICOS

MEMORIA FINAL

ALUMNO: SANTIAGO MARQUEZ SOLIS
CONSULTOR: SINUHÉ ARROYO GOMEZ
FECHA DE ENTREGA: 11-01-2007
(INGENIERÍA TÉCNICA EN INFORMATICA DE GESTIÓN)

A las mujeres de mi vida...

...a mi madre por ser mi padre,
...a mi abuela por ser mi madre,
...a mi mujer por ser el pilar de mi vida,
...a mi hija por darme una vida vacía de errores

Dice un viejo refrán “renovarse o morir”, el cual llevado al mundo de las Tecnologías de la Información es si cabe un axioma básico al cual estamos acostumbrados. Este Trabajo Fin de Carrera trata de una de estas renovaciones, del paradigma de la Web Semántica y los Servicios Web Semánticos y cómo su adopción dentro del marco de trabajo de Internet va a suponer una verdadera revolución en cuanto a la forma en la que los servicios se ponen a disposición de los usuarios, los cuales, en este caso, abandonan el mundo de lo humano para convertirse en agentes o entes semi-inteligentes, que basados en unos simples principios lógicos, son capaces de hacer de Internet un lugar mucho más “humano”.

Tabla de contenido

Relación de figuras

Capítulo 1

Introducción

La Web Semántica y los Servicios Web Semánticos, son una extensión de la Web tradicional en donde los recursos están anotados de forma que los ordenadores pueden comprender la función o servicio que proporcionan y que esta emergiendo con fuerza dentro del panorama tecnológico de la Web, motivo por el cual alrededor de ellos aparecen una serie de conceptos, ideas, lenguajes, etc. que hace que sea necesario clarificar y entender para llegar a una comprensión completa del tema de estudio.

En los siguientes puntos se explican las características fundamentales del presente TFC y la línea de trabajo seguida para cumplir con los objetivos propuestos.

1.1 Justificación y contexto
1.2 Objetivos del TFC.
1.3 Enfoque y método seguido.
1.4 Planificación del proyecto.
1.5 Productos obtenidos
1.6 Breve descripción de los otros capítulos de la memoria.

1.1. Justificación y contexto

El desarrollo de Internet ha propiciado que prácticamente desde sus comienzos, la propia Internet sea el caldo de cultivo en donde se hayan producido numerosas e importantes revoluciones en la manera en la cual las personas e incluso los propios sistemas hacian uso de ella.

Desde este TFC queremos estudiar una de estas revoluciones, que consiste en la evolución de la Web actual hacia la Web Semántica. Desde nuestra perspectiva creemos que estamos ante lo que probablemente será la adopción más importante por tarte de la red, tanto por la naturaleza del cambio que implica dentro de la forma de organizar la información, como por el acceso a la misma, que ya no queda confinada dentro de los límites de lo humano, sino que permite llevarla más haya al permitir que agentes inteligentes, y que actualmente tienen su máxima representatividad dentro de los robots de los motores de búsqueda, puedan trabajar por nosotros al ser capaces de entender que servicios están presentes y que capacidades presentan al exterior.

1.2. Objetivos del TFC

1.2.1. Objetivos Generales

Por este motivo, en la memoria del presente TFC se pretende dar explicación a los siguientes conceptos:

- Las diferencias entre las distintas generaciones de Internet, desde la Primera Generación, en donde lo importante era la navegación, pasando por la Segunda Generación, en donde se primaba la interactividad entre usuarios y aplicaciones, para finalizar con la Tercera Generación en donde lo que se persigue principalmente es la colaboración entre personas (mediante el uso de wikis por ejemplo) o entre aplicaciones (con el uso de Servicios Web y la extensión de uso que se hace de los mismos gracias a los Servicios Web Semánticos).
- Cual es el papel del XML, es decir, todos los lenguajes que surgen a raíz de la Web Semántica representan la información mediante documentos XML que con un formato específico (XML Schema) se aplican a un área en particular.
- Una vez clarificado lo anterior, definiremos que se entiende por Web Semántica y por Servicio Web Semántico y aquí veremos una serie de conceptos muy importantes y que son básicos para tener una perspectiva general sobre la Web Semántica y los Servicios Web Semánticos, los conceptos que abordaremos serán:
    - Qué son las ontologías y para que sirven
    - Qué son las herramientas de definición de contenidos, e introduciremos por un lado, que es RDF y RDF Schema, y por otro lado que es OWL y OWL Schema.
    - Qué diferencia hay tras los conceptos de Web Superficial y Web Profunda
    - Qué son los Mapas Semánticos y su utilidad (introduciremos los ontology mappings y alignments)

En resumen, podemos decir que los objetivos de esta parte son bastante ambiciosos ya que pretende dar una visión de conjunto de los conceptos básicos que creemos son imprescindibles conocer cuando nos acercamos al estudio de la Web Semántica y de los Servicios Web Semánticos.

Una vez clarificados los conceptos anteriormente expuestos, el siguiente paso será profundizar en dos aspectos diferentes de la Web Semántica: los Lenguajes de Modelado de Procesos y los Lenguajes de Modelado de Servicios Web Semánticos, y centraremos nuestro trabajo en dos implementaciones particulares: el BPEL y el WMOS

En ambos casos, el objetivo es conocer sus características fundamentales y su utilidad dentro del ámbito de la Web Semántica y realizar el modelado de un problema completo. Además, en el caso del lenguaje de modelado de procesos el objetivo a cubrir será realizar la implementación del problema con un conjunto de herramientas Open Source disponibles en Internet.

A continuación se detallan los objetivos particulares a conseguir con el estudio de ambos tipos de lenguajes:

1.2.2. Objetivos sobre los Lenguajes de Procesos

- Describir que son los Lenguajes de Modelado de Procesos y donde se encuadran dentro del contexto de la Web Semántica. Además, se pretende explicar las diferencias existentes entre coreografías y orquestación de servicios y que beneficios/limitaciones presenta cada una de ellas.
- Definir que es un proceso de negocio y la estructura que presentan, así como el ciclo de vida típico que implementan.
- Conocer los componentes principales de BPEL como Lenguaje de Procesos orientado al desarrollo de orquestación de procesos
- Conocer los componentes principales de WS-CDL como Lenguaje de Procesos orientado al desarrollo de coreografías, y realizar el modelado

1.2.3. Objetivos sobre los Lenguajes de Servicios Web Semánticos

- Describir que es WSMO y para que sirve dentro del contexto de los Servicios Web Semánticos
- Cuales son los componentes principales de WSMO, y en concreto que se entiende en este contexto por: Ontologies, WebServices, Goals y Mediators
- Comprender qué relaciones se establecen con WSML como lenguaje estándar para describir los elementos que forman parte de WSMO

1.3. Enfoque y método seguido

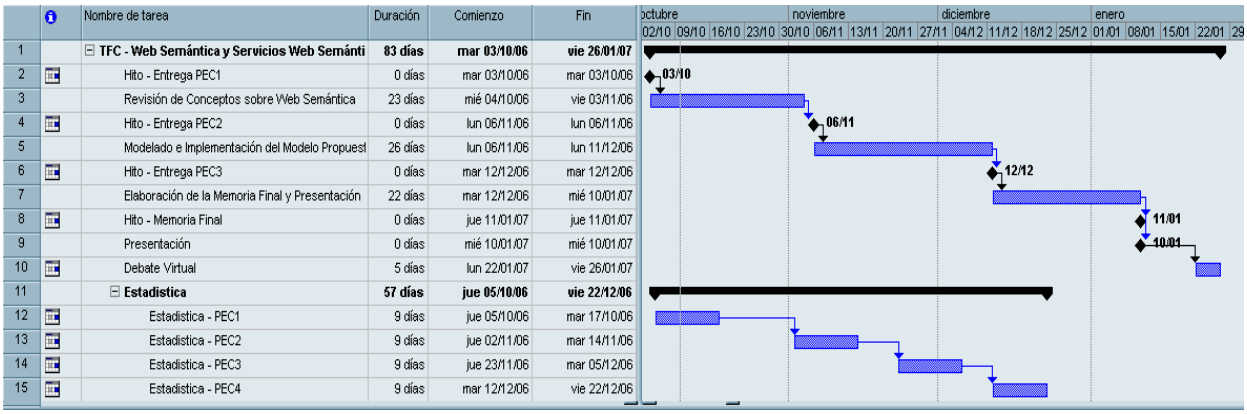
Para poder abordar los conceptos e ideas y por ende cumplir con los objetivos presentados, hemos seguido un método basado en la elaboración de cuatro PECs diferentes de complejidad creciente, tal y como se detalla en la planificación del proyecto en el punto siguiente.

1.4. Planificación del Proyecto

El proyecto se ha realizado mediante el desarrollo de diferentes PECs de dificultad creciente, en donde se han ido desgranado los diferentes conceptos. A continuación se detallan los hitos fundamentales a conseguir durante el desarrollo del presente TFC. Dado que en el enunciado del proyecto original no se indicó nada referente al contenido de los entregables de las diferentes PEC se ha supuesto lo siguiente:

- Contenido de la PEC2: Se corresponderá con la revisión de los conceptos fundamentales sobre Web Semántica. En esta PEC también se describió el Lenguaje de Modelización de Procesos BPEL y el Lenguaje para definir Servicios Web Semánticos WSMO.
- Contenido de la PEC3: Contenía la modelización e implementación de una coreografía de servicios con WS-CDL, y la modelización de un Servicio Web Semántico con WSML sobre el problema propuesto.
- Contenido de la PEC4 o Entrega Final: Contemplará el desarrollo completo del contenido del índice expuesto durante el desarrollo de la PEC1 (y corregido y ampliado a lo largo del desarrollo del resto de las PECs) y a partir del material desarrollado en las PEC 2 y 3. Además de la elaboración de dicho documento, se procederá a la creación de una presentación en PowerPoint resumen del mismo.

En la planificación se muestran también los hitos concernientes a las PECs de la asignatura de Estadística que se deben presentar y que es la única asignatura que resta para finalizar la carrera junto con el TFC.



	Nombre de tarea	Duración	Comienzo	Fin
1	**TFC - Web Semántica y Servicios Web Semánti**	**83 días**	**mar 03/10/06**	**vie 26/01/07**
2	Hito - Entrega PEC1	0 días	mar 03/10/06	mar 03/10/06
3	Revisión de Conceptos sobre Web Semántica	23 días	mié 04/10/06	vie 03/11/06
4	Hito - Entrega PEC2	0 días	lun 06/11/06	lun 06/11/06
5	Modelado e Implementación del Modelo Propuest	26 días	lun 06/11/06	lun 11/12/06
6	Hito - Entrega PEC3	0 días	mar 12/12/06	mar 12/12/06
7	Elaboración de la Memoria Final y Presentación	22 días	mar 12/12/06	mié 10/01/07
8	Hito - Memoria Final	0 días	jue 11/01/07	jue 11/01/07
9	Presentación	0 días	mié 10/01/07	mié 10/01/07
10	Debate Virtual	5 días	lun 22/01/07	vie 26/01/07
11	**Estadística**	**57 días**	**jue 05/10/06**	**vie 22/12/06**
12	Estadística - PEC1	9 días	jue 05/10/06	mar 17/10/06
13	Estadística - PEC2	9 días	jue 02/11/06	mar 14/11/06
14	Estadística - PEC3	9 días	jue 23/11/06	mar 05/12/06
15	Estadística - PEC4	9 días	mar 12/12/06	vie 22/12/06

Figura 1 - Planificación del Proyecto

1.5. Productos Obtenidos

Este documento, una presentación resumen del mismo y diferentes ficheros de implementación de las ideas expuestas en los diferentes capitulos del trabajo.

1.6. Breve descripción de los capítulos

El TFC se ha compuesto de 8 capítulos más dos anexos en donde se han tratado los siguientes temas:

- **Capitulo 1: Introducción**
  Se corresponde con la introducción general al TFC y las características del mismo tales como organización, descripción general, planificación, que sirven para introducir el contexto en el cual es desarrollado.

- **Capitulo 2. La Web Semántica**
  Este capítulo se dedica a explicar en detalle que es la Web Semántica y que características presenta, haciendo especial hincapié en las ontologías, el funcionamiento de los Servicios Web y los Servicios Web Semánticos, y la importancia de los lenguajes de modelado para facilitar su funcionamiento

- **Capítulo 3. Descripción del Problema a Modelizar**
  Este capítulo, se corresponde con el enunciado completo del problema que debemos modelizar utilizando los lenguajes de estudio

- **Capítulo 4. Modelizando e Implementado**
  En este capítulo abordamos la modelización e implementación del problema propuesto en el capítulo anterior con un Lenguaje de Modelado de Procesos. En este caso, dado que lo que queremos es crear una coreografía de servicios, nos centramos en las capacidades de WS-CDL para ello. Veremos como podemos crear la estructura (modelo) de nuestro proceso de negocio mediante una serie de herramientas gráficas, así como realizar la codificación de la solución propuesta

- **Capítulo 5. Modelado de Servicios Web Semánticos**
  En este capítulo realizaremos únicamente la modelización del problema propuesto en el capítulo 3 con el Lenguaje de Modelado de Servicios Web Semánticos, dado que WSMO no es un lenguaje físico propiamente dicho, utilizaremos WSML para realizar dicha modelización.

Capítulo 2

La Web Semántica

La Web Semántica y los Servicios Web Semánticos, son una extensión de la Web tradicional en donde los recursos están anotados de forma que los ordenadores pueden comprender la función o servicio que proporcionan y que está emergiendo con fuerza dentro del panorama tecnológico de la Web, motivo por el cual alrededor de ella, aparece una serie de conceptos, ideas, lenguajes, etc. que hace que sea necesario clarificar y entender para llegar a una comprensión completa del tema.

Este capítulo define varios términos clave y proporciona una descripción general sobre los conceptos básicos de la Web Semántica y de las tecnologías asociadas, en las siguientes secciones principales:

2.1 Los origenes de la Web
2.2 Generaciones de la Web
2.3 Problemas que presenta el modelo de Web actual
2.4 Definición. Qué es la Web Semántica
2.5 Ontologías
2.6 Necesidad del XML
2.7 Lenguajes para la descripción de Ontologías
2.8 Servicios Web y Servicios Web Semánticos
2.9 Inconvenientes de la Web Semántica

2.1. Los orígenes de la Web

La World Wide Web o simplemente Web como comúnmente se la denomina, tiene su origen haya por el año 1989 en los laboratorios del CERN[1], por tanto, y al contrario de lo que generalmente se cree, la Web no fue un invento americano, sino europeo. El padre de la Web fue Tim Berners-Lee quien en 1990 completó el primer servidor web y el primer cliente, para al año siguiente publicar el primer borrador de las especificaciones del HTML y del protocolo HTTP.

Originalmente la idea de la Web era disponer de un sistema, que permitiera servir como base para el intercambio de información entre investigadores, de forma que resultase cómodo y sencillo el proceso. No obstante, no es hasta la publicación del navegador Mosaic en el año 1993 por parte de la NCSA[2], cuando la Web se conoce a nivel mundial, extendiéndose como suele suceder en estos casos, primero por universidades y laboratorios y seguidamente al gran público. Mosaic fue un excelente programa, y gran parte de su éxito sin duda se debió a su disponibilidad en diversas plataformas (apareció

[1] CERN – Laboratorio Europeo de Física de Partículas, está ubicado en Ginebra (Suiza)
[2] NCSA – National Center for Supercomputing Application (Centro Nacional de supercomputación de Aplicaciones)

para Unix, Windows y Macintosh) y el ser completamente gratuito, además fue la base de otros navegadores como el popular Mozilla. Mosaic desapareció de manera oficial en el año 1997, aunque alguno de los miembros que lo desarrollaron crearon otro navegador que fue durante mucho tiempo el líder indiscutible del mercado, nos referimos a Netscape.

Hemos dicho, que la idea original de Tim Berners-Lee, era crear un sistema que sirviese de base para el intercambio de información entre investigadores, así como permitir la revisión de referencias que existiesen en el documento original mientras este se leía y siempre y cuando el usuario quisiera hacerlo, es así como tenemos la que sea probablemente la primera aplicación real de los conceptos de hipertexto.

Técnicamente, un documento de hipertexto consta de los elementos siguientes: por un lado tenemos los denominados **nodos o secciones**, que son las partes del texto que contiene información accesible para el usuario. Por otro lado, tenemos los **enlaces o hipervínculos**, que se establecen entre los nodos y permiten que el usuario realice una lectura secuencial o no de los mismos. Finalmente, el último elemento de un hipertexto son los **anclajes,** o puntos de activación de los enlaces.

Una vez que la web fue un hecho, su crecimiento fue, como hemos dicho, imparable, de manera que enseguida las posibilidades del hipertexto se quedaron cortas, el usuario empezó a demandar cada vez más la posibilidad de interactuar con las páginas, de manera que éstas pudiesen responder a las características particulares de cada uno de ellos. Es así como aparecen la creación de páginas web de manera automática (o dinámica), que permiten al usuario modificar el contenido de las páginas que visualiza en base a los datos que introduce en formularios y a una serie de procesos implementados, siguiendo alguna de las tecnologías disponibles, que en general, e interactuando con bases de datos generan la respuesta pedida.

La aparición de Java a principios de los años 90 supuso una gran revolución por su carácter multiplataforma, y la rápida expansión del uso de la tecnología JSP junto con el desarrollo por parte de Sun de la especificación J2EE, han sido sin duda determinantes para tener la posición dominante que ostentan.

No obstante, más adelante veremos que a pesar de todos estos esfuerzos por hacer la vida del usuario más fácil y agradable, aun quedan bastantes dificultades por resolver y que la Web Semántica no es más que otro intento por conseguir que la experiencia del usuario en su viaje por la Web sea mucho más placentera.

2.2. Generaciones de la Web

Con todo lo que hemos visto hasta el momento, podemos ver claramente que la Web ha pasado por tres etapas o generaciones muy claras, a saber:

- 1ª Generación o Generación de Contenido Estático
- 2ª Generación o Generación de Contenido Dinámico o Interactivo
- 3ª Generación o Generación de Contenido Colaborativo

No obstante, la evolución continua de la tecnología hace un poco complicado establecer los límites exactos de fecha en los cuales, podemos decir que la Web es de una generación u otra, ya que en la realidad lo que vemos es una superposición de cada una

de estas generaciones según el sitio web que examinemos. Sin embargo trataremos de hacer una aproximación grosera a efectos de clasificación[3].

2.2.1. 1ª Generación o Generación de Contenido Estático

La 1ª Generación de la Web o Generación de Contenido Estático, se corresponde con la Web que va desde que se crea por Tim Berners-Lee hasta la aparición de la 2ª Generación a mediados de los años 90.

Esta generación se caracteriza porque las páginas Web son completamente estáticas, es decir, el contenido que presentan es el que es y no permiten al usuario realizar ningún tipo de interacción con las mismas, salvo la manipulación propia del manejo del hipertexto y los hiperenlaces, esto es, saltar de una página web a otra.

2.2.2 2ª Generación o Generación de Contenido Dinámico

La 2ª Generación de la Web o Generación de Contenido Dinámico, se corresponde con la Web en la que aparecen las primeras técnicas para permitir la inclusión de contenido dinámico, se puede decir que es la Web que hoy por hoy está más extendida y es la que se utiliza más comúnmente.

Esta generación se caracteriza porque las páginas Web son generadas por alguna de las tecnologías vistas para la generación de contenido dinámico (CGIs, ASP, ASP.NET, JSP o PHP, entre otros), permiten la interacción con el usuario en un nivel en donde éste, puede hacer preguntas y el sistema presenta las respuestas en función de los criterios introducidos en formularios. La experiencia del usuario queda limitada a él y a la aplicación que utiliza.

2.2.3 3ª Generación o Generación de Contenido Colaborativo

De la 3ª Generación de la Web, podemos decir que es el modelo que se está imponiendo poco a poco, y que haciendo uso de las capacidades adquiridas en la generación anterior, permite que la experiencia del usuario con la Web mejore espectacularmente.

En esta generación, las aplicaciones van más allá de la mera interacción entre aplicación-usuario-aplicación, ahora ellas son el mecanismo que permiten que se produzca interacción entre usuario-usuario llegando a crearse un entorno de contenido colaborativo, en donde, el usuario es un participante más en la creación del contenido que aparece en la web. Ideas como los wikis, los blogs, etc. han cambiado la forma en la que el usuario interactúa con la web haciendo de esta un lugar mucho más rico e interesante para trabajar.

2.3. Problemas que presenta el modelo de Web actual

Por tanto, la Web que tenemos hoy día se podría decir que reúne las siguientes características:

- Funciona como una biblioteca digital hipermedia
- Puede funcionar usando una base de datos, y servir como una plataforma de aplicaciones.
- Es una plataforma para elementos multimedia
- Un esquema de nombrado

[3] Dan Gillmor autor de We Media (Nosotros el Medio) explica de una manera excelente estos periodos. Se puede descargar el artículo completo en http://www.hypergene.net/wemedia/espanol.php?id=P53

- La web es un lugar en el que los ordenadores se encargan de la presentación (lo fácil) y la gente se encarga de enlazar e interpretar (lo complicado).

Sin embargo, nos encontramos con que con la Web actual no resulta fácil dar respuesta a las siguientes preguntas:

- ¿cómo manejar los enormes volúmenes de información que se están generando?
- ¿cómo indexar de manera eficiente todo el material digital?
- ¿cómo encontrar de manera rápida, fácil y exacta, lo que buscamos entre tanta cantidad de información?
- ¿cómo saber qué servicios web se encuentran disponibles y para qué sirven?

Básicamente, todas estas cuestiones se resumen en dos puntos:

- Primero, la Web no incorpora mecanismos que permitan el procesado automático de la información necesario para manejar la gran cantidad de páginas que hay.
- Segundo, la Web no incluye mecanismos para la interoperabilidad completa de Sistemas de Información[4] basados en la misma Web.

Muchas de estas respuestas veremos pueden solucionarse, al menos en parte, con el uso de la Web Semántica.

2.4 Definición. Qué es la Web Semántica

En los puntos anteriores hemos visto cómo es la Web actual y hemos llegado a dos conclusiones importantes referentes a los problemas que ésta presenta y que se refieren a: la inexistencia de mecanismos de procesado automático para manejar la gran cantidad de páginas y a los problemas derivados de la falta de interoperabilidad. Vamos a centrarnos un poco en estos problemas y veremos como de la necesidad de resolverlos acabaremos definiendo de manera natural a la Web Semántica.

Aunque no es sencillo calcular el tamaño total de la web, se estima que hay alrededor de unos $4 * 10^9$ documentos disponibles en red, lo que viene a equivaler a unos 28 millones de libros, teniendo en cuenta que la American Research Libraries, que agrupa un total de 100 bibliotecas de EEUU tiene unos 3.7 millones de libros y que la biblioteca de Harvard (la mayor de EEUU) tiene catalogados 15 millones de libros, podemos hacernos una idea del tamaño que tiene la Web.

Resulta entonces que en la Web actual, tenemos un sistema que almacena, podríamos decir, prácticamente toda la información del mundo, y que presenta un acceso casi instantáneo a la misma desde cualquier lugar del planeta con una conexión a Internet. Por otro lado, la información disponible no es solamente de manera textual, sino que hay que sumar los documentos formados por imágenes, videos, presentaciones, etc. Además cualquier persona puede, al menos en teoría, añadir más información a la Web (mediante la creación de un nuevo sitio web por ejemplo). Pues bien, el conjunto de todas estas características es lo que viene a definir a la Web actual con el nombre de **Web Sintáctica.**

En la Web Sintáctica, nos vamos a encontrar con un conjunto de recursos enlazados entre si (formando un grafo dirigido) tal y como podemos ver en la siguiente figura:

[4] También conocidos por sus siglas S.I

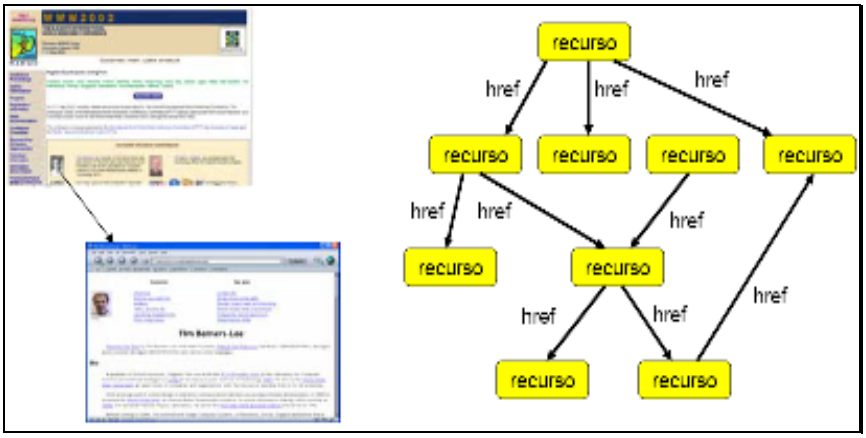


Figura 2 - La Web Sintáctica

Para movernos por esta gran cantidad de información, han aparecido numerosos buscadores que enlazan con ellas (Google, por ejemplo, indexa cerca de 9.000 millones de páginas), sin embargo, a pesar de la potencia que demuestran, aun quedan lejos de poder proporcionar al usuario las respuestas adecuadas a las preguntas que realizan, fundamentalmente por tres motivos:

- **no enlazan con la totalidad de páginas existentes,** se hace necesaria la actualización constante de los índices mediante robots de búsqueda automáticos (hablaremos más adelante sobre los robots)
- **la escasa precisión de los resultados**, algunas consultas devuelven varios millones de resultados distintos.
- y la **alta sensibilidad al vocabulario empleado en la búsqueda**, es decir, un documento que busquemos debe estar descrito con las mismas palabras que nosotros introduzcamos al buscar.

Por otro lado, tenemos también los problemas de interoperabilidad de aplicaciones, que se deben por falta de entendimiento técnico[5], sintáctico[6] y semántico[7].

Estos problemas, conllevan que las aplicaciones no puedan hablar entre ellas, lo que repercute en el coste de los servicios que las empresas proporcionan y relentiza la implantación de nuevos servicios útiles para el usuario.

La Web Semántica trata de resolver todos estos problemas, añadiendo a la Web Sintáctica la semántica que le falta para crear un entorno en donde podamos acceder a la información que necesitamos de un modo exacto y completo a la vez que se facilita el procesado de la misma y se resuelven los problemas de interoperabilidad entre aplicaciones que hemos resumido anteriormente.

Todo lo que hemos dicho está muy bien, pero todavía no hemos definido qué es la Web Semántica. ¿Qué es realmente la Web Semántica?, como suele suceder en estos casos, hay varias definiciones que nos interesa conocer.

[5] Interoperabilidad Técnica – Capacidad de un sistema de información para intercambiar datos
[6] Interoperabilidad Sintáctica – Capacidad de un sistema de información para leer datos de otros sistemas
[7] Interoperabilidad Semántica - Capacidad de un sistema de información para intercambiar información basándose en el significado común de los términos y expresiones que se utilizan.

La primera de las definiciones sobre Web Semántica viene de la mano del creador del concepto y del que ya hablamos anteriormente, Tim Berners-Lee define la Web Semántica de las siguientes maneras:

> "El primer paso es colocar los datos en la Web de un modo en que las máquinas puedan entenderlos naturalmente o convertirlos a esa forma. Esto crea lo que yo llamo la Web Semántica: una red de datos que pueden ser procesados directa o indirectamente por máquinas"
> **[Weaving the Web, 1999]**

> "La Web Semántica es una extensión de la Web en la cual la información se da mediante un significado bien definido, lo que facilita que los ordenadores y la gente trabajen en cooperación"
> **[The Semantic Web, Scientific American, Mayo de 2001]**

Tim Berners-Lee plasmó su idea original de la Web Semántica en la siguiente figura, en donde vemos que la Web Semántica se articula en base a conceptos como el enlazado de información, el hipertexto, los sistemas jerárquicos, etc.:

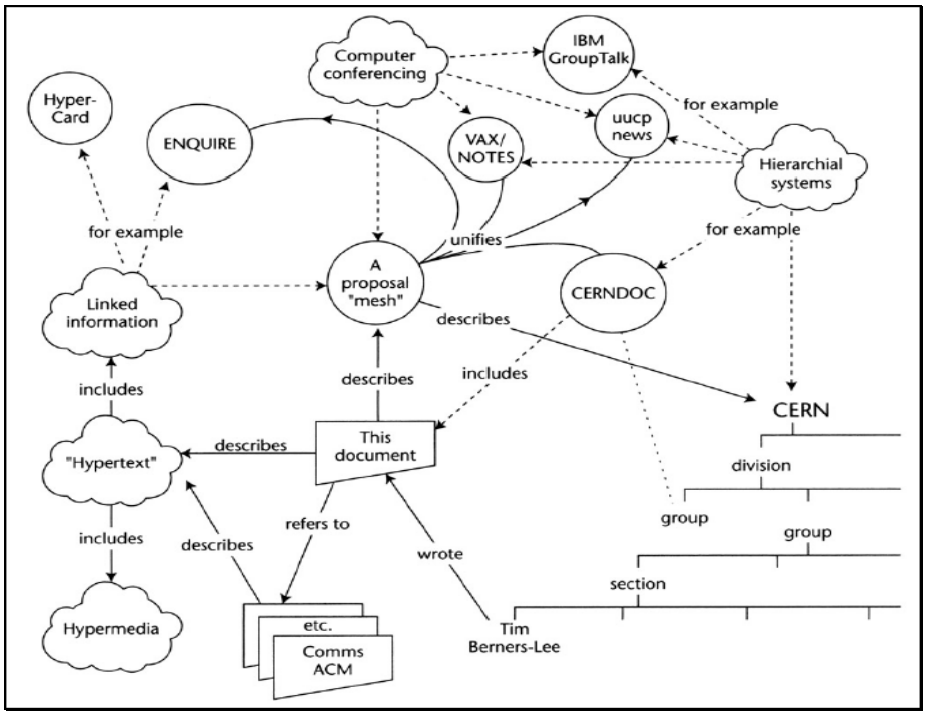


Figura 3 - La Web Semántica - Imagen de Tim Berners-Lee

Podemos precisar un poco si vamos a la definición oficial que existe en la página del W3C, en ella vemos que la Web Semántica se define como:

> "La Web Semántica es la Web de los datos. Hay muchos datos que usamos cada día y que no son parte de la Web. Podemos ver mis apuntes bancarios en la web, e incluso nuestras fotografías y citas en el calendario. Pero ¿podemos ver las fotos en un calendario para ver que es lo que estaba haciendo cuando las hice? ¿Puedo ver mis apuntes bancarios en el calendario?

La respuesta a estas preguntas es no. Y ¿porqué no? La respuesta es porque no tenemos una Web de datos. Y esto es debido a que los datos están controlados por las aplicaciones, y cada una los guarda y trata de manera particular

La Web Semántica trata sobre dos cosas. Sobre formatos comunes para el intercambio de datos, donde la Web original solamente se intercambian documento. Y trata sobre los lenguajes que representan los datos como objetos del mundo real.

La Web Semántica proporciona un framework común que permiten a los datos ser compartidos y reutilizados a través de las límites impuestos por aplicaciones, empresas o comunidades. Es un esfuerzo de desarrollo colaborativo liderado por la W3C y un gran número de investigadores y socios industriales. La Web Semántica se basa en el uso de Resource Description Framework (RDF[8]).”
[Introducción a la Web Semántica en http://www.w3.org/2001/sw/]

Dicho de otro modo, la Web Semántica se fundamenta en el hecho de que las máquinas comprendan el significado de la información disponible, pero desde un punto de vista diferente al humano. Estamos diciendo entonces que la Web Semántica es pura Inteligencia Artificial, aunque a las máquinas aun les queda un largo camino por recorrer para poder llegar a comprender siguiendo un esquema de razonamiento como el que hacemos los humanos, si que son capaces de llegar a conclusiones (deducciones o inferencia) mediante procesos de lógica-matemática. Ni que decir tiene que las conclusiones a las que se lleguen dependerán de la validez, o de lo buenas que sean, las reglas que se utilicen para llegar a estas.

Gráficamente entenderemos mejor esto con el siguiente gráfico:

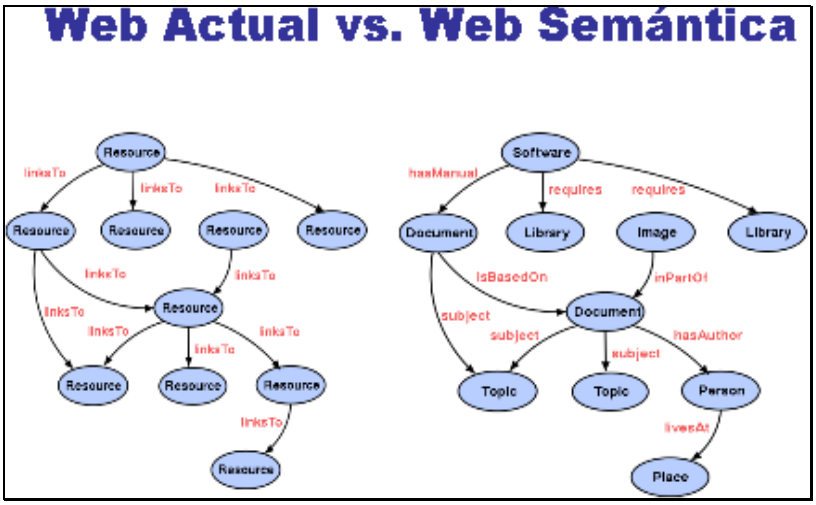


Figura 4 - Web Actual vs. Web Semántica

La figura muestra la forma en la que en la Web actual (sintáctica) se compone de recursos enlazados a través de enlaces (hipertexto) frente a la Web Semántica, en donde los elementos quedan caracterizados, de este modo vemos que una entidad “Software” **requiere** de una entidad “Biblioteca”, y que **tiene** un documento que es su manual de uso, de este modo podemos proseguir a lo largo de todo el grafo. De este modo podemos establecer una estructura semántica de un concepto al que luego podemos aplicar reglas lógicas para inferir nuevo conocimiento.

[8] RDF – Hablaremos de él más adelante como la forma más habitual de definir ontologías

La W3C utiliza la siguiente figura para mostrar cuales son los elementos estructurales básicos que forman parte de la Web Semántica:

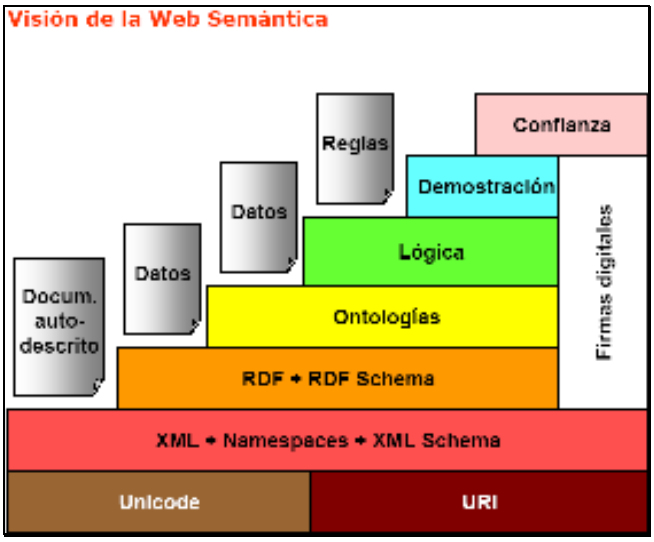


Figura 5 - Estructura de la Web Semántica vista desde el W3C

De los elementos anteriores, algunos de ellos como Unicode o URI ya los hemos mencionado, otros como XML (y tecnologías asociadas), ontologías y RDF, los veremos más adelante. Del resto reseñar lo siguiente:

- **Confianza**: Se refiere a las técnicas que aseguran la identidad y fiabilidad de los datos y servicios
- **Prueba**: Explican y verifican los pasos de los razonamientos lógicos
- **Lógica**: El razonamiento lógico determina si los datos son correctos e infiere las conclusiones oportunas a partir de los mismos.

Aunque como hemos dicho Tim Berners-Lee es el autor original de la Web Semántica, hoy en día en Europa, las personas que más están actuando como evangelizadores de la Web Semántica son Rudi Studer y Dieter Fensel.

Rudi Studer nació en 1951 en Stuttgart y es profesor en la Universidad de Karlsruhe (Alemania), es el responsable del grupo de investigación sobre gestión del conocimiento dentro del instituto AIFB[9], aparte de este cargo ostenta la presidencia de la Semantic Web Science Association y es director técnico del proyecto SEKT[10]

Dieter Frensel de origen alemán, es profesor y director del DERI Innsbruck Research Institute, en la Universidad de Innsbruck en Austria. A lo largo de su larga carrera profesional a impartido gran multitud de seminarios técnicos, dedicadonse en los últimos años a la divulgación e investigación de la aplicación de la Web Semántica. En el 2005 fue galardonado con el segundo premio sobre Sistemas Semánticos del FIT-IT, con su

[9] AIFB – Institute of Applied Informatics and Formal Description Methods, traducido como el Instituto de Informática Aplicada y Métodos de Descripción Formal.
[10] SEKT – Semantically Enabled Knowledge Technology, se engloba dentro del programa FP6 o Sixth Framwork Programme de la Unión Europea, que tiene por objeto promover la investigación y el desarrollo tecnológico.

proyecto de nombre GRISINO cuyo objetivo era la creación de un sistema Grid Semántico que permitiese la comunicación entre objetos distribuidos de manera inteligente.

Para finalizar este epígrafe vamos a ver una situación de ejemplo que podría resolverse con la implantación de la Web Semántica, este ejemplo es original de Tim Berners-Lee y fue publicado en la revista Scientific American en 2002:

- Los Beatles cantan “We can work it out” en el equipo de sonido de Bob cuando el teléfono suena…
- El teléfono avisa a todos los dispositivos que tienen “control de volumen” para que lo bajen automáticamente…
- Es su hermana Lucy al teléfono. Su madre va a necesitar sesiones de fisioterapia y tienen que turnarse para acompañarla. Acuerdan que sus agentes (que residen probablemente en algún móvil o PDA) hagan los arreglos necesarios.
- El agente de Lucy recoge la prescripción de los sistemas del médico, busca clínicas en un área de 20 millas que tengan un nivel de confianza por parte de organismos fiables de “alto” o “muy alto”, y que su rango de precios esté en un margen determinado.
- Una vez escogida la clínica, el agente de Lucy contacta con el de su hermano. Cotejan sus agendas personales, y fijan las fechas en las que cada uno acompañará a su madre.
- Los agentes plantean las opciones a sus usuarios. A Lucy le parece bien, pero a Bob no. A la hora a la que son las sesiones, hay mucho tráfico desde su casa al hospital escogido. Así que Bob especifica a su agente criterios más estrictos de hora y localización.
- El agente de Bob busca otras opciones y las encuentra siempre que cambie un par de citas de su agenda. Bob lo acepta.
- El nuevo plan es enviado al agente de Lucy, quién lo acepta.

2.5 Ontologías

Las ontologías proveen de una comprensión compartida y consensuada del conocimiento de un dominio que puede ser comunicada entre personas y sistemas heterogéneos. Fueron desarrolladas en el área de Inteligencia Artificial (IA) para facilitar el intercambio y reutilización del conocimiento.

El término ontología no es nuevo, y de hecho podemos remontarnos a la civilización griega para encontrarlo por primera vez. Aristóteles lo definió como la ciencia del ser, aunque en informática deberemos de ser un poco menos metafísicos y nos quedaremos con la idea siguiente: vocabulario compartido que describe un determinado dominio [11]y que se define en términos de un lenguaje formal de manera que sea manipulable automáticamente.

Decimos que toda ontología representa cierta visión del mundo con respecto a un dominio. Por ejemplo, una ontología que defina "ser humano" como "espécimen vivo o muerto correspondiente a la especie Homo sapiens; primate bípedo que pertenece a la familia de los homínidos, como los chimpancés, gorilas y orangutanes" expresa una visión del mundo totalmente distinta a la de una ontología que lo defina como "sujeto consciente y libre, centro y vértice de todo lo que existe; todos tienen la misma dignidad, pues han sido creados a imagen y semejanza de Dios". Toda ontología modela, mediante el uso de conceptos y relaciones, lo que conocemos sobre un dominio o área de conocimiento y por tanto, constituyen un vocabulario aceptado por una comunidad más o menos amplia.

[11] La idea de dominio se debe de entender como una parte del mundo real que resulta de interés. La descripción de un dominio de interés se llama modelo conceptual de dominio.

Así como la Ontología –nótese la mayúscula inicial– estudia los tipos de objetos que pueblan la realidad (así como sus propiedades y relaciones), las ontologías catalogan y definen los tipos de cosas que existen en un cierto dominio, así como sus relaciones y propiedades. Por ejemplo, una ontología del mundo empresarial usará conceptos como Venta, Compra, Transferencia, Pago, etc.; y relaciones como "Una Transferencia corresponde a una Venta o a una Compra", "Un Pago corresponde a una o varias Transferencias", etc.

Como decimos la idea de ontología no es nueva, y hay muchos ejemplos de ontologías que podemos encontrarnos tanto en el mundo informático como en otras ciencias, por ejemplo:

- Cyc[12]: Se utiliza para modelado de conceptos de sentido común para I.A. Utiliza lógica de predicados mediante lenguaje CycL.
- WordNet[13]: Se utiliza para el modelado de conceptos lingüísticos
- GALEN[14]: En este caso sirve para modelar conceptos médicos

Generalmente las ontologías se representan mediante clases, propiedades, atributos de las clases, relaciones entre clases y restricciones que pueden producirse tanto en los atributos como en las propiedades. Estas características que acabamos de indicar, deberán poder establecerse en los lenguajes de modelización de ontologías.

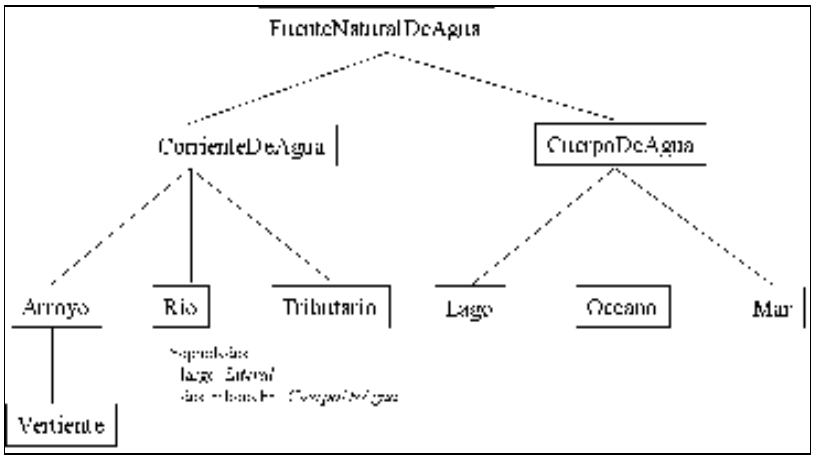


Figura 6 - Ejemplo de Ontologías para definir Fuentes Naturales de Agua

Más adelante veremos los lenguajes más comunes para la definición de ontologías.

Con las ontologías, los usuarios organizarán la información de manera que los agentes de software podrán interpretar el significado y, por tanto, podrán buscar e integrar datos mucho mejor que ahora. Gracias al conocimiento almacenado en las ontologías, las aplicaciones podrán extraer automáticamente datos de las páginas web, procesarlos y sacar conclusiones de ellos, así como tomar decisiones y negociar con otros agentes o personas. Por ejemplo, un agente inteligente que busque un vino que satisfaga las preferencias de un usuario, usará las ontologías vinícolas para elegir el vino (color, sabor, olor, embotellado) y empleará las ontologías empresariales para encargarlo a alguna tienda y regatear en el precio (siempre que se pueda). Otro ejemplo: mediante las ontologías, un agente encargado de comprar viviendas se podrá comunicar con agentes hipotecarios (de entidades bancarias) y con agentes inmobiliarios (de empresas constructoras e inmobiliarias).

[12] URL: http://www.cyc.com
[13] URL: http://www.globalwordnet.org
[14] URL: http://www.opengalen.org

2.6 Necesidad del XML

Probablemente la Web Semántica no podría haberse desarrollado sin la creación del elemento básico que va a permitir definir sus piezas clave. Uno de los resultados del empuje general hacia una estructura más semántica en la Web, fue el desarrollo del lenguaje XML[15], que permite a los desarrolladores usar su propio conjunto de etiquetas (markup-tags).

Pero esto que parece tan simple, esconde una potencia muy grande, ya que XML podemos considerarlo como un metalenguaje, o dicho de otra manera, un lenguaje para definir otros lenguajes de etiquetas estructurados. Como veremos más adelante, cuando abordemos los capítulos 3 y 4 dedicados a los lenguajes de modelado de procesos y a los lenguajes de modelado de servicios web semánticos respectivamente, hablaremos de BPEL y WSMO y veremos que estos dos lenguajes son lenguajes que utilizan XML para su definición, de mismo modo, los lenguajes para la definición de ontologías como RDF y OWL también son lenguajes XML. De hecho hoy día se acepta como válido que la Web Semántica se construirá basándose en XML y en las tecnologías asociadas al mismo.

Mientras que HTML es un lenguaje de marcado para documentos de hipertexto, XML es un lenguaje de marcado para documentos de todas las clases. Se dice que XML es extensible porque permite al programador asociar sus propias etiquetas a los datos.

La motivación original que impulso la aparición de XML fue la necesidad de la creciente globalización en los negocios electrónicos; el B2B[16], B2C [17]o C2C[18] necesitan por definición del intercambio de datos, además las soluciones del tipo EDI[19] han demostrado ser insuficientes en el mundo de hoy en día

Un documento XML consiste en una serie de etiquetas anidadas abiertas (<) y cerradas (>), entre los delimitadores de apertura y cierre se indica un nombre, denominado nombre de la etiqueta, donde cada etiqueta tiene ciertos valores. El programador define el nombre de cada una de las etiquetas y las combinaciones que pueden darse entre ellas, mediante dos técnicas diferentes: los documentos DTD[20] o los XML Schema.

La ventaja que presenta el uso de estas técnicas es que facilitan a los analizadores sintácticos o parsers comprobar si un documento es válido o no, es decir, verificar la gramática del documento y validar si las etiquetas que contiene junto con los anidamientos de las mismas son válidas o no.

El uso de XML aporta una serie de beneficios muy importantes:

- Cualquier documento y cualquier tipo de dato puede expresarse como un documento XML, de este modo se pueden definir los datos independientemente del lenguaje o plataforma siendo un formato universal para el intercambio de datos. Cualquiera que lo desee puede crear un documento XML conforme a una DTD o XML Schema que necesite y utilizarlo en sus aplicaciones o transacciones. Este beneficio se resume generalmente diciendo que XML es un lenguaje abierto.

[15] Extensible Markup Language
[16] B2B – Business to Business
[17] B2C – Business to Customers
[18] C2C – Customers to Customers
[19] EDI – Electronic Data Interchange (Intercambio de datos electrónico)
[20] DTD – Document Type Definition o Definición de Tipo de Documento

- Es auto descriptivo, se almacenan datos y la estructura de los mismos. Resulta muy fácil entender un documento XML echando simplemente un vistazo al mismo. Dicho de otra forma, XML es un metalenguaje.
- Es fácilmente compartible a través de la red, los ficheros XML son ficheros de texto, cualquier plataforma es capaz de operar con ellos.
- Se basa en el uso de Unicode[21], lo que permite crear documentos para cualquier idioma.
- Al hacer uso de los DTDs o XML Schema se pueden validar, lo que resulta muy útil cuando realizamos transacciones.
- Permite la integración tanto de datos estructurados (como las tablas relacionales) y poco estructurados (como los documentos)
- Finalmente XML es un lenguaje neutro, entendiendo por neutro que le es independiente el mecanismo de presentación que se utilice para mostrar los datos del mismo, lo que facilita aun más su interoperabilidad entre dispositivos de distinto tipo (por ejemplo la web vista desde un navegador y vista a través del móvil)

2.7 Lenguajes para la descripción de Ontologías

En las secciones anteriores hemos descrito que se entiende por ontologías y hemos explicado como las características de metalenguaje del XML le hacen ideal para la Web Semántica, sin embargo necesitamos adaptar XML para que realmente sirva para representar ontologías.

XML se ha adaptado de diversas maneras, pero las más conocidas son las siguientes:

- **SHOE**: lenguaje de representación del conocimiento basado en HTML (extensión de las etiquetas XML a las que se añade significado)
- **OWL (Online Writing Lab)**: habilita el procesado semántico de la información mediante máquinas.
- **OIL**: lenguaje de intercambio de ontologías. La unión de DAML+OIL puede derivar en la vía más práctica para que los agentes inteligentes comprendan el contenido de las páginas y, por tanto, desarrollen la Web Semántica.
- **XOL**: es un lenguaje para el intercambio de ontologías: se usa como lenguaje intermedio; maneja metadatos, definiciones de clases y de objetos, bases de hechos, etc...XOL tiene una sintaxis basada en XML y una semántica que emula el modelo de conocimiento
- **OKBC** (Open Knowledge Base Connectivity): Es expresivo, fácilmente legible, comprensible y procesable por agentes humanos y no humanos.
- **IFF:** es el lenguaje de marcas asociado a la Information Flow Framework. Extiende XML incorporando ideas de RDF, OIL, OML.
- **RDF** (Resource Description Framework): es una recomendación del W3C, basado en XML, que proporciona la tecnología para escribir metadatos que describen recursos en la Web.
- **WSML** (Web Service Modeling Language) es un lenguaje para el modelado de Servicios Web Semánticos basado en WSMO (Web Service Modeling Ontology), el lenguaje WSML consta de una serie de lenguajes accesorios que definen determinados aspectos referentes a la semántica del servicio web, estos lenguajes son: WSML-Core, WSML-DL, WSML-Flight, SWRL y WSML-Full.

[21] Unicode – Asigna enteros no negativos a los caracteres escritos de cada idioma y establece la codificación binaria de esos números, por ejemplo UTF-8 es un ejemplo de codificación Unicode

Debido a la multitud de variantes posibles que existen para representar las ontologías, se ha hecho necesaria la aparición de herramientas denominadas **editores de ontologías** que permiten automatizar y facilitar el desarrollo de ontologías mediante un entorno de trabajo visual. Muchos de ellos permiten la conversión entre diferentes lenguajes de representación.

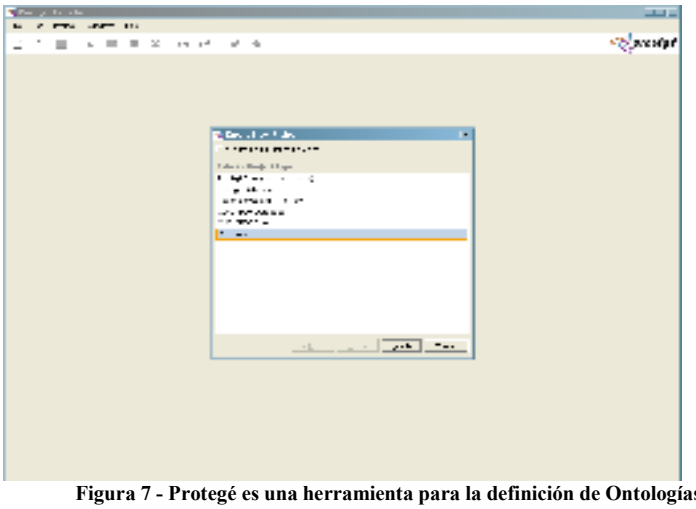


Figura 7 - Protegé es una herramienta para la definición de Ontologías

Por su importancia, de los lenguajes anteriores para la definición de ontologías vamos a ver con un poco más de detalle dos de ellos, concretamente centraremos nuestro estudio en RDF y en OWL ya que a corto plazo son los estándares que la industria parece que está adoptando.

2.7.1 RDF y RDF Schema

RDF [22]o Resource Description Framework, no es realmente un lenguaje que sirva para el modelado de ontologías, sino más bien para la definición de metadatos, ya que proporciona un modo sencillo de expresar afirmaciones acerca de recursos web, tratando de aportar interoperabilidad ante la multiplicidad de formatos incompatibles existentes.

Decimos que no es un lenguaje que permita modelar ontologías, porque vimos que una ontología se compone de una clase, relaciones, atributos, etc. y estos elementos no son del todo implementables en un documento RDF.

RDF pretende ser un soporte para la expresión de relaciones entre recursos de cualquier tipo con carácter universal y distribuido de modo que facilite la identificación de la información sin dar lugar a ambigüedades principalmente encaminado a ser procesado por aplicaciones en lugar de clientes humanos.

RDF se basa en el uso de afirmaciones (o statements). Por ejemplo, una afirmación podría ser la siguiente: “la página http://www.lawebsemantica.com” tiene como creador a Santiago Márquez”. Aunque otros ejemplos de afirmaciones que podríamos hacer serían: “La página http://www.lawebsemantica.com/index.html” tiene como fecha-de-creación el 01/01/2006” o “la página http://www.lawebsemantica.com/index.html” tiene como idioma el español”

En el estándar RDF, las afirmaciones se componen de una serie de elementos principales, como son:

- **Sujeto**: Sobre qué vamos a hacer una afirmación (la página en este caso). También se suelen denominar recursos y la idea es ver los recursos como objetos

[22] RDF – Se puede consultar la definición del W3C en: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/

de los que queremos hablar (otros ejemplos podrían ser libros, editores, gente, hoteles, etc.). Todo sujeto o recurso queda identificado por una URI, que en el caso de los documentos RDF hay que entenderlas como identificadores que apuntan a partes especificas del documento.

- **Predicado**: La propiedad del recurso que estamos describiendo (quién es su creador en este caso). Se utilizan para describir las relaciones entre los sujetos y también quedan identificados por una URI.
- **Objeto**: Lo que vamos a asignar como valor a la propiedad anterior (el nombre de su autor).

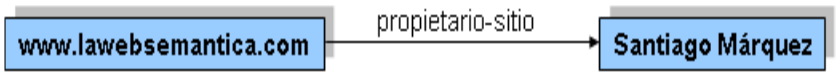
Existen varias formas de representar las afirmaciones RDF,

- **Declaración Explícita:** Se basa en el uso de una fórmula lógica del tipo: (x, P, y), donde el predicado binario P relaciona el objeto x con el objeto y. Hay que fijarse en que hemos indicado predicado binario adrede y esto es debido a que RDF solo acepta este tipo de predicados. Un ejemplo de declaración explícita sería la siguiente:

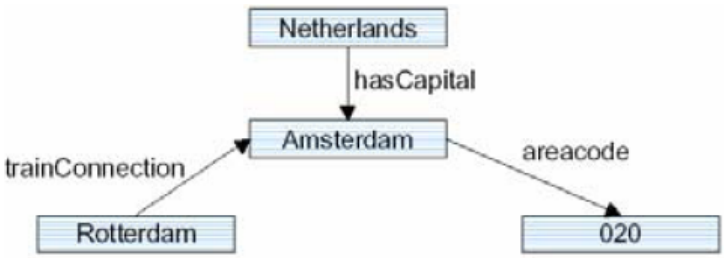
  ("Santiago Márquez", http://www.lawebsemantica.com/propietario-sitio, http://www.lawebsemantica.com)

  Qué indica que el propietario del sitio web "www.lawebsemantica.com" es "Santiago Márquez". La parte asociada al predicado se corresponde con el espacio de nombres utilizado para definir de donde obtenemos el predicado.

- **Grafos RDF**: se basa en la identificación de la terna[23] sujeto - predicado - objeto como un grafo en el que se conectan los dos nodos correspondientes a los elementos sujeto y objeto a través de un arco que representa el predicado. La afirmación anterior quedaría representada del modo siguiente:



  No debemos olvidar y que es un aspecto importante de RDF, y es que el sujeto de una afirmación puede ser sujeto de otra declaración diferente, tal y como se muestra en el siguiente grafo (en donde los Paises Bajos tiene como capital Ámsterdam que a su vez sirve de sujeto para las relaciones que se establecen entre Rótterdam (como conexión de tren) y 020 (como código de área):



- Mediante un documento XML que sigue una estructura determinada.

[23] Terna – En alguna bibliografía también aparece como tripleta

```
<rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar">
  <ex:editor>
    <rdf:Description>
      <ex:homePage>
        <rdf:Description rdf:about="http://purl.org/net/dajobe/">
        </rdf:Description>
      </ex:homePage>
    </rdf:Description>
  </ex:editor>
</rdf:Description>
```

De las tres, a nosotros nos interesa la representación mediante un documento XML que es la que permite su proceso automático.

Si nos fijamos en cualquiera de los ejemplos, vemos que la forma en la cual se indican las afirmaciones es un poco farragosa y demasiado larga, por lo que se suele utilizar lo que se denomina Nombre Cualificados (o QName), los cuales contienen un prefijo que identifica el espacio de nombres (namespaces24) utilizado y luego el nombre local. Por ejemplo, cuando escribíamos http://www.lawebsemantica.com/propietario-sitio" podemos asignar al espacio de nombres el prefijo, por ejemplo, "dc" de modo que el predicado anterior quedaría del modo siguiente: "dc:propietario-sitio". El uso de los nombres cualificados es una tónica general en los documentos XML que definen RDF.

En XML un documento RDF es una lista de descripciones de recursos. Cada descripción contiene un recurso y la lista de las propiedades que lo definen, los valores de una propiedad pueden ser literales u otros recursos (que a su vez se definen con su URI correspondiente).

Las descripciones se representan con los elementos XML <rdf:Description> con uno de los siguientes atributos: rdf:about o rdf:ID, en el caso de que no se utilice ninguno de estos atributos estaríamos ante lo que se denomina como **recurso anónimo.** La diferencia entre el uso de about o ID es que el primero se utiliza para describir un recurso y el segundo para definirlo (lo normal es definir el recurso y luego describirlo).

Las propiedades pueden contener texto:

```
<dc:Title>Sitio web La Web Semántica</dc:Title>
```

O pueden tener como valor otro recurso, en cuyo caso se representará mediante un elemento vacío con el atributo rdf:resource cuyo valor es el URI del objeto.

```
<dc:Creator rdf:resource="mailto:smarquezsolis@gmail.com "/>
```

O pueden contener otro elemento rdf:Description:

```
<dc:Creator>
  <rdf:Description rdf:about="mailto:smarquezsolis@gmail.com">
      <os:TrabajaCon rdf:resource="mailto:smarquezsolis@gmail.com "/>
  </rdf:Description>
</dc:Creator>
```

En 1995 se creó la iniciativa Dublín Core que es un estándar abierto de metadatos para describir objetos usando RDF y facilitar una manera sencilla de interactuar y acceder a ellos al compartir la definición de las propiedades. Podemos clasificar el conjunto de

[24] namespaces – En XML los namespaces se representan mediante la palabra xmlns

elementos Dublin Core en 3 grupos que indican la clase o el ámbito de la información que contienen:

- Elementos relacionados principalmente con el contenido del recurso:
    - **Title** (titulo)
    - **Subject** (tema)
    - **Description** (descripción)
    - **Source** (fuente)
    - **Languaje** (lenguaje)
    - **Relation** (relación)
    - **Coverage** (cobertura).
- Elementos relacionados principalmente con el recurso cuando es visto como una propiedad intelectual:
    - **Creator** (autor)
    - **Publisher** (editor) y, otras colaboraciones
    - **Contributor** (otros autores/colaboradores)
    - **Rights** (derechos).
- Elementos relacionados principalmente con la instanciación del recurso:
    - **Date** (fecha)
    - **Type** (tipo de recurso)
    - **Format** (formato)
    - **Identifier** (identificador)

Más adelante cuando hablemos de WSMO, veremos que a la hora de describir las propiedades no funcionales, está muy extendido el uso de este vocabulario.

Uno de los problemas que tiene RDF es que no proporciona vocabularios específicos para hacer afirmaciones acerca de recursos, esta limitación se ha tratado de resolver con la creación de **RDF Schema**, que permite la creación de vocabularios con sus clases, propiedades, etc.

Las clases en RDF Schema se utilizan para clasificar los diferentes recursos que deseamos describir, de modo que una clase debemos verla como un concepto genérico, generalmente podremos definir las clases haciendo uso de las palabras "**es un".** Por ejemplo: "www.google.es es un buscador" o "www.cursos.com/cursoXML es un curso".

2.7.2 OWL

El OWL (Web Ontology Language) o Lenguaje de Ontologías para la Web, se ha convertido en recomendación del W3C el 10 de febrero de 2004 (http://www.w3.org/TR/owl-ref/). La web: OWL Web Ontology Language Overview (http://www.w3.org/TR/owl-features/) ofrece una explicación detallada de en qué consiste este lenguaje, para qué se usa y cuáles son los conceptos fundamentales empleados por dicho lenguaje.

El Web Ontology Language OWL está diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos. OWL puede usarse para representar explícitamente el significado de términos en vocabularios y las relaciones entre aquellos términos. Esta representación de los términos y sus relaciones se denomina una ontología. En realidad, OWL es una extensión del

lenguaje RDF y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo que éste.

OWL posee más funcionalidades para expresar el significado y semántica que XML, RDF, y RDFS, pero OWL va más allá que estos lenguajes pues ofrece la posibilidad de representar contenido de la Web interpretable por máquina. OWL es una revisión del lenguaje de ontologías web DAML+OIL que incorpora lecciones aprendidas desde el diseño y aplicaciones de DAML+OIL.

El lenguaje OWL tiene 3 sub-lenguajes que incrementan su expresión: OWL Lite, OWL DL, y OWL Full.

La recomendación introductoria de este lenguaje está indicada para quienes pretendan obtener una primera impresión de la capacidades de OWL. La especificación ofrece una introducción a OWL para describir informalmente las características de cada uno de los sub-lenguajes de OWL. Se precisa algún conocimiento de RDF Schema para comprender este documento, pero no es esencial. El documento recomienda que los lectores interesados consulten OWL Guide para descripciones más detalladas y para ejemplos más extensos de las características de OWL. La definición formal normativa de OWL puede encontrarse en OWL Semantics and Abstract Syntax.

El Web Ontology Language OWL es, en realidad, un lenguaje de etiquetado semántico para publicar y compartir ontologías en la World Wide Web. OWL se ha desarrollado como una extensión del vocabulario de RDF (Resource Description Framework) y deriva del lenguaje DAML+OIL Web Ontology.

El motivo del desarrollo de este lenguaje ha sido la puesta en marcha de la Web Semántica, en realidad, una visión para el futuro de la Web en la cual el significado de la información será dado de forma explícita haciendo que las máquinas automaticen de forma más fácil los procesos e integren la información disponible en la Web. La Web Semántica se construirá sobre la sintaxis del lenguaje XML que se mejorará mediante el uso de esquemas RDF para representar el contenido de los datos. El primer nivel sobre RDF requerido para la Web Semántica es un lenguaje de ontologías que pueda describir formalmente el significado de la terminología usada en los documentos web. Si las máquinas son capaces de realizar tareas de razonamiento sobre los documentos en los que se utilice una semántica que vaya más lejos que la semántica básica de RDF Schema, la Web Semántica irá por buen camino.

OWL ha sido diseñado para conocer las necesidades para un lenguaje de ontología de la Web y es, pues, parte de las recomendaciones del W3C relacionadas con la Web Semántica.

OWL añade más vocabulario para describir propiedades y clases: entre otras, relaciones entre clases (ejemplo, inconexas), cardinalidad (ejemplo "exactamente uno"), igualdad, más ricos tipos de propiedades, características de las propiedades (por ejemplo, simetría), y clases enumeradas.

OWL ofrece tres sub-lenguajes de expresión incremental diseñados para ser usados por comunidades específicas de desarrolladores y usuarios según el nivel de expresividad que precisen éstos.

- **OWL Lite**: da soporte a aquellos usuarios que primordialmente necesitan una clasificación jerárquica y restricciones simples. Por ejemplo, soporta restricciones cardinales, pero solamente permite valores cardinales de 0 ó 1. Así pues, es más simple proveer herramientas de soporte para OWL Lite. OWL Lite ofrece una

rápida ruta de migración para tesauros y otras taxonomías. En resumen, OWL Lite tiene una más baja complejidad formal que OWL DL.

- **OWL DL**: da soporte a aquellos usuarios que quieren la máxima expresividad mientras conservan completamente la computacionalidad (todas las conclusiones son garantizadas para ser computables) y resolubilidad (todas las computaciones terminarán en tiempo finito). OWL DL incluye todos los constructos del lenguaje OWL, pero pueden usarse solamente bajo ciertas restricciones (por ejemplo, mientras una clase puede usarse por una subclase de muchas clases, una clase no puede ser una instancia de otra clase). OWL DL se denomina así debido a su correspondencia con las descripciones lógicas (DL), un campo de investigación que han estudiado los lógicos para la fundación formal de OWL.
- **OWL Full**: da soporte a usuarios que requieren el máximo de expresividad y la libertad sintáctica de RDF sin garantías computacionales. Por ejemplo, en OWL Full una clase puede ser tratada simultáneamente como una colección de individuos y como un individuo por derecho propio. OWL Full permite a una ontología aumentar el significado del vocabulario predefinido (RDF ó OWL). Es poco probable que algún software racional pueda soportar por completo el razonamiento para cada característica de OWL Full.

2.7.3. WSML

Para describir los servicios web, el W3C recomienda utilizar el Web Service Modeling Language (WSML) o Lenguaje de Modelado de Servicios Web que provee una sintaxis formal y una semántica para el modelado de ontologías de servicios web (Web Service Modeling Ontology -WSMO-). WSML está basado en diferentes lógicas formales:

- Lógica de descripción (Description Logics)
- Lógica de Primer Orden (First-Order Logic)
- Lógica de Programación (Logic Programming)

Cada una de ellas, se usan para el modelado de Servicios de la Web Semántica. Más adelante volveremos sobre este lenguaje.

2.8 Servicios Web y Servicios Web Semánticos

2.8.1. Servicios Web

En este apartado vamos a hablar de la tecnología central objeto de estudio de este TFC, comprender los fundamentos que hay detrás de los Servicios Web será imprescindible cuando abordemos otros conceptos más complejos como la evolución que presentan y que se materializan en los Servicios Web Semánticos.

Un Servicio Web es un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario.

Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar, generalmente esta arquitectura de referencia recibe el nombre de **Arquitectura SOA o Arquitectura Orientada a Servicios**,

y se caracteriza por la utilización de servicios para dar soporte a los requerimientos de software de los usuarios.

SOA proporciona una metodología y un marco de trabajo para documentar las capacidades de negocio y puede dar soporte a las actividades de integración y consolidación.

En un ambiente SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. Llegados a este punto conviene hacer una aclaración importante y es que si bien la mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP y WSDL) en su implementación, se puede implementar una SOA utilizando cualquier tecnología basada en servicios.

Al contrario de las arquitecturas orientadas a objetos, las SOAs están formadas por servicios de aplicación débilmente acoplados y altamente interoperables (ya vimos que objetivos de interoperabilidad necesitábamos conseguir). Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación, generalmente esta definición formal se realiza en WSDL[25] o Web Service Description Language del que en breve hablaremos.

La definición de la interfaz encapsula (oculta) las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo (como Java o .NET). Con esta arquitectura, se pretende que los componentes software desarrollados sean muy reusables, ya que la interfaz se define siguiendo un estándar; así, un servicio programado en C# podría ser usado por una aplicación Java.

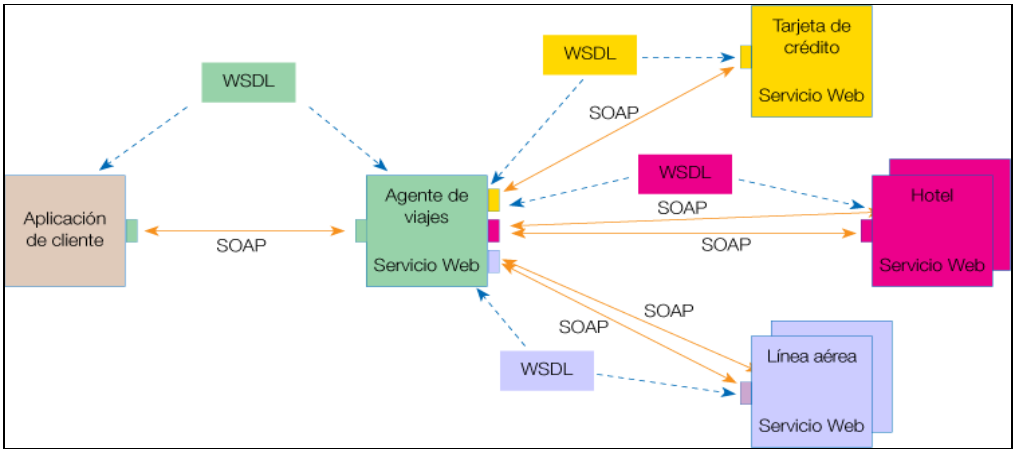


Figura 8 - Ejemplo tomado de las guías breves del W3C

En el gráfico, un usuario (que juega el papel de cliente dentro de los Servicios Web), a través de una aplicación, solicita información sobre un viaje que desea realizar haciendo una petición a una agencia de viajes que ofrece sus servicios a través de Internet. La agencia de viajes ofrecerá a su cliente (usuario) la información requerida. Para proporcionar al cliente la información que necesita, esta agencia de viajes solicita a su vez información a otros recursos (otros Servicios Web) en relación con el hotel y la línea aérea. La agencia de viajes obtendrá información de estos recursos, lo que la convierte a su vez en cliente de esos otros Servicios Web que le van a proporcionar la información solicitada

[25] WSDL – En castellano Lenguaje de Descripción de Servicios Web

sobre el hotel y la línea aérea. Por último, el usuario realizará el pago del viaje a través de la agencia de viajes que servirá de intermediario entre el usuario y el servicio Web que gestionará el pago.

En todo este proceso intervienen una serie de tecnologías que hacen posible esta circulación de información. Por un lado, estaría SOAP (Protocolo Simple de Acceso a Objetos). Se trata de un protocolo basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja. Los datos pueden ser transmitidos a través de HTTP, SMTP, etc. SOAP especifica el formato de los mensajes.

El mensaje SOAP está compuesto por un **envelope** (sobre), cuya estructura está formada por los siguientes elementos: **header** (cabecera) y **body** (cuerpo). Esto es una ventaja ya que facilita su lectura por parte de humanos, pero también es un inconveniente dado que los mensajes resultantes son más largos.

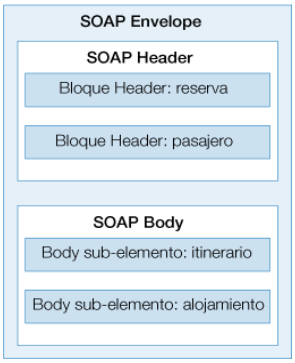
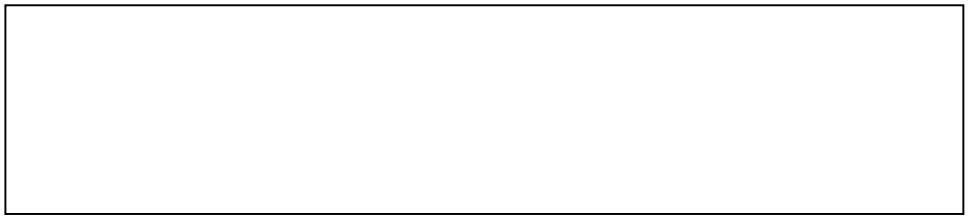


Figura 9 - Estructura del protocolo SOAP

El intercambio de mensajes se realiza mediante tecnología de componentes. El término Object en el nombre significa que se adhiere al paradigma de la programación orientada a objetos.

SOAP es un marco extensible y descentralizado que permite trabajar sobre múltiples pilas de protocolos de redes informáticas. Los procedimientos de llamadas remotas pueden ser modelados en la forma de varios mensajes SOAP interactuando entre sí. SOAP funciona sobre cualquier protocolo de Internet

Como ejemplo se muestra la forma en que un cliente solicitaría información de un producto a un proveedor de servicios Web:



Y esta sería la respuesta del proveedor:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
        <soap:Body>
            <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
              <getProductDetailsResult>
                <productName>Toptimate 3-Piece Set</productName>
                <productId>827635</productId>
                <description>3-Piece luggage set.  Black Polyester.</description>
                <price>96.50</price>
                <inStock>true</inStock>
              </getProductDetailsResult>
            </getProductDetailsResponse>
          </soap:Body>
</soap:Envelope>
```

Por otro lado, WSDL permite que un servicio y un cliente establezcan un acuerdo en lo que se refiere a los detalles de transporte de mensajes y su contenido, a través de un documento procesable por dispositivos. WSDL representa una especie de contrato entre el proveedor y el que solicita. WSDL especifica la sintaxis y los mecanismos de intercambio de mensajes, es decir, los métodos y objetos que se intercambian durante la comunicación con el Servicio Web. WSDL describe cuatro piezas principales:

- Información sobre la interfaz exportada
- Tipos de datos de los mensajes intercambiados (peticiones, respuestas, errores)
- Información de “binding” (enlace) definiendo como transportar los mensajes en la comunicación
- Direcciones para localizar los servicios

Los documentos WSDL definen los servicios como colecciones de puntos finales de red o puertos. En WSDL, la definición abstracta de puntos finales y de mensajes se separa de la instalación concreta de red o de los enlaces del formato de datos. Esto permite la reutilización de definiciones abstractas: mensajes, que son descripciones abstractas de los datos que se están intercambiando y tipos de puertos, que son colecciones abstractas de operaciones. Las especificaciones concretas del protocolo y del formato de datos para un tipo de puerto determinado constituyen un enlace reutilizable. Un puerto se define por la asociación de una dirección de red y un enlace reutilizable; una colección de puertos define un servicio. Por esta razón, un documento WSDL utiliza los siguientes elementos en la definición de servicios de red:

- **Types**: contenedor de definiciones del tipo de datos que utiliza algún sistema de tipos (por ejemplo XSD).
- **Message**: definición abstracta y escrita de los datos que se están comunicando.
- **Operation**: descripción abstracta de una acción admitida por el servicio.
- **Port Type**: conjunto abstracto de operaciones admitidas por uno o más puntos finales.
- **Binding**: especificación del protocolo y del formato de datos para un tipo de puerto determinado.
- **Port**: punto final único que se define como la combinación de un enlace y una dirección de red.
- **Service**: colección de puntos finales relacionados.

Ventajas de los Servicios Web

Podemos citar las siguientes:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.

- Los Servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar.

Inconvenientes de los Servicios Web

En este caso tenemos:

- Para realizar transacciones no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA.
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como RMI, CORBA, o DCOM. Es uno de los inconvenientes derivados de adoptar un formato basado en texto. Y es que entre los objetivos de XML no se encuentra la concisión ni la eficacia de procesamiento.
- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.
- Existe poca información de servicios web para algunos lenguajes de programación
- Saber que Servicio Web se encuentra disponible y para qué sirve no resulta sencillo. El mecanismo estándar para catalogar los Servicios Web es a través de un mecanismo denominado UDDI o Universal Description, Discovery and Integration. UDDI es una iniciativa industrial abierta y sirve como un registro de Servicios Web. Cada registro se compone de tres partes:
    - Páginas Blancas: Incluye la dirección, contacto y otros identificadores conocidos
    - Páginas Amarillas: Incluye la categorización industrial basada en taxonomías
    - Páginas Verdes: Información técnica sobre servicios que aportan las propias empresas

Cuando se consulta sobre un registro UDDI se envía un mensaje SOAP al mismo, el cual devuelve el documento WSDL asociado al Servicio Web que se quiere acceder. Toda esta labor debe de hacerse de manera manual o bien mediante la programación de agentes específicos para el Servicio Web que se desea invocar, lo que en la mayoría de las ocasiones implica un trabajo bastante pesado.

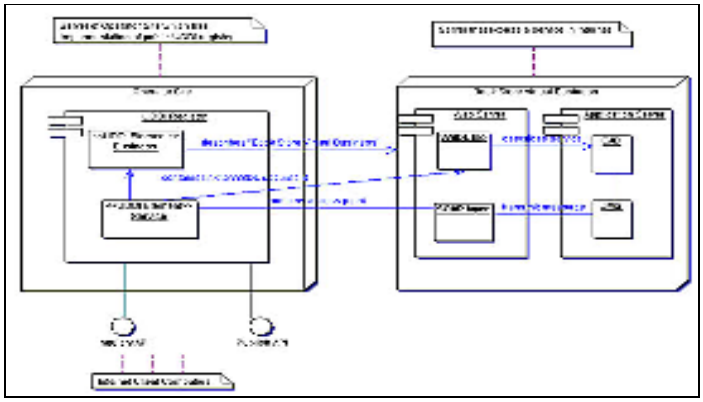


Figura 10 - Estructura de UDDI

2.8.2 Servicios Web Semánticos

Con los Servicios Web Semánticos estamos ante una nueva aproximación que consiste en la integración de la Web Semántica y los Servicios Web como tecnologías complementarias.

Los Servicios Web Semánticos proporcionan una nueva infraestructura para los Servicios Web al añadir la posibilidad de definir la semántica de los mismos. De este modo los problemas que hemos visto cara al descubrimiento de los Servicios Web y de las capacidades que presentan podrán resolverse de manera automática o semi-automática.

En el capítulo 5, veremos los denominados Lenguajes para la Modelización de Servicios Web Semánticos y nos centraremos en el estudio de los lenguajes WSMO y WSML, el cual es un ejemplo claro que permite añadir la semántica que falta mediante el uso de ontologías.

Algunas ventajas que obtendremos con el uso de esta tecnología:

- Buscadores basados en Web Semántica para Servicios Web: Por ejemplo, podremos realizar búsquedas basadas en restricciones complejas del tipo "encontrar el hotel más *bonito cercano a la playa"*
- Tendremos la posibilidad de proporcionar un Servicio Web que haga deducciones lógicas por mí, por ejemplo comprobar la validez de la descripción de un Servicio Web.

2.9 Inconvenientes de la Web Semántica

Con todo lo que hemos contado hasta ahora, puede parecer que solamente es cuestión de tiempo que la Web Semántica acabe imponiéndose, como lo hizo la Web actual, en el momento que se resuelvan algunas cuestiones pendientes. Sin embargo, la ealidad es mucho más dura (como pasa siempre) y hay mucha gente que tiene serias dudas sobre la viabilidad actual de la Web Semántica. Estas dudas surgen por varios motivos que vamos a ir viendo.

El primero motivo que se argumenta en contra de la Web Semántica, es uno relacionado con la necesidad de usar reglas de inferencia, vimos que las reglas de inferencia se han utilizado mucho en el área de la I.A, pero los sistemas basados en este tipo de reglas no han funcionado del todo bien salvo en el caso de dominios muy específicos y limitados.

Otro motivo que podemos encontrar, es que aunque resolviendo el anterior, no resulta sencillo crear una infraestructura que permita crear redes de confianza a unos costes asequibles, entendiendo redes de confianza a un conjunto de redes en las que intercambiar información de forma segura dejando que se gestione de manera casi automática. Además la arquitectura necesaria para montar esta infraestructura tendría una complejidad tan elevada que sería muy difícil de montar.

Hay otro motivo que se denomina **problema del "momento cero"**, que básicamente consiste en que nos encontramos con un círculo vicioso difícil de romper y que viene a ser algo del estilo siguiente: *"¿Por qué el médico tendría un sistema de Web Semántica si*

montarlo tiene un coste importante y casi ningún paciente lo usa? ¿Por qué los usuarios tendrían agentes, si casi ningún proveedor (médico) lo soporta?"

No obstante, y a pesar de que lo que acabamos de decir es cierto y estos problemas se encuentran ahí, realmente los inconvenientes mayores que tiene que resolver la Web Semántica son fundamentalmente dos:

- El denominado Coste de la Transición de la Web actual a la Web Semántica
- Y la consensuación de ontologías

2.9.1 Coste de la Transición

La transición de la web actual a la web semántica puede implicar un coste altísimo si tenemos en cuenta el volumen de contenidos que ya forman parte de la web. Crear y poblar ontologías supone un esfuerzo extra que puede resultar tedioso cuando se agregan nuevos contenidos, pero directamente prohibitivo por lo que respecta a integrar los miles de gigabytes de contenidos antiguos. Las estrategias más viables combinan una pequeña parte de trabajo manual con la automatización del resto del proceso. Las técnicas para la automatización incluyen, entre otras, el mapeo de la estructura de bases de datos a ontologías, el aprovechamiento, previa conversión, de los metadatos y estándares de clasificación presentes en la web (y fuera de ella), y la extracción automática de metadatos a partir de texto y recursos multimedia.

2.9.2 Consensuación de Ontologías

Definir ontologías de amplio uso es un proceso lento y difícil, así como consensuar ontologías en una comunidad por poco amplia que sea. Converger a una representación común es una labor más compleja de lo que puede parecer, ya que típicamente cada parte del sistema conlleva peculiaridades necesarias, y un punto de vista propio que a menudo necesitan incidir en la propia ontología. La representación del mundo no es neutra respecto al uso que se le va a dar: tanto un dietista como un biólogo tienen conocimiento sobre las plantas, pero su representación de esa materia es muy distinta, y probablemente no sería adecuado imponer la misma representación para ambas perspectivas. Las vías para salvar esta dificultad consisten en compartir ontologías para las áreas comunes en que puede tener lugar una interacción o intercambio de información entre las partes, y establecer formas de compatibilidad con las ontologías locales, mediante extensión y especialización de las ontologías genéricas, o por mapeo y exportación entre ontologías.

Todos estos problemas se han intentado de algún modo de camuflar hablando de la creación de una Web Semántica menos ambiciosa y que podría ser también muy útil. Por ejemplo, de la aplicación de algunas de las ideas vistas podríamos obtener beneficios tan importantes como:

- solo con la etiquetación, las organizaciones podrían utilizar enormes volúmenes de información no estructurada y semiestructurada disponible en la actualidad en Internet.
- realizar la etiquetación de información no estructurada permite realizar consultas precisas y procesamientos complejos que de otra manera no sería posible.
- hay tareas que aparentemente requieren de inteligencia pero que realmente se resuelven con una estandarización adecuada de comunicación entre aplicaciones.

Lo que nos deparará el futuro está, por tanto, aun por decidir. Pero hay algo que es un hecho y es que la Web Semántica se ha convertido en el área de investigación de moda en centros de la importancia de la Universidad de Stanford, el MIT, la Open University del Reino Unido, etc. y porqué no decirlo este TFC es otro ejemplo más.

Desde nuestro punto de vista y después de leer las opiniones de numerosos autores, creemos que la Web Semántica ha llegado y lo ha hecho para quedarse.

Capítulo 3

El Problema que modelizaremos

En este capítulo se describe en detalle el problema que debemos de modelar e implementar en los capítulos 4 y 5 del presente documento. Para ello haremos en primer lugar una descripción general del problema tal y como aparecía en el enunciado original, después explicaremos las modificaciones que hemos considerado incorporar al problema para enriquecerlo y adaptarlo al entorno tecnológico de estudio.

Según esto, abordaremos el problema exponiéndolo en los siguientes puntos:

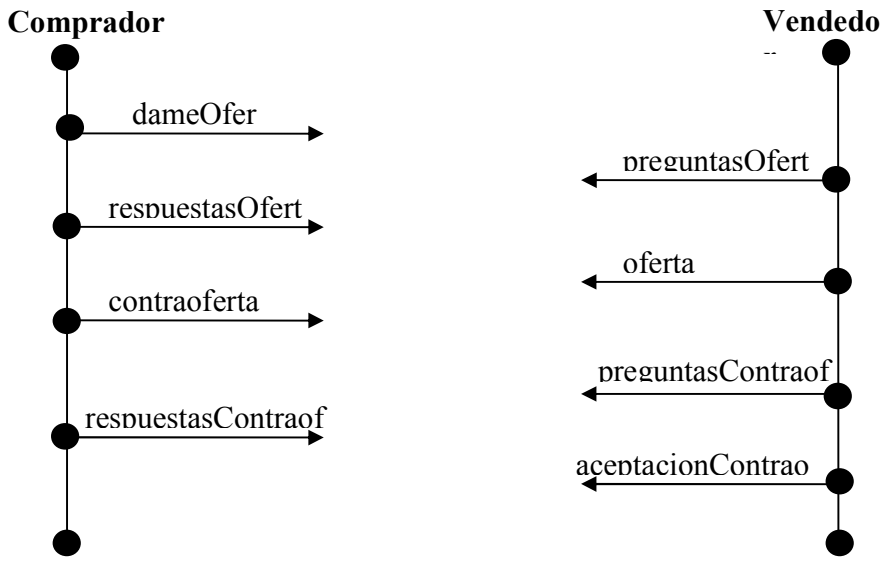
3.1. Descripción del Problema
3.2. Adaptaciones la Problema

3.1 Descripción del Problema

La Web Semántica es una extensión de la Web tradicional en la que los recursos tienen un significado concreto y preciso entendible por el ordenador. Dicho entendimiento se consigue gracias a la anotación de los recursos mediante el uso de ontologías. Los Servicios Web Semánticos combinan las ideas de la Web Semántica con la tecnología propia de los Servicios Web (UDDI, SOAP, WSDL) con el objetivo de de realizar tareas o ejecutar lógicas de negocio de forma semiautomática, usando una infraestructura existente y barata: Internet.

El objetivo del proyecto es estudiar el Lenguaje de Modelado de Procesos de Negocio y el Lenguaje para el Modelado de Servicios Web Semánticos, prestando especial atención a sus capacidades para representar coreografías (interacción entre dos servicios).

Como caso de estudio concreto se propone la descripción del modelo de negocio detallado en la Figura 1.



Las flechas representan mensajes con un nombre determinado. Como es natural los mensajes contienen información concreta. Por ejemplo, el número de artículos, el precio por artículo, etc. El contenido concreto de los mensajes se deja abierto a la imaginación del alumno. Cuanto mas completo sea el escenario mejor será la puntuación. Así mismo, se apreciará cualquier extensión al escenario propuesto aunque no se considera obligatorio.

3.2 Adaptaciones al Problema

Dado que en el enunciado no se especifica el ámbito del problema y se deja a criterio del alumno las adaptaciones oportunas, se ha decidido lo siguiente:

- El entorno de negocio en donde el sistema se está desarrollando es el sector de la compra-venta de vehículos, por tanto un “Comprador” es aquella entidad que tiene necesidad de comprar un vehículo, mientras que un “Vendedor” es aquella entidad que dispone de un catálogo de vehículos que desea vender a un posible “Comprador”. Por tanto, el objetivo consistirá en crear un dialogo (coreografía) entre un “Comprador” y un “Vendedor” a fin de ser capaces de realizar la compra de un vehículo que se adecue lo más posible a las necesidades del comprador. Hay que fijarse que se establecen una serie de criterios de valoración (heurísticas) dentro de cada uno de las entidades “Comprador” y “Vendedor”, por lo que puede ser o no posible realizar la compra en función de si las condiciones fijadas se cumplen durante el proceso.

- Un vehículo se caracteriza por la siguiente información:
  - Marca
  - Modelo
  - Año de fabricación
  - Matricula
  - Color
  - Número de puertas
  - Número de plazas
  - Tipo de Vehículo (berlina, todoterreno, pickup, etc.)
  - Precio
  - Descuento

- Precio mínimo de venta (55% del precio del vehículo)
- De oferta (si o no)
- Provincia de procedencia
- Accesorios (como puede ser aire acondicionado, climatizador, llantas de aleación, elevalunas, etc.)
- Tiempo en stock (en meses)

- El flujo de mensajes típico entre las entidades "Comprador" y "Vendedor" podría ser el siguiente:

1. El proceso se inicia cuando el "Comprador" decide que quiere comprar un vehículo y dispone de una cierta cantidad de dinero para hacerlo. El nombre de esta operación será "**dameOfertas**" y se situará en la entidad "Comprador", recibirá como parámetro la cantidad de dinero máxima que se puede gastar en la compra de un nuevo vehículo y las características del vehículo que se desea comprar, cada característica tendrá para el "Comprador" una puntuación de 0 a 10 indicando cuanto de importante es que el vehículo conste de dicha característica (0 nada importante – 10 muy importante)

2. El "Comprador" solicita al "Vendedor" que: le proporcione aquellos vehículos que se encuentren en oferta, por lo que el "Vendedor" le proporcionará la información de todos los vehículos en oferta.

   El nombre de esta operación sería "**preguntarOfertas**" y se establecería en la entidad "Vendedor", se le pasará las características del vehículo a comprar y si tiene que ser de oferta o no.

3. El "Comprador" de las respuestas recibidas tomará la decisión de elegir cuales son las que más le interesan, por defecto el criterio que se seguirá es el siguiente:
   - De los vehículos recibidas (sean o no de oferta) descartará aquellos cuyo precio supere la cantidad de dinero disponible para la compra
   - En caso de que el precio de los vehículos estén por debajo del precio de compra permitido, la preferencia del vehículo se realizará ordenándolos por la mayor puntuación obtenida como suma de las características que presente y que se adecuen a las preferencias del usuario, en caso de empate se ordenara por año de fabricación (del más nuevo al más viejo) y precio.
   - Finalmente si no se indican características, los vehículos se ordenaran por año de fabricación (del más nuevo al más viejo) y precio

   El nombre de la operación en este caso será "**respuestasOfertas"** y será parte de la entidad "Comprador".

4. El "Vendedor" de los modelos elegidos por parte del "Comprador" realizará una oferta de venta, que consistirá en aplicar sobre el precio de compra del vehículo un descuento entre un 1% a un 20%, en función del año de compra, de modo que si el coche tiene:
   a. más de 10 años se le aplicará un descuento del 20%,
   b. más de 5 años pero menos de 10 años, se le aplicará un 10%
   c. más de 3 años pero menos de 5 años, se le aplicará un 5%
   d. menos de 3 años, se le aplicará 1%

   El nombre de la operación será "**ofertas"** y se incluirá dentro del comportamiento de la entidad "Vendedora"

5. De las ofertas presentadas por la entidad “Vendedor”, el “Comprador” podrá realizar una contraoferta a las mismas, para realizar esta contraoferta se sigue el siguiente criterio:
    a. Si no se indicaron características del vehículo y la oferta presentada por el “Vendedor” es igual o menor a la cantidad de la que disponemos se contraoferta con un 5% menos del precio propuesto por el “Vendedor” (5% será el máximo valor permitido para contraofertar)
    b. Si se indicaron características del vehículo, se contraoferta en función de la puntuación obtenida, así a mayor puntuación menor contraoferta (ya que el coche vale más y será más difícil que el vendedor acepte), de este modo se establece el siguiente criterio:
        i. Vehículo con una puntuación mayor de 80, se contraoferta con un precio del 1% menos del precio propuesto por el “Vendedor”
        ii. Vehículo con una puntuación entre 60 y 80, se contraoferta con un precio del 3% menos del precio propuesto por el “Vendedor”
        iii. Vehículo con una puntuación menor de 60, se contraoferta con un precio del 5% menos del precio propuesto por el “Vendedor”

El nombre de la operación será “**contraofertas”** y estará formando parte de la entidad “Compradora”

6. De las contraofertas presentadas por el “Vendedor”, el “Comprador” decidirá cuales son las que acepta devolviendo el resultado en “**preguntasContraoferta”.** Las contraofertas que el “Vendedor” aceptará serán aquellas por aquellos vehículo que:
    a. Lleven más de 12 meses sin haberse vendido (tiempo en stock), siempre y cuando el precio no sea inferior al precio mínimo de venta y cuya antigüedad sea menor de tres años
    b. Lleven más de 24 meses sin haberse vendido (tiempo en stock) y cuya antigüedad sea mayor de 10 años.

7. El “Comprador” mediante la operación”**respuestaContraoferta”**, decidirá si acepta o no quedarse con alguno de los vehículos, en este caso se quedará con aquel vehículo que tenga el menor precio de compra y cuyas puntuación supere los 60 puntos (si hay alguno)

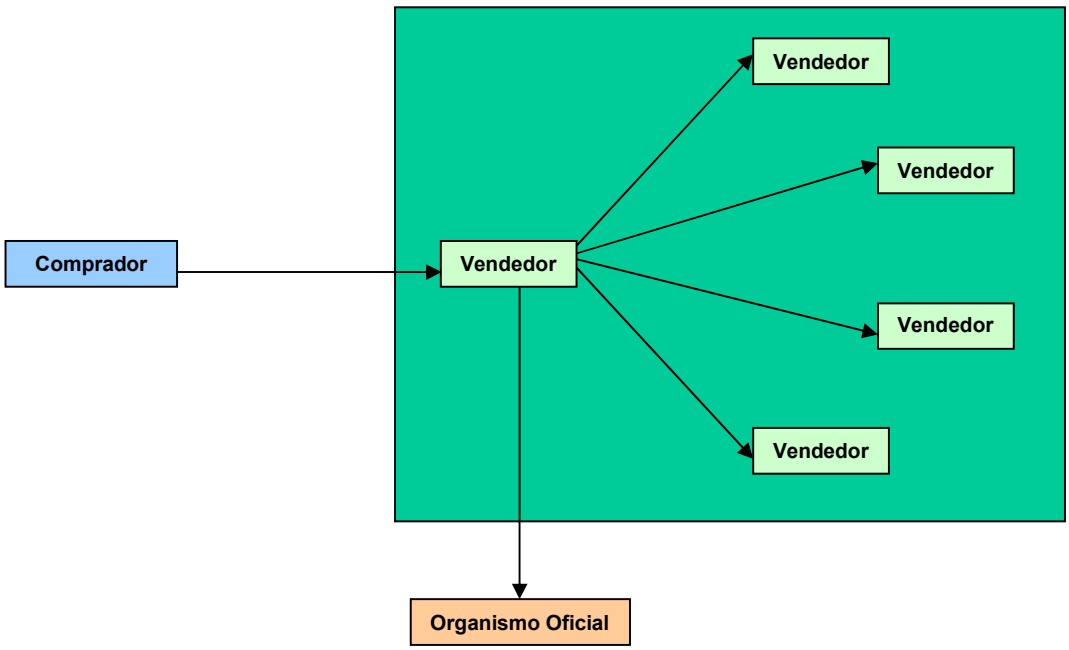
8. El “Vendedor” cierra el trato con el “Comprador” mediante la operación “**contrato**”

- Comportamientos adicionales a contemplar dentro del sistema
    - La entidad “Vendedor” forma parte de un “Consorcio de Vendedores de Coches”. El objetivo de este consorcio consiste en aprovechar las capacidades de venta de cada uno de los vendedores a fin de poder responder a las necesidades del comprador. La forma en la que trabaja este consorcio es la siguiente:
    - Si después de realizar el proceso de compra-venta descrito, no se produce un contrato entre las entidades “Vendedor” y “Comprador”, el vendedor solicita al comprador si desea buscar en otros vendedores del consorcio a fin de poder responder a su necesidad de compra.
    - En caso de que el “Comprador” acepte la sugerencia, será el “Vendedor” el que realice la consulta a los vendedores del consorcio a fin de ver si alguno dispone de un vehículo con las características demandadas por el comprador e iniciando el diálogo con ellos (ahora sin mediación del

"Comprador") y teniendo en cuenta la información que el "Comprador" proporciono al "Vendedor" como respuesta en el método "respuestaContraoferta".

- Por otro lado, en el caso supuesto de que se realice el contrato entre "Vendedor" y "Comprador", se hace necesario que la información de compra se pase a los organismos oficiales oportunos, a fin de que se proceda a la tramitación de la correspondiente documentación del vehículo y que deberá ser facilitada por el "Comprador" una vez que se ponga a disposición de éste el vehículo. La información que se proporcionará al Estado se corresponderá con los datos del vehículo y de la persona que lo adquiere. El Estado realizará las verificaciones oportunas y determinará si se puede o no dar de alta la documentación (por ejemplo el vehículo ha sido robado o el comprador tiene una deuda con Hacienda y está embargado)

El esquema global que presenta el enunciado a modelar se corresponde con el siguiente esquema:



Finalmente indicar, que si es necesario realizar alguna aclaración adicional al enunciado propuesto en esta sección, se indicará expresamente en el capitulo correspondiente en donde dicha aclaración tenga cabida.

Capítulo 4

Modelizando e Implementando Coreografías de Procesos

En este capítulo vamos a adentrarnos en la modelización e implementación del problema propuesto mediante la utilización de algún lenguaje que permita la construcción de un proceso de negocio desde alguno de los punto de vista posibles.

Existen dos formas diferentes de crear procesos de negocio en donde intervinienen Servicios Web, estas dos formas que denominamos, orquestación de servicios por un lado y coreografías de servicios por otro, permiten abordar la creación de los procesos de negocio desde un amplio espectro de posibilidades. Y es precisamente esta riqueza la que nos obliga a que tengamos que centrarnos en un caso concreto y debamos centrar con mucha precisión el alcance del proceso de modelado e implementación que vamos a desarrollar.

En este capítulo vamos a hacer, en primer lugar, una introducción a los Lenguajes de Procesos en general y veremos que características tienen, y las diferencias que existen entre coreografías y orquestación. Una vez explicadas estas ideas, centraremos nuestro estudio en la modelización e implementación de una coreografía de servicios utilizando un lenguaje en particular, denominado WS-CDL o Web Services Choreography Description Language. La elección de este lenguaje se ha hecho por ser el que tiene más posibilidades de convertirse en el estándar de facto para el desarrollo de este tipo de problemas.

Fijémonos en que hablamos de posibilidades de convertirse en estándar, y esto que puede parecer trivial no lo es en absoluto, ya que condiciona, como veremos más adelante, las herramientas disponibles para trabajar, así como la fiabilidad que dichas herramientas pueden aportan al usuario final.

Abordaremos la modelización e implementación en los siguientes puntos principales:

4.1 Orquestación y Coreografía de Servicios

4.1.1 Orquestación

Un proceso Web es de orquestación de servicios cuando es controlado totalmente por una única entidad. Ésta define completamente las interacciones con los servicios componentes, y la lógica requerida para conducir correctamente esas interacciones. Este tipo de proceso puede entenderse como privado y ejecutable: privado porque la definición de la lógica del proceso es hecha enteramente por un participante en la interacción; y ejecutable porque tiene un comportamiento de conversión de entradas en salidas y tiene efectos en el mundo real. En la figura se muestran dos procesos de orquestación de servicios. Se supone un escenario donde participan una "Entidad Compradora" y una "Entidad Proveedora". Cada una de ellas cuenta con un sistema ERP[26] para la gestión de recursos y utiliza servicios web para interactuar con su la otra parte.

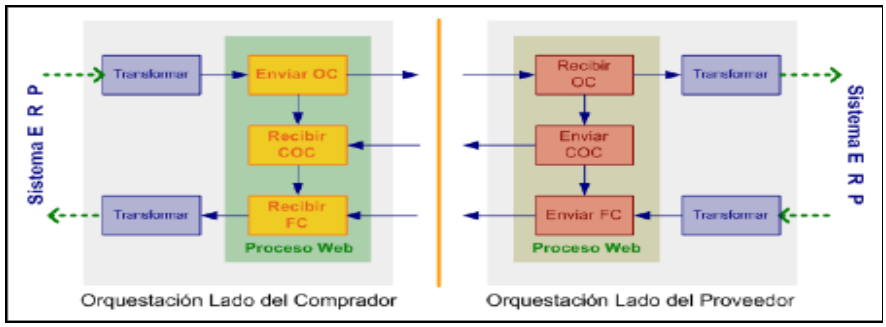


Figura 11 - Ejemplo de Orquestación de Servicios Web

En la Entidad Compradora el sistema ERP nota que el inventario de algún producto ha bajado y notifica al proceso de orquestación de servicios. Éste genera una orden de compra (OC), la envía a la Entidad Proveedora y queda a la espera de una confirmación de recepción de la orden (COC). Una vez recibida la confirmación, el proceso queda a la espera de la respuesta a la orden de compra, la cual es la Factura de Compra (FC), para ser retornada finalmente al sistema ERP. En el otro lado, el proceso de orquestación de la Entidad Proveedora se comunica con su propio sistema ERP para efectuar la compra. En este lado, el proceso inicia recibiendo una orden de compra, respondiendo con una confirmación de recepción al tiempo que envía la orden al sistema interno y quedando a la espera de la respuesta a la orden para ser reenviada a la Entidad Compradora. Nótese cómo cada organización participante en la interacción ajusta un proceso que usa servicios web para interactuar con la otra parte. Cada entidad implementa y controla su propio proceso al tiempo que ignora totalmente cómo está implementado el proceso en el otro extremo

Algunos ejemplos de sistemas que llevan implícito el uso de la orquestación lo tenemos por ejemplo en Microsoft BizTalk Server, que contiene un motor que se utiliza en la administración de procesos de negocio (BPM) y permite a los desarrolladores rápidamente orquestar complejos procesos de negocio que involucran sistemas muy diferentes. Otros ejemplos, en este caso, open-source[27] de motores BPEL son los siguientes:

- ActiveBPEL (http://www.activebpel.org)
- AgilaBPEL (http://swik.net/Agila-BPEL)

[26] ERP – Enterprise Resource Planning
[27] Existen en la actualidad infinidad de proyectos open-source dedicados a BPEL, hemos señalado los que a nuestro juicio son los más interesantes

- Apache ODE (http://incubator.apache.org/ode)
- BPEL Execution Engine (http://bexee.sourceforge.net)

4.1.2 Coreografía

Un proceso web es de coreografía de servicios cuando define las colaboraciones entre cualquier tipo de aplicaciones componentes, independientemente del lenguaje de programación o de la plataforma de soporte de cada una de ellas. Un proceso de coreografía no es controlado por uno solo de los participantes de la interacción.

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. En la figura anterior se ilustra la interacción entre dos entidades que soportan parte de su lógica de negocio con un proceso web, resaltando la parte privada de la interacción para cada una de ellas. Sin embargo en la figura siguiente se muestra la misma interacción, esta vez resaltando la parte pública de la interacción.

Como puede verse, la interacción entre estas dos entidades sigue un modelo: primero se envía la orden de compra en un sentido, luego se envía una confirmación a la orden y finalmente se envía una factura de compra, estas últimas en el otro sentido. Con base en este modelo y la representación de datos requerida para cada mensaje, cualquier otra entidad compradora podría comunicarse con la entidad proveedora. Dicho de otro modo, el orden en el flujo de la información se constituye en un contrato que dicta premisas a las entidades que deseen participar en una interacción de negocio.

Nótese que en la coreografía no es relevante la forma en la cual cada entidad implementa su participación, simplemente exige que se rija por ella.

Los lenguajes más interesantes que permiten la definición de coreografías son los siguientes:

- WSCDL (http://www.w3.org/TR/2004/WD-ws-cdl-10-20040427/): Web Service Choreography Description Language, se utiliza para describir colaboraciones punto a punto entre participantes, mediante la definición de un punto de vista global y del comportamiento común y complementario que puede observarse en el orden de intercambio de los mensajes entre los participantes a fin de conseguir cumplir un determinado objetivo de negocio común.
- WSCI (http://www.w3.org/TR/wsci/): Web Service Choreography Interface, describe como las operaciones de un servicio web, definidas a través de su documento WSDL pueden ser coreografiadas en el contexto del intercambio de mensajes en el cual los servicios web participan.
- IRS-III (http://kmi.open.ac.uk/projects/irs/): Internet Reasoning Service es un framework de trabajo para el desarrollo de aplicaciones basadas en el uso de Servicios Web Semánticos. Existen dos implementaciones la II y la III, la diferencia entre ambos es que la implementación II sigue el estándar del framework UPML desarrollado dentro del proyecto IBROW, mientras que la implementación III se basa en la anterior pero con el objetivo de crear servicios que cumplan con el estándar WSMO.

4.1.3 Orquestación vs. Coreografía

Dado que en ocasiones hablar de orquestación o coreografías parece indicar que se refiere a los mismos conceptos e ideas, ya que en ocasiones el matiz puede llegar a ser muy sutil

en función de lo purista que queramos ser, vamos a indicar una serie de ideas que pueden ser útiles cara a saber diferenciar que es cada cosa. Según esto tenemos:

- Orquestación:
  - Con la orquestación lo que realizamos es componer servicios para generar un nuevo proceso de negocio y puede usarse de dos formas distintas: para preparar la información que se intercambia en la ejecución de una coreografía o como la invocación siguiendo unas determinadas reglas a un servicio web.
  - Es un sistema jerárquico en el que la idea de proceso Web organiza la manera de invocar a otros mediante un flujo de control que determina qué invocar y cuando hacerlo.
  - La orquestación debe verse como algo atómico en si mismo, ya que lo que se ve es el proceso de negocio como tal.
- Coreografía:
  - Con la coreografía lo que realizamos es componer servicios para permitir la colaboración entre negocios, de forma en que se definan la forma en la que múltiples participantes colaboran entre pares (peer-to-peer), como parte de una operación o transacción de negocio duradera, que se realiza mediante el intercambio de mensajes tanto síncronos como asíncronos y que requiere una descripción formal de los protocolos de intercambio de mensajes y que debe estar visible para cada participante que interviene dentro de la coreografía.
  - Es un sistema entre pares, por tanto muchos participantes pueden colaborar, por tanto es posible la existencia de varios flujos de control diferentes.
  - Se describen interacciones con estado y duraderas.

4.2 Entorno de trabajo que utilizaremos

Como ya hemos indicado al principio en los objetivos del documento, para la elaboración de las distintas partes que componen este capítulo, nos hemos basado en el uso de herramientas de tipo Open Source. Concretamente en el caso de la modelización e implementación de coreografías las herramientas de trabajo que usaremos serán las siguientes:

- Eclipse 3.2 para la creación de los Servicios Web, el framework de trabajo que encapsula la implementación SOAP, las llamadas e invocaciones, etc, se hará mediante AXIS, aunque gracias a los asistentes que proporciona Eclipse este trabajo es completamente transparente para el desarrollador. En el Anexo de este documento se explica paso a paso, la secuencia de acciones que hay que seguir para crear un Servicio Web desde Eclipse, utilizando las herramientas que se habilitan desde este entorno.
- El plug-in para Eclipse de PI4SOA (http://www.pi4tech.com/xwiki/bin/view/Main/WebHome) que permite el diseño de coreografías mediante un editor gráfico, teóricamente también permite su ejecución y testeo, pero hemos sido incapaces de conseguir que esta funcionalidad este operativa. Por otro lado, el plug-in denominado “Choreography Model” solamente funciona para Eclipse 3.1 por lo que este entorno de desarrollo también será necesario.
- El repositorio de datos se encuentra en una base de datos MySQL 5.0, la base de datos contiene toda la información persistente que se maneja entre las aplicaciones, por simplicidad suponemos que todos los servicios atacan contra la misma instancia de base de datos.

- Finalmente para la creación de los diferentes diagramas en UML hemos utilizado la versión Comunity de la herramienta Visual Paradigm descargable desde http://www.visual-paradigm.com

4.3 Creando los Servicios Web

Para la creación de los diferentes servicios que necesitamos en nuestro sistema vamos a seguir los pasos que proporciona el asistente de creación de Servicios Web de Eclipse 3.2.

El modelo de programación que vamos a proponer se basa en un modelo estándar de tres capas, consistentes en una capa de presentación, una capa de negocio y una capa de acceso a datos. Los clientes que forman parte de la capa de presentación son los correspondientes a los que se generan mediante los asistentes de Eclipse 3.2 y que como hemos indicado se encuentra explicado la forma de utilizarlo en el Anexo, siendo la capa de negocio y de acceso a datos las que propiamente hay que codificar. El esquema que desarrollaremos es similar al que se muestra en la siguiente figura:

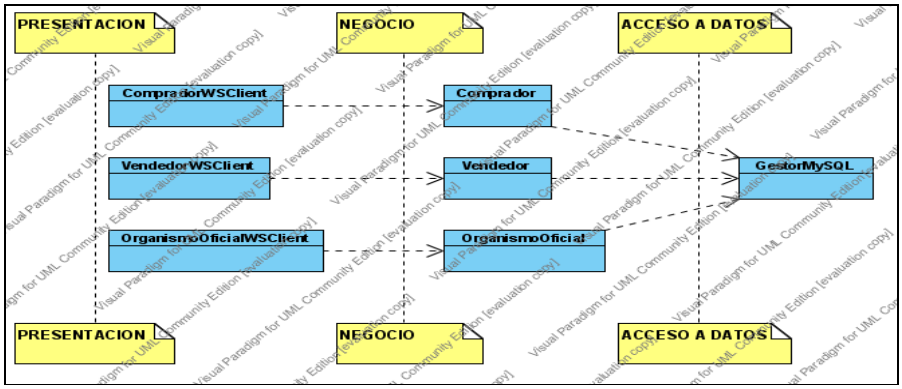


Figura 12 - Estructura de clases de implementación

La persistencia se ha montado utilizando el gestor de base de datos MySQL 5.0 y los drivers JDBC estándar que están disponibles para su descarga en la página Web http://www.mysql.com. Siendo el modelo de datos utilizado el formado por las entidades de base de datos siguientes:

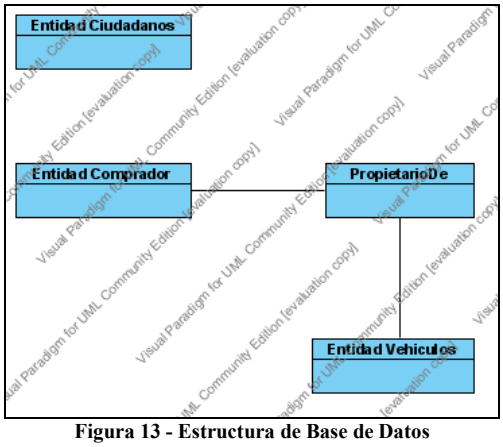


Figura 13 - Estructura de Base de Datos

En donde:

- **Entidad Ciudadanos**: En esta tabla se almacenan los ciudadanos que tienen problemas con el Estado, se utiliza para consultar que el NIF del Comprador no está dado de alta en ella
- **Entidad Comprador**: Contiene los datos de los Compradores que a través del Servicio Web "Comprador" realizan peticiones de ofertas de vehículos.
- **Entidad Propietario De**: En esta entidad se establece la relación de compra cuando un Comprador formaliza un contrato con el Vendedor de un vehículo de oferta. Se utiliza también para almacenar información sobre la documentación generada por el Organismo Oficial.
- **Entidad Vehículos**: Contiene los datos de los vehículos que se encuentran de oferta.

El acceso a datos se ha implementado mediante la clase "GestorMySQL", esta clase está desarrollada siguiendo el patrón Singleton por simplicidad y para permitir su utilización del modo más simple posible.

La estructura de paquetes que vamos utilizar para agrupar todas las clases de cada uno de los servicios es siempre homogénea y sigue la siguiente nomenclatura:

- edu.uoc.tfc.websemantica.beans: En este paquete almacenaremos todas las clases que actúan como Java beans y que son utilizadas por las clases de negocio.
- edu.uoc.tfc.websemantica.negocio: En este paquete se encuentran las clases propiamente dichas de negocio de nuestros servicios: "Comprador", "Vendedor" y "OrganismoOficial"
- edu.uoc.tfc.websemantica.ad: Este paquete se encuentra la clase de acceso a datos definida "GestorMySQL".

Gráficamente la relación entre paquetes es la siguiente:

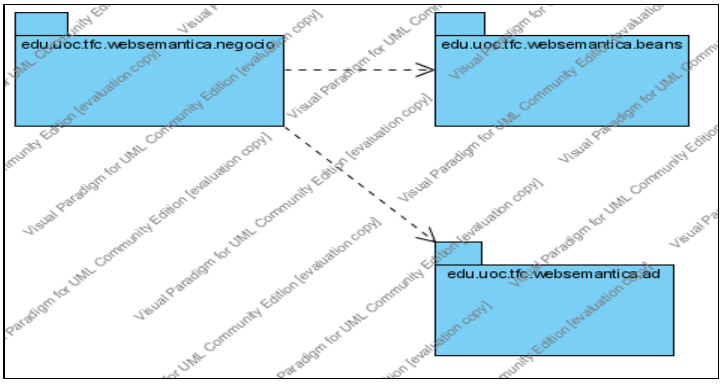


Figura 14 - Paquetes de implementación

4.3.1 Creación de la Entidad Comprador

El servicio asociado al Comprador tiene como misión realizar las operaciones necesarias para asegurar que se consigue la mejor oferta posible de entre los vehículos disponibles por parte del Vendedor estableciendo un dialogo de ofertas-contraofertas en donde el Comprador valora las ofertas que el Vendedor le ofrece en función de una serie de parámetros que se encuentran determinados en el enunciado propuesto.

El diagrama de clases que permite modelizar el problema es el que presentamos a continuación, en donde vemos que el método que inicia el diálogo "dameOfertas" recibe como parámetros el precio máximo que el Comprador puede permitirse así como los datos

del mismo junto con las características que le interesan de un vehículo y la valoración que de estas hace.

De este modo, en los sucesivas operaciones, el Comprador realizará la valoración de las ofertas que el Vendedor le realiza, quedándose con aquella que se ajuste mejor a sus necesidades.

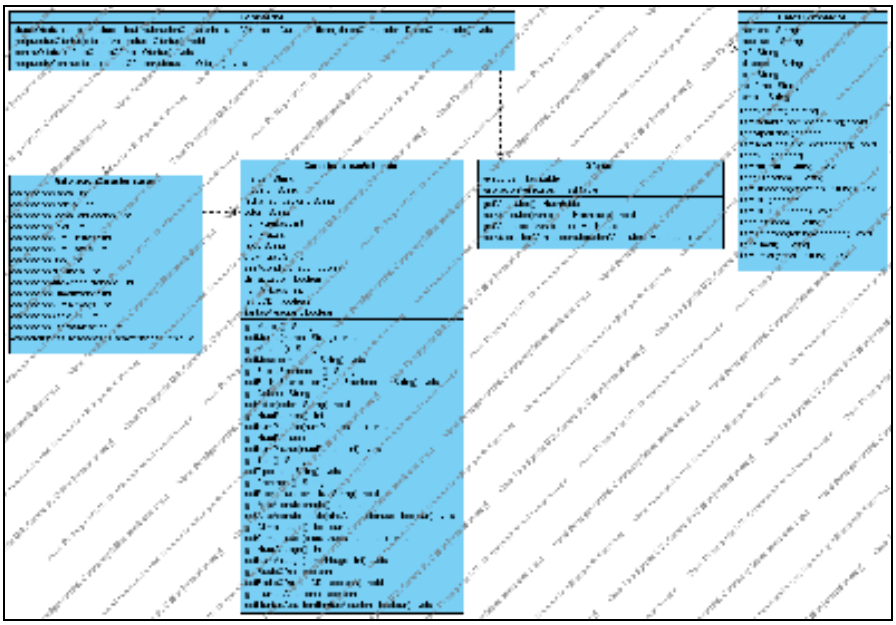


Figura 15 - Diagrama de Clases Entidad Comprador

4.3.2 Creación de la Entidad Vendedor

El servicio asociado al Vendedor tiene como misión proporcionar al usuario diferentes ofertas de vehículos que se encuentren de oferta y que en la medida de lo posible se ajuste a las preferencias del Comprador y que han sido proporcionadas por éste cuando inicio el diálogo.

La clase fundamental que usa el Vendedor para realizar su cometido es la clase “Oferta”, la cual consiste en una “Hashtable” de objetos “Vehículo” con los datos de los vehículos ofrecidos, así como de otra hash con las valoraciones de los mismos.

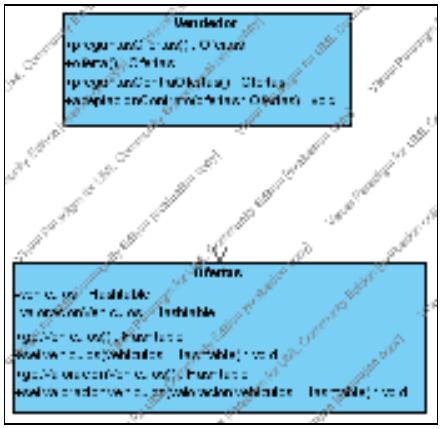


Figura 16 - Diagrama de Clases Entidad Vendedor

4.3.3 Creación de la Entidad Organismo Oficial

El servicio asociado al Organismo Oficial tiene como misión verificar si el vehículo que adquiere una determinada persona se encuentra y la propia persona, se encuentran en orden referente a su situación con el Estado. Cuando decimos en orden con el Estado, nos referimos a que el vehículo no se sea un vehículo robado y que esté al corriente del pago de todos los impuestos derivados de su uso, así como que el comprador también se encuentre al día de sus obligaciones con el Estado. En el caso de que todo esté correctamente se procederá a la expedición de la documentación del vehículo.

Teniendo, por tanto, en cuenta estas indicaciones, para modelizar este comportamiento tendremos que hacer uso del siguiente diagrama de clases:

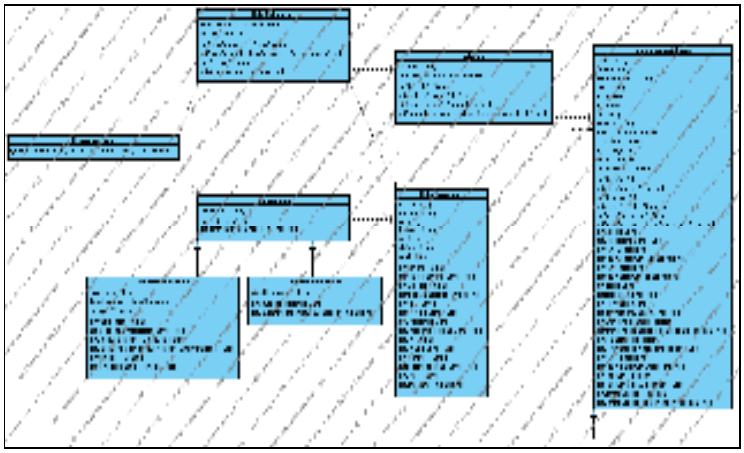


Figura 17 - Diagrama de Clases Entidad Organismo Oficial

Como vemos por el diagrama anterior, la clase que actúa como fachada para el acceso a la funcionalidad del Organismo Oficial, es la clase “OrganismoOficial”, la cual dispone de un método denominado “generarDocumentación” que recibe como parámetros los datos del comprador y del vehículo para los cuales queremos generar la documentación. El método realizará las consultas oportunas y en función de si es posible generar la documentación creará el objeto de “DocumentacionValida” o “DocumentacionInvalida” según el caso.

En Eclipse 3.2 el proyecto que encapsula el Servicio Web es el que hemos denominado como “OrganismoOficialWSProject” y el cliente que generamos para probar el servicio se denomina “OrganismoOficialWSProjectClient”.

En la siguiente figura se ve en marcha la aplicación cliente en donde se nos solicita que introduzcamos los datos necesarios para realizar la invocación a nuestro servicio web “Organismo Oficial”.

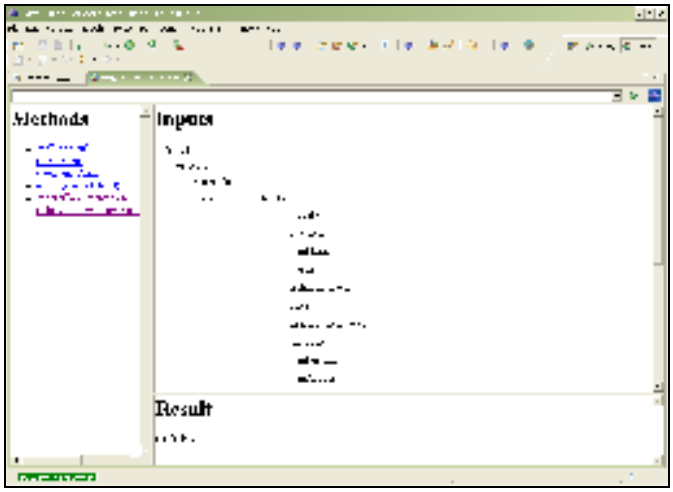


Figura 18 - Organismo Oficial en funcionamiento (I)

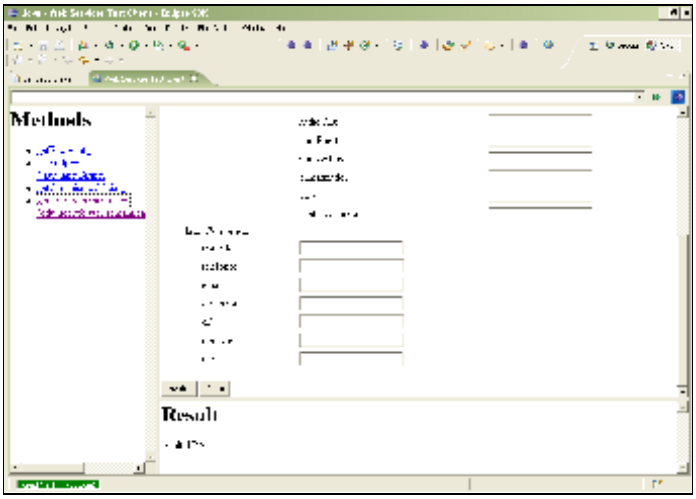


Figura 19 - Organismo Oficial en funcionamiento (II)

4.4 El Lenguaje WS-CDL

El lenguaje WS-CDL o Web Services Choreography Description Language es un lenguaje basado en XML que describe colaboraciones punto a punto entre participantes, definiendo, desde un punto de vista global, su comportamiento común y complementario, y en donde los mensajes intercambiados en un determinado orden resultan en la consecución de un objetivo de negocio común.

La especificación de los Servicios Web ofrece un Puente de comunicación entre entornos computacionales heterogéneos usados para desarrollar y albergar aplicaciones. El futuro de las aplicaciones de e-business requiere de la habilidad de mejorar el tiempo de vida, entre las colaboraciones punto a punto entre los participantes de los servicios, tanto dentro como a través de determinados dominios de confianza de las organizaciones

Mediante las Coreografías de Servicios Web se intenta conseguir un mecanismo de composición interoperable punto a punto entre cualquier tipo de participante, plataforma o modelo de programación usado en la implementación de los servicios

4.4.1 Propósito de WS-CDL

Los negocios así como otras actividades que involucran diferentes organizaciones o procesos independientes están enlazados de una manera colaborativa de manera que juntas se consigue un determinado objetivo de negocio, como puede ser completar una orden de venta.

Para que la colaboración funcione exitosamente, las reglas del contrato entre todos los participantes que interactúan, deben de quedar establecidas y deben de proporcionarse a priori. La especificación WS-CDL se dirige a ser el mecanismo que permita describir de manera precisa estas reglas de colaboración entre cualquier tipo de participante independientemente de la plataforma o el modelo de programación utilizado.

Usando la especificación WS-CDL, el contrato contiene una definición global del orden de las condiciones comunes y las restricciones bajo las cuales los mensajes se intercambian, lo que produce, que desde un punto de vista global se puedan observar el comportamiento común de todos los participantes involucrados. De este modo cada participante puede usar esta definición global para construir y probar soluciones acordes con el.

La ventaja de un contrato basado en un punto de vista global, tiene como ventaja que todos los sistemas de negocio individuales se encuentran bajo lo que se denomina un “dominio de control” en donde cada sistema intercambia información con los otros, esto significa que mientras el comportamiento observable que se encuentra dentro del dominio de control no cambia, la interoperabilidad se encuentra garantizada.

Por otro lado, en escenarios reales de aplicación es común que suceda que las entidades corporativas este poco dispuestas a delegar el control de sus procesos de negocio a otras entidades con las que se integran, las coreografías ofrecen un medio en el cual las reglas de participación dentro de la colaboración que definen pueden ser definidas claramente y aceptadas de manera conjunta por las partes. Cada entidad puede entonces implementar su porción de la coreografía como ha quedado determinado por el punto de vista global y común del que hablamos anteriormente. WS-CDL intenta proporcionar un mecanismo adecuada para establecer estas ideas de una manera que resulte lo más sencilla posible.

En la siguiente figura podemos ver una posible aplicación de WS-CDL dentro de un escenario típico.

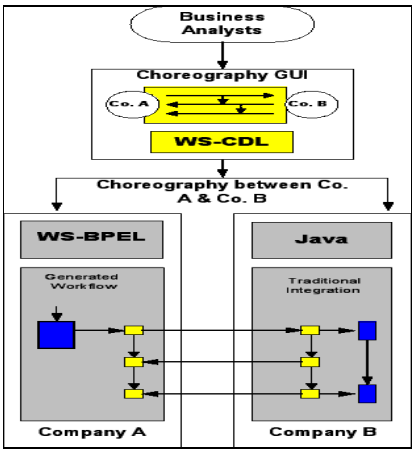


Figura 20 - Ejemplo de aplicación de WS-CDL

En la figura podemos ver dos empresas, la empresa A y la empresa B (Company A y B respectivamente) que desean integrar sus aplicaciones basadas en Servicios Web. Los respectivos analistas de ambas compañías convienen en cuales son los servicios implicados en la colaboración, cuales serán sus interacciones y las reglas y restricciones comunes bajo las cuales las interacciones deben de producirse. Una vez que estos términos han quedado clarificados se procede a la generación del documento WS-CDL que represente estos acuerdos, en el ejemplo la coreografía especifica la interacción entre los servicios a través de las entidades de negocio asegurando la interoperabilidad, mientras que deja las decisiones de implementación en manos de cada empresa. De este modo vemos que la empresa A utilizará una solución basada en BPEL para implementar su parte de la coreografía mientras que la empresa B, dado que dispone de gran cantidad de sistemas legados utilizara una aproximación basada en .NET o J2EE para implementar su parte de la coreografía.

4.4.2 Relación de WS-CDL con los Lenguajes de Procesos de Negocio

Muy importante, y que debe de quedar claro cuando se hace un estudio de WS-CDL es que éste lenguaje no puede ser considerado como un lenguaje de descripción de procesos de negocio o un lenguaje de implementación.

WS-CDL no depende de ninguna implementación de lenguaje de proceso de negocio. Así, puede ser utilizado para especificar colaboraciones interoperable entre cualquier tipo de participante sin importar la plataforma de soporte o el modelo de programación usado por la puesta en práctica del entorno. WS-CDL puede acoplarse con otros lenguajes tales como los que agreguen otras definiciones semánticas computables.

Cada participante adherido a representación de colaboración WS-CDL, puede ser implementado usando mecanismos completamente diferentes tales como:

- Aplicaciones cuya implementación esta basada en lenguajes de modelización de procesos de negocio como XLANG, WSFL, WSBPEL, BPML o XPDL
- Aplicaciones cuya implementación esta basada en lenguajes de programación de propósito general como Java o C#
- O agentes de software controlados por humanos.

4.4.3 Descripción de los elementos de WS-CDL

Hemos indicado que WS-CDL describe colaboraciones interoperables, punto a punto entre participantes. A fin de facilitar estas colaboraciones, los servicios confían en sus responsabilidades mutuas estableciendo relaciones formales. Las colaboraciones tienen lugar en un entorno común convenido, en donde hay definidas una serie de reglas y restricciones que deben de seguirse en el intercambio de la información.

El modelo WS-CDL se compone de los siguientes elementos:

- **roleType, relationshipType y participantType**: Dentro de una coreografía, la información es siempre intercambiada entre participantes dentro o a través de limites de confianza. Todas las interacciones ocurren entre roles que son exhibidos por los participantes y que son restringidos por las relaciones. Dentro de WS-CDL, un participante es modelado de manera abstracta a través de:
    - del denominado "participantType" o tipo de participante. Este elemento agrupa de manera conjunta aquellas partes del comportamiento observable que debe ser implementado por la misma entidad lógica o organización abstracta.
    - el papel que juega dentro de la coreografía o "roleType". Enumera el comportamiento observable potencial que un "participantType" puede exhibir en una interacción.
    - y el tipo de relación que establece o "relationshipType". Identifica la confianza mutua que debe ser satisfecha para que la colaboración sea exitosa.
- **informationType, variable y token**: Una variable contiene información sobre objetos comúnmente observables dentro de la colaboración, tales como la información intercambiada o la información observable de los "roleTypes" involucrados. Un "token" es un alias que puede ser usado para referencia partes de una variable. Las variables de intercambio de información, las variables de captura de estados y los tokens tienen un "informationType" que define el tipo de información que la variable contiene o el token referencia.
- **choreography**: El elemento "choreography" define una colaboración entre "participantTypes" que interactúan en la coreografía. Existen varios tipos:
    - **choreography life-line**: Sirve para expresar la progresión de la colaboración. Inicialmente la colaboración se establece entre los participantes, entonces el trabajo se realiza y termina (bien o mal).
    - **choreography exception block:** Un bloque de excepción especifica que acciones adicionales deberían ocurrir cuando la coreografía se comporte de una forma anormal.
    - **choreography finalizer block:** Un bloque de finalización especifica acciones adicionales que deberían de ocurrir para modificar el efecto de una coreografía finalizada exitosamente.
- **channelType:** Un canal realiza un punto de colaboración entre un "participantTypes" especificando donde y como la información es intercambiado. Dentro de WS-CDL, los canales son modelados abstractamente mediante "channelTypes".
- **workunit:** A workunit indica las restricciones que deben de ser completadas para realizar el progreso y así realizar el trabajo dentro de la coreografía.

- **activities y ordering structures:** Las actividades (activities) describen las acciones a realizar dentro de la coreografía. Las estructuras ordenadas (ordering structures) combinan actividades con otras estructuras ordenadas en una estructura enlazada para expresar el orden de las reglas de las acciones a realizar dentro de la coreografía.
- **interaction activity:** Una interacción es el bloque de construcción básico de una coreografía. Da lugar a un intercambio de información entre los participantes y a la posible sincronización de cambios observable en dicha información.
- **semantics:** La semántica permite la creación de descripciones que pueden almacenar la definición semántica de cada componente del modelo.

4.4.4 Estructura de un documento WS-CDL

Un documento WS-CDL es simplemente un conjunto de definiciones. Cada definición es una construcción nominal que puede ser referenciada. El elemento raíz se denomina "package" y dentro de él se incluyen las definiciones de cada coreografía individual.

La sintaxis de la construcción "package" es la siguiente:

```
<package
  name="NCName"
  author="xsd:string"?
  version="xsd:string"?
  targetNamespace="uri"
  xmlns="http://www.w3.org/2005/10/cdl">

  <informationType/>*
  <token/>*
  <tokenLocator/>*
  <roleType/>*
  <relationshipType/>*
  <participantType/>*
  <channelType/>*

  Choreography-Notation*
</package>
```

Los atributos de alto nivel "name", "autor" y "versión" definen propiedades de autoria del documento de coreografía. El atributo "targetNamespace" proporciona el espacio de nombre asociado con todas las definiciones de tipos contenidas en el paquete de coreografía.

Los elementos "informationTypes", "token", "tokenLocator", "roleType", "relationshipType", "participantType" y "channelType" pueden ser usados como elementos para todas las coreografías definidas dentro del paquete de coreografía.

A continuación se muestran las reglas de sintaxis de cada uno de los elementos anteriormente descritos, por brevedad solamente indicamos las etiquetas principales que forman parte de WS-CDL, en caso de necesitar de una referencia completa de todas las posibilidades del lenguaje puede consultarse en la siguiente dirección web: http://www.w3.org/TR/ws-cdl-10/

Etiqueta "<roleType>"

A "roleType" enumera el potencial comportamiento observable que un participante puede exhibir a fin de interactuar. Por ejemplo, el roleType "Comprador" está asociado con la compra de bienes y servicios y el roleType "Proveedor" esta asociado con proporcionar estos bienes y servicios a cambio de una cantidad de dinero. La sintaxis de "roleType" es la siguiente:

```
<roleType name="NCName">
  <behavior name="NCName" interface="QName"? />+
```

```
</roleType>
```

Etiqueta “<relationshipType>”

A “relationshipType” identifica el “roleType” y los “behavior” que se establecen entre dos participantes dentro de una coreografía. La sintaxis es la siguiente:

```
<relationshipType name="NCName">
  <roleType typeRef="QName" behavior="list of NCName"? />
  <roleType typeRef="QName" behavior="list of NCName"? />
</relationshipType>
```

Etiqueta “<participantType>”

“participantType” agrupa junto partes de un comportamiento observable que debe ser implementado por un participante. La sintaxis es la siguiente:

```
<participantType name="NCName">
  <roleType typeRef="QName" />+
</participantType>
```

Etiqueta “<channelType>”

Los “channelType” indica un punto de colaboración entre “participantTypes” indicando donde y como la información es intercambiada. La sintaxis de los “channelType” es la siguiente:

```
<channelType  name="NCName"
          usage="once"|"distinct"|"shared"?
          action="request-respond"|"request"|"respond"? >
  <passing  channel="QName"
         action="request-respond"|"request"|"respond"?
         new="true"|"false"? />*
  <roleType typeRef="QName"  behavior="NCName"? />
  <reference>
    <token name="QName"/>
  </reference>
  <identity usage="primary"|"alternate"|"derived"|"association">
    <token name="QName"/>+
  <identity>*
</channelType>
```

Etiqueta “<informationType>”

Describe el tipo de información usada dentro de la coreografía, gracias a esta abstracción la definición de la coreografía evita directamente referenciar tipos de datos como se definen en un documento WSDL o en un XMLSchema. La sintaxis es la siguiente:

```
<informationType name="NCName"
        type="QName"?|element="QName"? />
```

El atributo “name” se utiliza para asignar un nombre único al “informationType”. El siguiente es un ejemplo que indica como se usa “informationType” para referenciar a un mensaje WSDL de nombre “pns:purchaseOrderMessage”.

```
<informationType name="purchaseOrder"
                    type="pns:purchaseOrderMessage"/>
```

Etiqueta “<variables>”

Las variables capturan información sobre los objetos de la coreografía por ejemplo: variables de captura de información intercambiada, variables de captura de estados

conteniendo información acerca de los cambios observables en un “roleType” como resultado de la información que esta siendo intercambiada, variables de captura de canales que contienen información tales como URLs y variables de captura de excepciones con información sobre situaciones anómalas.

Una variable se define siguiendo la siguiente sintaxis:

```
<variableDefinitions>
  <variable   name="NCName"
              InformationType="QName"?|channelType="QName"?
              mutable="true|false"?
              free="true|false"?
              silent="true|false"?
              roleTypes="list of QName"? />+
</variableDefinitions>
```

Etiqueta “<token>”

Los “token” son alias para piezas de datos dentro de una variable o mensaje que necesita usarse dentro de una coreografía. Los “tokens” son diferentes de las “variables” porque estas contienen valores que pueden ser publicados como resultado de acciones o eventos dentro de la linea de vida (life-line) de la coreografía. La sintaxis para definir un token es la siguiente:

```
<token  name="NCName"  informationType="QName" />
```

Etiqueta “<tokenLocator>”

Los usaremos para identificar fragmentos de documento que se asociaran a los “token”. La sintaxis de un “tokenLocator” es la siguiente:

```
<tokenLocator  tokenName="QName"
          informationType="QName"
          part="NCName"?
          query="XPath-expression" />
```

Etiqueta “<choreography>”

Esta etiqueta define reglas comunes que pueden ser reutilizadas. Sigue la siguiente sintaxis:

```
<choreography  name="NCName"
    complete="xsd:boolean XPath-expression"?
    isolation="true"|"false"?
    root="true"|"false"?
    coordination="true"|"false"? >
  <relationship  type="QName" />
  <exceptionBlock  name="NCName">
  </exceptionBlock>?
  <finalizerBlock  name="NCName">
  </finalizerBlock>*
</choreography>
```

4.5 Diseño de Coreografías de Servicios

Una vez que ya tenemos claro cuales son las características que se persiguen con WS-CDL, y conocemos los elementos constituyentes del lenguaje vamos a ver como definiríamos la coreografía para los servicios expuestos anteriormente.
Lo primero que debemos de hacer es entrar dentro de Eclipse 3.1, el cual debemos de tener actualizado con el plugin de “Choreography Model” que podemos descargar desde: http://www.pi4tech.com/xwiki/bin/view/Main/WebHome.

Con el entorno preparado para trabajar, la siguiente acción a realizar es crear un nuevo proyecto al que llamaremos “ProyectoCoreoCompraVenta” y en donde crearemos la coreografía completa para nuestros servicios. La forma en que creamos el proyecto es mediante “File > New > Project” y en el cuadro de dialogo que aparece seleccionamos “Simple > Project”, introducimos la ruta en donde guardaremos nuestros ficheros y el nombre del proyecto que hemos elegido.

El plugin “Choreography Model” se integra completamente dentro de Eclipse y dispone de un editor gráfico que permite la creación de los diferentes elementos de nuestra coreografía. Para crear un fichero de coreografía simplemente debemos seleccionar en el menú “File > New > Other”, en el cuadro de diálogo que aparece a continuación desplegar la carpeta SOA y seleccionar el asistente “Choreography Description”.

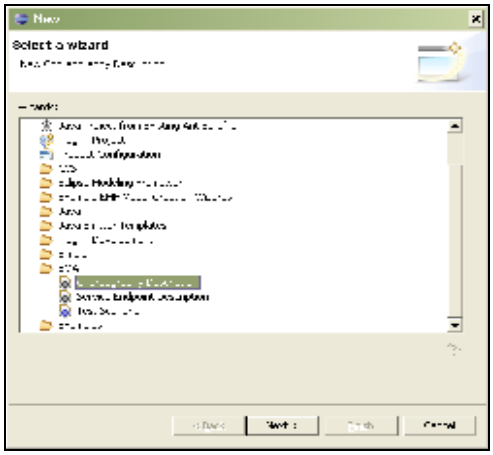


Figura 21 - Creación de un Choreography Description

Hacemos clic en “Next” y ponemos nombre a nuestra coreografía, por ejemplo “CoreCompraVenta”, se nos creará dentro de nuestro proyecto un fichero denominado “CoreoCompraVenta.cdm” que es con el que trabajaremos a partir de ahora, en un entorno que presenta la siguiente apariencia:

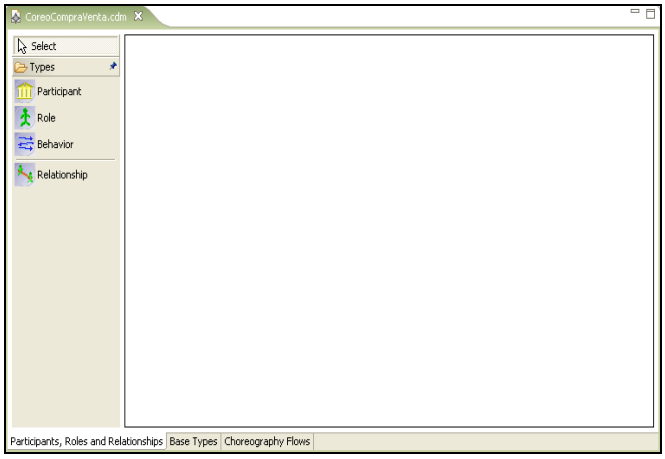


Figura 22 - Entorno de creación de coreografías

Los elementos básicos que formarán parte de nuestra coreografía son los definidos por:

- Entidad “Comprador” dispone de los métodos “dameOferta”, “respuestasOfertas”, “contraOfertas” y “respuestasContraofertas”.

- Entidad “Vendedor” dispone de los métodos “preguntarOfertas”, “ofertas”, “preguntasContraOfertas” y “aceptacionContrato”.
- Entidad “Organismo Oficial” dispone del método “generarDocumentación”

La coreografía además quedará descrita por las siguientes propiedades:

- **Autor**: Santiago Márquez Solís
- **Descripción**: Coreografía que sirve para la compra-venta de vehiculos de oferta disponibles
- **Nombre**: CoreoCompraVenta
- **Namespace**: http://www.uoc.edu/tfc/webSemantica/coreografia/compraventa#
- **Versión:** 1.0.0

Por tanto, teniendo esto, nuestra coreografía constará de tres participantes diferentes, “Comprador”, “Vendedor” y “Organismo Oficial”. Lógicamente, los roles que cada uno de estos participantes tendrá dentro de la coreografía depende de su función en el sistema, por tanto, el “Comprador” tendrá el rol de compra de vehículos, el “Vendedor” el rol de venta de vehículos y el “Organismo Oficial” el rol de generación de documentación. Los cuales a su vez van asociados a comportamientos (behaviour) específicos de cada participante y que hemos denominado:

- “ComportamientoVenta”,
- “ComportamientoCompra”
- “ComportamientoGenerar”

Asociados respectivamente con “Vendedor”, “Comprador” y “OrganismoOficial”. Todas estas ideas llevadas a WS-CDL dan lugar al siguiente fichero XML:

```
<?xml version="1.0" encoding="Cp1252"?>
<org.pi4soa.cdl:Package xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:org.pi4soa.cdl="http:///org/pi4soa/cdl.ecore" description="Coreografía que sirve para la compra-
venta de vehiculos de oferta disponibles" name="CoreoCompraVenta" author="Santiago Marquez Solis"
version="1.0.0" targetNamespace="http://www.uoc.edu">
 <typeDefinitions>
  <roleTypes name="VentaDeVehiculos">
   <behaviors name="ComportamientoVenta"/>
  </roleTypes>
  <roleTypes name="CompraDeVehiculos">
   <behaviors name="ComportamientoCompra"/>
  </roleTypes>
  <roleTypes name="GenerarDocumentos">
   <behaviors name="ComportamientoGenerar"/>
  </roleTypes>
  <relationshipTypes name="VentaDocumentacion"
      firstRoleType="//@typeDefinitions/@roleTypes.2"
      firstRoleTypeBehaviors="//@typeDefinitions/@roleTypes.2/@behaviors.0"
      secondRoleType="//@typeDefinitions/@roleTypes.0"
      secondRoleTypeBehaviors="//@typeDefinitions/@roleTypes.0/@behaviors.0"/>
  <relationshipTypes name="CompraVenta"
      firstRoleType="//@typeDefinitions/@roleTypes.0"
      firstRoleTypeBehaviors="//@typeDefinitions/@roleTypes.0/@behaviors.0"
      secondRoleType="//@typeDefinitions/@roleTypes.1"
       secondRoleTypeBehaviors="//@typeDefinitions/@roleTypes.1/@behaviors.0"/>
<participantTypes name="Comprador" roleTypes="//@typeDefinitions/@roleTypes.1"/>
  <participantTypes name="Vendedor" roleTypes="//@typeDefinitions/@roleTypes.0"/>
  <participantTypes name="OrganismoOficial"
                    roleTypes="//@typeDefinitions/@roleTypes.2"/>
 </typeDefinitions>
</org.pi4soa.cdl:Package>
```

Fijémonos que el elemento “package” que actúa como root en WS-CDL ha sido substituido por un elemento “package” propio de la herramienta que estamos utilizando, pero que a efectos prácticos no tiene mayor inconveniente.

El fichero anteriormente descrito tiene asociado la siguiente representación gráfica:

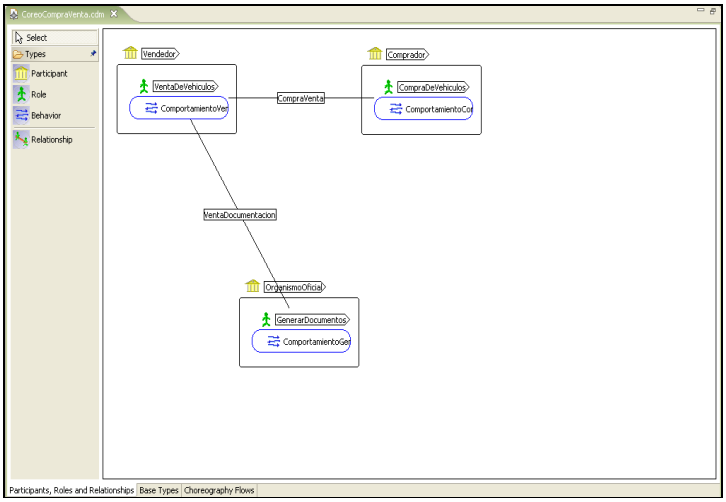


Figura 23 - Representación gráfica de la coreografía

Una vez tenemos los elementos que forman parte de la coreografía definidos, podemos pasar a crear la coreografía siguiendo el esquema del enunciado del problema propuesto y que viene a ser algo del estilo siguiente:

```
<choreographies name="Inicio" root="true">
    <activities xsi:type="org.pi4soa.cdl:While" name="NoAceptadaOferta">
    <activities xsi:type="org.pi4soa.cdl:Interaction" name="dameOfertas"
          operation="dameOfertas" relationship="//@typeDefinitions/@relationshipTypes.1"/>
    <activities xsi:type="org.pi4soa.cdl:Interaction" name="preguntasOfertas"
          operation="preguntasOfertas" relationship="//@typeDefinitions/@relationshipTypes.1"/>
    <activities xsi:type="org.pi4soa.cdl:Interaction" name="respuestasOfertas"
          operation="respuestasOfertas" relationship="//@typeDefinitions/@relationshipTypes.1"/>
    <activities xsi:type="org.pi4soa.cdl:Interaction" name="oferta" operation="oferta"
          relationship="//@typeDefinitions/@relationshipTypes.1"/>
    <activities xsi:type="org.pi4soa.cdl:Conditional" name="OfertaAceptada" expression="">
    <activities xsi:type="org.pi4soa.cdl:Interaction" name="generarDocumentacion"
         operation="generarDocumentacion" relationship="//@typeDefinitions/@relationshipTypes.0"/>
     </activities>
    </activities>
  </choreographies>
```

Por simplicidad, en la coreografía no hemos creado todos los ChannelTypes e InformationTypes ya que la idea fundamental que perseguirmos es la de ver como el flujo de la coreografía se define.

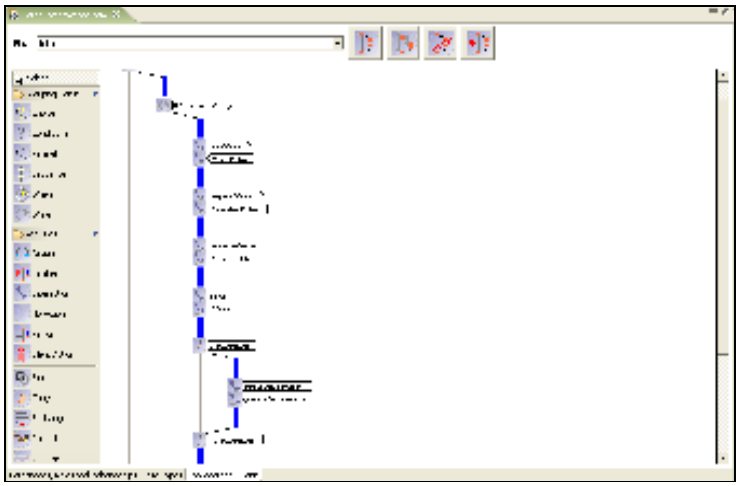


Figura 24 - Representación gráfica del flujo de la coreografía

Capítulo 5

Modelando Servicios Web Semánticos: WSML

Durante el desarrollo del capítulo 2, vimos que para describir los Servicios Web Semánticos, disponíamos del lenguaje WSMO o Web Service Modeling Ontology. Sin embargo, dicho lenguaje carece de una formalización (implementación) real de los conceptos que propone en un lenguaje que sea procesable por una máquina. Por esto motivo, si queremos realizar una aproximación práctica al mismo, debemos de utilizar algún otro lenguaje que implemente o formalice estos conceptos.

Teniendo en cuenta lo que acabamos de indicar el W3C recomienda utilizar el Web Service Modeling Language (WSML) o Lenguaje de Modelado de Servicios Web que provee una sintaxis formal y una semántica para el Modelado de Ontologías de Servicios Web (WSMO).

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 de la Web Semántica. 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 el 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 de esta sintaxis legible por seres humanos, WSML tiene una sintaxis XML y RDF para el intercambio sobre la Web y para la interoperabilidad con aplicaciones basadas en RDF. También es posible el mapeo entre ontologías WSML y ontologías OWL para interoperar con ontologías OWL por medio de un subconjunto semántico común de OWL y WSML.

Por otro lado, Web Service Modeling Ontology (WSMO) se basa en el Marco de Modelado de Servicios Web (WSMF), en donde el modelo de servicios describe el comportamiento del servicio en términos de procesos y de estructuras de control. La base conecta procesos, entradas y salidas en el modelo de proceso a un determinado protocolo de transporte descrito en un documento de WSDL.

WSMO supone uno de los mayores esfuerzos con el propósito de especificar la información semántica de los Servicios Web para hacer posible la automatización de servicios, su composición y ejecución. WSMO recomienda el uso de vocabularios ampliamente aceptados tales como Dubling Core y FOAF (de los que hablamos también durante el capítulo 3). En WSMO, un objetivo especifica qué quiere el usuario y la

descripción del Servicio Web definen la capacidad que proporciona el servicio. Con WSMO también pueden expresarse las propiedades no funcionales del servicio.

El objetivo de este capítulo es proporcionar un estudio del lenguaje WSML partiendo de una breve introducción de WSMO y aplicarlo a la modelización del problema descrito en el capítulo 3 de este documento. Para ello dividiremos el capítulo en las siguientes secciones:

5.1 Descripción general de WSMO
5.2 Entorno de trabajo que utilizaremos
5.3 El lenguaje WSML
5.4 Diseño de Servicios Web Semánticos

5.1 Descripción general de WSMO

WSMO proporciona una especificación basada en el uso de ontologías para los elementos principales de lo que se denomina como Servicios Web Semánticos y que describimos en el capítulo 2. Como vimos, 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 transforman la Internet que conocemos, de un repositorio de información para consumo humano a un sistema global de computación web distribuida. Es por este motivo, que un framework apropiado para Servicios Web Semánticos necesita integrar varias cosas:

- los principios básicos que hay detrás del diseño de aplicaciones web,
- los conceptos y tecnologías surgidos a raíz del uso de la Web Semántica,
- y los principios de aplicaciones distribuidas orientadas a servicios (SOA + Servicios Web).

WSMO cumple con estas premisas, de manera que se ha desarrollado siguiendo los siguientes principios básicos de diseño:

- **Compatible con la Web**: WSMO hereda el concepto de URI (del que hablamos en el capítulo 1) para identificar unívocamente los recursos como un principio de diseño esencial de la Web. Yendo más lejos, WSMO adopta el concepto de namespaces (de los que hablamos en el capítulo 2 al referirnos al XML) para denotar espacios de información consistentes, por tanto, se hace uso de XML y otras recomendaciones de la W3C así como de los principios de descentralización de recursos.
- **Basado en Ontologías**: Las ontologías es modelo de datos usado en WSMO, de manera que la descripción de los recursos así como los datos intercambiados durante el uso de los servicios están basados en ellas. Las ontologías pueden considerarse como el representación de conocimiento más ampliamente aceptada y su uso permiten realzar el significado semántico de la información permitiendo su proceso e interoperatividad, WSMO soporta los lenguajes de ontologías definidos en la Web Semántica.
- **Estrictamente desacoplado**: El desacoplamiento en WSMO denota que los recursos se definen de manera aislada, lo que significa que cada recurso está especificado de manera independiente sin necesidad de tener que interactuar para ello con otros recursos.
- **Mediación**: Es un principio de diseño complementario al de estrictamente desacoplado, consiste en la heterogeneidad natural que existe en los entornos abiertos y en la necesidad de tratarla. Esta heterogeneidad se puede dar a nivel de los datos, en los protocolos o en los procesos.

- **Separación en Roles Ontológicos**: Los clientes existen en un contexto específico los cuales no tienen porque coincidir con el de un servicio web disponible. Por ejemplo, un usuario (cliente) puede querer reservar unas vacaciones en función de sus preferencias de tiempo, cultura o guarderías disponibles en destino, mientras que el servicio web proporciona información sobre viajes de avión y disponibilidad hotelera. La epistemología que subyace en WSMO diferencia entre los deseos de usuarios y clientes y la disponibilidad de lo que ofrecen los servicios.
- **Descripción vs Implementación**: WSMO diferencia entre la descripción de los elementos de un Servicio Web Semántico (descripción) y las tecnologías de ejecución (implementación). De este modo se garantiza que WSMO proporciona un modelo apropiado de descripción sin importar la tecnología que lo utilice (actual o futura)
- **Servicio vs Servicio Web**: Un Servicio Web es una entidad computacional que es capaz de realizar un objetivo del usuario al invocarlo. Un servicio, es el valor proporcionado por la invocación. WSMO proporciona maneras para describir un servicio web que proporciona acceso a servicios.

Una vez vistas cuales son las características básicas que hay detrás de WSMO, veamos ahora cuales son los elementos básicos en los que se dividen y que analizaremos en detalle en posteriores secciones. WSMO identifica cuatro elementos de alto nivel que agrupan los conceptos principales que tienen que ser descritos para poder definir un Servicio Web Semántico. Estos elementos son los siguientes:

- **Ontologías**: Proporcionan la terminología que utilizan otros elementos de WSSMO para describir los aspectos relevantes, las ontologías proporcionan definiciones formales que son procesables por la máquina.
- **Servicios Web**: Representan entidades computacional capaces de proporcionar acceso a los servicios. La descripción del servicio web abarca los interfaces, capacidades y el trabajo interno del servicio. Todos estos aspectos del servicio web son descritos utilizando la terminología definida por las ontologías.
- **Goals**[28]: Describen aspectos relacionados con los deseos de los usuarios (clients) respecto a la funcionalidad solicitada, igual que antes, se pueden usar las ontologías para describir la terminología usada en el dominio así como los aspectos más relevantes de los propios Goals.
- **Mediators**: Se utilizan para describir aquellos elementos que gestionan (manipulan) los problemas de interoperatividad entre diferentes elementos de WSMO.

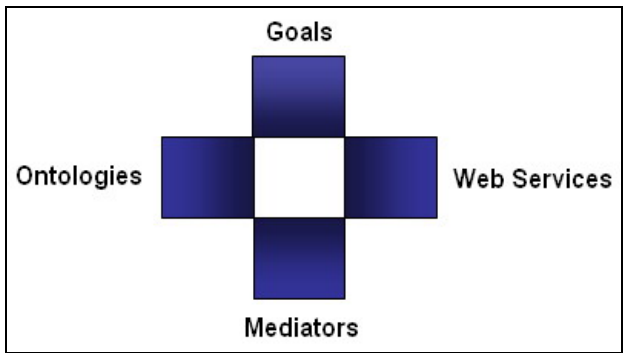


Figura 25 - Elementos de WSMO

[28] Goals – En castellano se traducen como Metas, nosotros mantendremos el término en inglés por claridad

5.2 Entorno de trabajo que utilizaremos

Como ya hemos indicado al principio en los objetivos del documento, para la elaboración de las distintas partes que componen este capítulo, nos hemos basado en el uso de herramientas de tipo Open Source. Concretamente en el caso de la modelización de Servicios Web Semánticos vamos a utilizar como herramienta fundamental Web Service Modeling Toolkit en su versión 1.3, puede descargarse desde la siguiente URL: http://wsmt.sourceforge.net

En la figura siguiente vemos la apariencia que tiene esta herramienta:

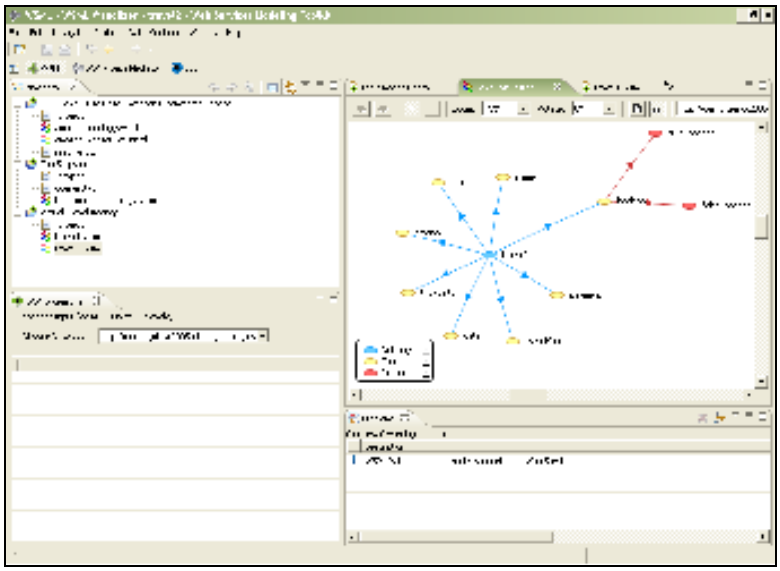


Figura 26 - Apariencia de Web Service Modeling Toolkit

5.3 El lenguaje WSML

WSMO propone un modelo conceptual para la descripción de Ontologías (Ontologies), Servicios Web Semántico (Semantic Web Services), Metas (Goals) y Mediadores (Mediators), siendo WSMO el punto de partida que siempre debemos de seguir cuando se estudia la familia de lenguajes destinados a la descripción de Servicios Web Semánticos. En nuestro caso, de la familia de lenguajes que permite describir un Servicio Web Semántico hemos seleccionado el lenguaje WSML ya que proporciona una mecanismo formal para describir todos los elementos definidos en WSMO, además de ser dentro de la industria el que más avanzado se encuentra, disponiendo de herramientas que facilitan el trabajo con él.

Las diferentes variaciones de WSML se corresponden con los diferentes niveles de expresividad lógica y al uso de diferentes paradigmas de lenguajes. Más específicamente, la Lógica Descriptiva, la Lógica de Primer Orden y la Lógica de Programación fueron los puntos de inicio para el desarrollo de las diferentes variaciones del lenguaje WSML, y estas variaciones afectan tanto a los aspectos sintácticos como a los aspectos semánticas del lenguaje. Sin embargo, hay una cualidad que se mantiene constante en las diversas variantes y es que WSML fue creado en términos de tener una sintaxis entendible de manera directa por los humanos con elementos similares a los elementos que se definen en el modelo conceptual de WSMO. Yendo más lejos, WSML proporciona una sintaxis de intercambio para XML y RDF, así como utilidades de mapeo entre las ontologías que se

definen en WSML y las que podemos definir con OWL para mejorar la interoperabilidad con las aplicaciones basadas en OWL.

Las Ontologías como los Servicios Web Semánticos, necesitan de lenguajes formales para su especificación a fin de permitir su procesamiento automatizado. Como lenguaje para descripción de Ontologías, la recomendación de la W3C es OWL pero este lenguaje presenta limitaciones tanto a nivel conceptual como en la definición de alguna de sus propiedades formales, para superar estas limitaciones se creo el lenguaje OWL-S para la definición de Servicios Web Semánticos, pero el diseño de este lenguaje también presenta limitaciones a nivel conceptual, y las propiedades formales del lenguaje no están enteramente claras. Por ejemplo, OWL-S ofrece la elección entre diferentes lenguajes para especificar las precondiciones y los efectos, sin embargo, no explica como estos lenguajes deben de interactuar con OWL, y como debe de ser usados para especificar las entradas y las salidas. Todo esto hizo que fuese necesaria la aparición de WSMO como lenguaje conceptual y WSML como lenguaje que implementa estos conceptos.

5.3.1 Variantes en WSML

La siguiente figura muestra las diferentes variaciones de WSML y las relaciones que se establecen entre cada una de ellas. Cada variación difiere fundamentalmente en la expresividad lógica que ofrece y en el paradigma del lenguaje subyacente. El motivo de que existan estas variaciones, es para permitir a los usuarios elegir entre la expresividad que necesitan en sus aplicaciones y la complejidad, de manera que es el propio usuario el que determina qué variante de WSML elegir.

Como se puede ver en la figura, lo que se considera como lenguaje básico WSML esta definido como WSML-Core que se ramifica en dos direcciones diferentes denominadas, Lógica de Descripción o WSML-DL y Lógica de Programación o WSML-Flight o WSML-Rule. A continuación veremos un poco más en detalle las características de cada una de las variantes presentes en WSML.

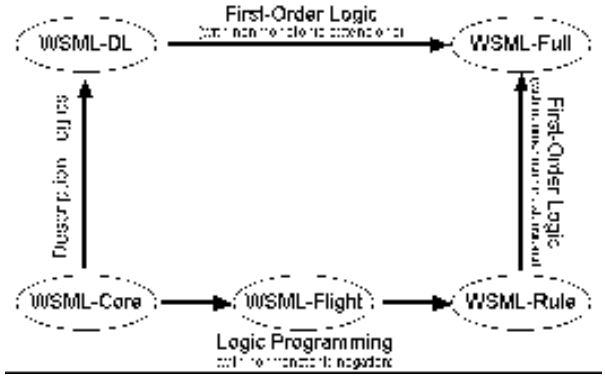


Figura 27 - Sublenguajes WSML

- **WSML-Core**: Este lenguaje se define por la intersección entre la Lógica Descriptiva y la Lógica en la Descripción de Programas. Es el lenguaje con menos poder expresivo de todos los lenguajes de la familia WSML. Las características principales del lenguaje son su soporte para el modelado de clases, atributos, relaciones binarias e instancias. También soporta la herencia de clases y las relaciones jerárquicas.
- **WSML-DL:** Este lenguaje es una extensión de WSML-Core, contiene gran parte de la expresividad en la definición de ontologías que proporciona el lenguaje OWL.

- **WSML-Flight:** Otra extensión del WSML-Core que incluye la posibilidad de incluir restricciones. Se basa en una variante de programación lógica denominada F-Logic y proporciona un lenguaje muy potente para la definición de reglas.
- **WSML-Rule:** Este lenguaje es una extensión de WSML-Flight añadiendo más posibilidades dentro de la lógica de programación. El lenguaje añade extensiones tales como el uso de símbolos de función y la posibilidad de definir reglas no seguras.
- **WSML-Full:** Unifica los lenguajes WSML-DL y WSML-Rule de forma que contiene todas las posibilidades de ambos.

Gráficamente la representación de estos lenguajes podemos verla en la siguiente figura:

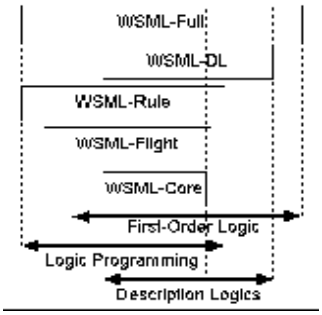


Figura 28 - Solapamientro entre sublenguajes WSML

Tal y como se puede ver en la figura, WSML tiene dos capas alternativas, una formada por "WSML-Core → WSML-DL → WSML-Full" y otra formada por "WSML-Core → WSML-Flight → WSML-Rule → WSML-Full". En ambas capas, WSML-Core es el lenguaje menos expresivo y WSML-Full el que añade mayor expresividad. Ambas capas son disjuntas ya que la interoperabilidad que puede darse entre WSML-Flight y WSML-Rule junto con WSML-DL solo es posible a través del elemento común que proporciona WSML-Core.

A lo largo de las siguientes secciones la descripción de los elementos que presentaremos se corresponderá al lenguaje WSML-Core, que será el que posteriormente en el punto 4.3 utilicemos para especificar la semántica de nuestros Servicios Web.

5.3.2 Especificación de Ontologías en WSML

Para especificar una ontología en WSML se hace uso de la palabra "ontology" que es seguida opcionalmente de una IRI la cual sirve para identificar de manera univoca a la ontología.

Ejemplo:

ontology *family*

La especificación de ontologías en WSML sigue la siguiente sintaxis:

ontology = *'ontology' id? header* ontology_element**

ontology_element = *concept* | *relation* | *instance* | *relationinstance* | *axiom*

En donde, podemos utilizar en nuestra definición lo cualquiera de los siguientes elementos:

- **Conceptos (concepts)**: La definición de un concepto se realiza con "concept" seguido de un identificador opcional del concepto. Opcionalmente se puede seguir con la definición de un súper concepto si utilizamos la palabra "subConceptOf" seguido de uno o mas identificadores de conceptos. Dado que WSML permite la herencia, la definición de los atributos se hereda, lo que significa que un concepto hereda todos los atributos de su súper concepto. Si dos súper conceptos tiene en común la misma definición de un atributo pero los rangos de ambos atributos son distintos, los atributos se interpretan de manera conjunta, de forma que el rango del atributo en el concepto es el rango que tiene el atributo en cada uno de los súper conceptos. La sintaxis completa para definir conceptos es la siguiente:

```
concept      =  'concept' id superconcept? nfp? attribute*
superconcept =  'subConceptOf' idlist
```

A continuación tenemos un ejemplo:

```
concept Human subConceptOf {Primate, LegalAgent}
  nonFunctionalProperties
    dc#description hasValue "concept of a human being"
    dc#relation hasValue humanDefinition
  endNonFunctionalProperties
  hasName ofType foaf#name
  hasParent impliesType Human
  hasChild impliesType Human
  hasAncestor impliesType Human
  hasWeight ofType _float
  hasWeightInKG ofType _float
  hasBirthdate ofType _date
  hasObit ofType _date
  hasBirthplace ofType loc#location
  isMarriedTo impliesType Human
  hasCitizenship ofType oo#country
```

En este ejemplo definimos el concepto Humano el cual deriva de los concepto Primate y AgenteLegal. Además definimos una serie de propiedades como son el nombre, peso o fecha de nacimiento, entre otras.

- **Axiomas (axioms)**: WSML permite la creación de axiomas a fin de refinar la definición dada por la sintaxis conceptual, por ejemplo al definir un subconjunto y atributos. Por ejemplo, si queremos definir el concepto Humano como la intersección de los conceptos Primate y AgenteLegal, podemos usar la siguiente definición axiomática:

```
axiom humanDefinition
  definedBy
          ?x memberOf Human equivalent
          ?x memberOf Primate and
          ?x memberOf LegalAgent.
```

La sintaxis completa para la definición de axiomas es la siguiente:

```
axiom        =  'axiom' axiomdefinition
axiomdefinition =  id
                |  id? nfp? log_definition
```

```
log_definition   =   'definedBy' log_expr+
```

Y otro ejemplo para definir a Humano puede ser la quedamos a continuación:

```
axiom humanDefinition
  definedBy
    ?x memberOf Human equivalent
    ?x memberOf Animal and
    ?x memberOf LegalAgent.
```

Como dato importante indicar que WSML permite la especificación de restricciones al estilo de las que se incluyen al definir las entidades en una base de datos. El siguiente ejemplo ilustra esta idea:

```
axiom humanBMIConstraint
  definedBy
          !- naf bodyMassIndex(bmi hasValue ?b,
                               length hasValue ?l,
                               weight hasValue ?w) and
             ?x memberOf Human and
   ?x[length hasValue ?l, weight hasValue ?w,
     bmi hasValue ?b].
```

- **Relaciones (relations)**: Las relaciones comienzan con la palabra "relation" seguidas de un identificador de la relacion. WSML permite especificar relaciones con una cardinalidad arbitraria, aunque se puede indicar al definir la relación. El dominio de los parámetros de la relacion pueden ser especificados opcionalmente usando la palabra "impliesType" o "ofType", y es importante no olvidar que el orden de los parametros de una relación es algo estricto y no pueden cambiarse de manera arbitraria. La definición de una relacion puede completarse para indicar si es una subrelación mediante la palabra "subRelationOf" seguida de los identificadores de las relaciones que actuan como padre. Se puede incluir también y al igual que sucede con los conceptos un bloque de propiedades no funcionales mediante "nonFunctionalProperties". La sintaxis completa para definir relaciones en WSML es la que se indica a continuación:

```
relation        =   'relation' id arity? paramtyping? superrelation? nfp?
arity           =   '/' pos_integer
paramtyping     =   '(' paramtype moreparamtype* ')'
paramtype       =   att_type idlist
moreparamtype   =   ',' paramtype
superrelation   =   'subRelationOf' idlist
```

Y aquí vemos dos ejemplos de relaciones que pueden definirse siguiendo este esquema:

```
relation distance
          (ofType City, ofType City, impliesType _decimal)
           subRelationOf measurement
relation distance/3
```

- **Atributos**: WSML permite la definición de dos clases de atributos mediante el uso de las palabras "ofType" y "impliesType". Un atributo definido de la forma "A ofType D", indica que "A" es un identificador de atributo y "D" es un identificador de conceptos que actúa como resctriccion de los valores que puede tomar el atributo

"A", ya que si el valor del atributo no está dentro del tipo "D" se producirá una violación en la restricción de la definición del mismo. Por otro lado, definir un atributo como "impliesType" es útil si queremos inferir el tipo de un valor de atributo particular.

A continuación se muestra la sintaxis completa para definir un atributo:

```
attribute          = [attr]: id attributefeature* att_type cardinality? [type]: idlist nfp?

att_type           = 'ofType'
                   | 'impliesType'

cardinality        = '(' [min_cardinality]: digit+ [max_cardinality]: cardinality_number? ')'

cardinality_number = {finite_cardinality}   digit+
                   | {infinite_cardinality} '*'

attributefeature   = 'transitive'
                   | 'symmetric'
                   | 'inverseOf' '(' id ')'
                   | 'reflexive'
```

En el ejemplo de la definición del concepto Humano vemos como se definen diferentes tipos de atributos:

```
concept Human
  nonFunctionalProperties
    dc#description hasValue "concept of a human being"
  endNonFunctionalProperties
  hasName ofType foaf#name
  hasParent inverseOf(hasChild) impliesType Human
  hasChild impliesType Human
  hasAncestor transitive impliesType Human
  hasWeight ofType (1) _float
  hasWeightInKG ofType (1) _float
  hasBirthdate ofType (1) _date
  hasObit ofType (0 1) _date
  hasBirthplace ofType (1) loc#location
  isMarriedTo symmetric impliesType (0 1) Human
  hasCitizenship ofType oo#country
```

- **Instancias**: La definición de una instancia se realiza mediante la palabra "instance" seguida de manera opcional por un identificador de instancia, a continuación se pone la palabra "memberOf" para indicar el nombre del concepto al cual la instancia pertenece. La sintaxis para definir una instancia es la siguiente:

```
instance       = 'instance' id? memberof? nfp? attributevalue*

memberof       = 'memberOf' idlist

attributevalue = id 'hasValue' valuelist
```

Por ejemplo, si queremos indicar que Mary es un instancia de Mujer y a la vez Madre podemos hacerlo del modo siguiente:

```
instance Mary memberOf {Parent, Woman}
  nfp
        dc#description hasValue "Mary is parent of the twins Paul and Susan"
  endnfp
  hasName hasValue "Maria Smith"
  hasBirthdate hasValue _date(1949,9,12)
```

hasChild ***hasValue*** *{Paul, Susan}*

Otra posibilidad de las instancias es la capacidad que tiene WSML para relacionarlas entre si, para ello se hace uso de la palabra “relationInstance” siguiendo la siguiente sintaxis:

relationinstance = *'relationInstance' [name]: id? [relation]: id '(' value (',' value)* ')' nfp?*

Un ejemplo de uso puede ser el siguiente:

relationInstance *distance(Innsbruck, Munich, 234)*

5.3.3 Especificación de Capacidades e Interfaces en WSML

La funcionalidad deseada y proporcionada se describe en WSML en forma de capacidades (capabilities). 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 peticionario y el proveedor de la capacidad interactúa es descrita a través de una interface, que a su vez puede ser también parte de una meta o de un 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 poscondiciones describen la relación entre la entra y la salida del servicio.

Una meta o Servicio Web en WSML solo puede tener una única capacidad, siendo opcional su especificación.

La forma en la que se define una capacidad comienza con la palabra “capability” seguida del nombre de la capacidad, a la que se le puede sumar un bloque opcional de propiedades no funcionales (nonFunctionalProperties), un bloque de importación de ontologías (importsOntology) y un bloque de mediadores a usar (usesMediator).

El bloque denominado “sharedVariables” se usa para indicar las variables que son compartidas entre la precondición y la poscondición, que se definen mediante “precondition” y “postcondition” respectivamente. No hay limitación en el número de precondiciones y poscondiciones que se pueden definir, y al igual que las capacidades pueden incluir su propio bloque de propiedades no funcionales.

La sintaxis completa para definir una capacidad es la siguiente:

capability = *'capability' id? header* sharedvardef? pre_post_ass_or_eff**

sharedvardef = *'sharedVariables' variablelist*

pre_post_ass_or_eff =
'precondition' axiomdefinition
| *'postcondition' axiomdefinition*
| *'assumption' axiomdefinition*
| *'effect' axiomdefinition*

Por ejemplo:

```
capability
   sharedVariables ?child
   precondition
      nonFunctionalProperties
```

```
        dc#description hasValue "The input has to be boy or a girl
            with birthdate in the past and be born in Germany."
    endNonFunctionalProperties
    definedBy
      ?child memberOf Child
          and ?child[hasBirthdate hasValue ?birthdate]
          and ?child[hasBirthplace hasValue ?location]
          and ?location[locatedIn hasValue oo#de]
          or (?child[hasParent hasValue ?parent] and
                ?parent[hasCitizenship hasValue oo#de] ) .

effect
    nonFunctionalProperties
      dc#description hasValue "After the registration the child
          is a German citizen"
    endNonFunctionalProperties
    definedBy
      ?child memberOf Child
          and ?child[hasCitizenship hasValue oo#de].
```

Como ya hemos indicado, una meta puede solicitar múltiples interfaces y un Servicio Web puede ofrecerlos. Para definir una interface se hace uso de la palabra “interface” seguida de manera opcional de un identificador y de un bloque de propiedades no funcionales. Conviene indicar que mediante los interfaces se puede indicar la coreografía u orquestación que queremos utilizar. La sintaxis completa es:

```
interfaces     =  interface
                | minterfaces
minterfaces    =  'interface' '{' id moreids* '}'
interface      =  'interface' id? header* choreography? orchestration?
choreography   =  'choreography' id
orchestration  =  'orchestration' id
```

A continuación se muestra un ejemplo de interface:

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

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

5.3.4 Especificación de Metas en WSML

La especificación de una meta (goal) en WSML se realiza con la palabra “goal” seguida de una IRI que sirve para identificar de manera univoca a la meta.

La sintaxis completa para definir una meta es la siguiente:

```
goal =  'goal' id? header* capability? interfaces*
```

Y un ejemplo de utilización podemos verlo en la siguiente línea:

```
goal _http://example.org/Germany/GetCitizenShip
```

5.3.5 Especificación de Mediadores en WSML

WSML permite especificar los cuatro tipos de mediadores que se definen en WSMO genéricamente como: Mediadores Ontológicos, Mediadores entre Servicios Web, Mediadores entre Metas y Mediadores entre Servicios Web y Metas, y que se referencia

con los nombres de ooMediator, wwMediator, ggMediator y wgMediator respectivamente. La sintaxis para indicar el tipo de mediator que definimos es la siguiente:

mediator = *oomediator* | *ggmediator* | *wgmediator* | *wwmediator*

Los mediadores del tipo ooMediators, se utilizan para conectar ontologías con otras ontologías, servicios web, metas u otros mediadores, aunque también pueden utilizarse para importar otras ontologías, para ello se basan en la siguiente sintaxis:

oomediator = *'ooMediator' id? nfp? importedontology? sources target? use_service?*

Los mediadores del tipo wwMediators, se utilizan para conectar Servicios Web, resolver cualquier dato, proceso o protocolo existentes entre los servicios que conectan, por este motivo los wwMediators solo pueden tener un origen y un destino unico. La sintaxis que utilizan para su definición es la siguiente:

wwmediator = *'wwMediator' id? header* source? target? use_service?*

Los mediadores del tipo ggMediators, se utilizan para conectar diferentes metas, permitiendo metas que refinan a otras metas de carácter más general. Al igual que sucede con los wwMediator solo pueden tener un origen y un único destino, los cuales pueden ser o bien una meta u otro ggMediator. La sintaxis de definición es la siguiente:

ggmediator = *'ggMediator' id? header* source? target? use_service?*

Finalmente los mediadores del tipo wgMediators, se utilizan para conectar metas y Servicios Web. La sintaxis de definición es la siguiente:

wgmediator = 'wgMediator' id? header* source? target? use_service?

5.3.6 Especificación de Servicios Web en WSML

La especificación de un Servicio Web se identifica en WSML con la palabra “webService” seguida de una IRI que lo identifica de manera única. La sintaxis para especificar completamente un Servicio Web es la siguiente:

webservice = 'webService' id? header* capability? interface*

Por ejemplo:

webService _http://example.org/Germany/BirthRegistration

5.4 Diseño de Servicios Web Semánticos

Para realizar el diseño y aplicar las ideas que hemos visto en el punto anterior tenemos que entrar dentro de la herramienta Web Service Modeling Toolkit, el cual como hemos comentado anteriormente permite la creación de documentos WSML de una manera intuitiva, pensemos que en la práctica no resulta eficiente realizar la codificación de este tipo de documentos de manera manual, ya que hay gran cantidad de cuestiones sintácticas que debemos de conocer y sería muy fácil cometer errores.

Una vez dentro del entorno, lo primero que debemos de hacer es crear un nuevo proyecto WSML, para ello tenemos seleccionamos “File > New Project” y en el cuadro de dialogo que parece indicamos “WSML Project”.

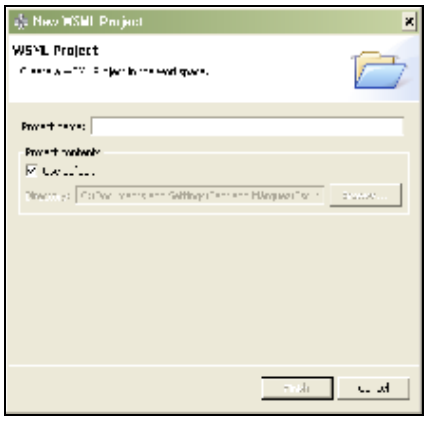


Figura 29 - Creación de un nuevo proyecto WSML

El nombre que vamos a dar a nuestro proyecto es “WebSemanticaTFCProject”, y en él vamos a crear varios elementos diferentes que sirvan para ejemplificar diferentes tipos de modelos que pueden plantearse con WSML. En concreto vamos a realizar los dos siguientes supuestos:

- Modelización de una ontología
- Modelización de un Servicio Web

5.4.1 Diseño de una ontología

Antes de indicar como realizaríamos la creación de la ontología vamos a añadir algunas restricciones adicionales al enunciado que propusimos a fin de que el ejemplo resulte más ilustrativo. En este caso vamos a realizar una ontología para el caso de la entidad “Vendedor”, para ello suponemos lo siguiente:

- Todo “Vendedor” tiene una serie de características que lo definen como tal y que son las siguientes: tipo, nombre comercial, dirección, código postal, localidad, provincia, país, email y dirección de su página web.
- Por otro lado, un “Vendedor” lleva un registro de sus clientes de los que guarda un código de cliente. No obstante, a efectos de realizar campañas de marketing los vendedores se han hecho con una lista de personas que no necesariamente tienen que ser clientes, y a las cuales pueden ofrecer publicidad de sus productos. Cara a este ejemplo y por simplicidad, hemos supuesto que se disponen de datos de tres personas de las cuales solamente una de ellas es cliente, además se establecen las relaciones que vinculan a cada una de estas personas, según esto tenemos:
    - Tres personas: “Santiago Márquez Solís”, “Encarnación Márquez Solís” y “Sofía Márquez Márquez”
    - De las tres personas solamente “Santiago Márquez Solís” es cliente.
    - “Santiago Márquez Solís” es el marido de “Encarnación Márquez Solís” y tiene por hija a “Sofía Márquez Márquez”
- Un “Vendedor” vende diferentes modelos de coche, de cada modelo almacena la información siguiente: marca, modelo, matricula, fecha de fabricación, color, número de puertas, número de plazas, precio de venta, descuento y si está o no de

oferta. A su vez los modelos solamente pueden ser de uno de los siguientes cuatro tipos: berlina, todoterrenos, deportivos y monovolumen.

- El "Vendedor" puede ofrecer el precio de los modelos de coches en diferentes monedas, concretamente los precios pueden venir dados en euros, en dólares o en pesetas.

Con todas estas ideas ya podemos pasar a definir nuestra ontología WSML para la entidad "Vendedor". Dado que tenemos creado el proyecto en donde vamos a albergar nuestros ficheros, lo que debemos de hacer es seleccionar la opción "File > New > Ontology", desde donde se nos abrirá un pequeño asistente para cumplimentar los datos básicos de nuestra ontología, que son los siguientes:

- Nombre de la ontología: le hemos dado el nombre "OntologíaVendedor", de manera que el fichero que se nos creará será "OntologiaVendedor.wsml".
- namespace: para este ejemplo hemos optado por utilizar el siguiente espacio de nombres:

  http://www.uoc.edu/tfc/webSemantica/ontologias/vendedor#

- ontology ID: IRI identificativa, en nuestro caso y dado que es un valor opcional lo dejamos en blanco
- Variante WSML: Sirve para indicar que tipo de lenguaje WSML queremos usar, en nuestro caso hemos elegido WSML-Core.

Indicadas las propiedades anteriores, pulsamos sobre el botón de "Finish" y podemos pasar a crear la ontología propiamente dicha.

La herramienta permite que definamos la ontología de dos formas distintas, bien escribiendo el documento directamente, siguiendo las reglas de WSML, lo que no deja de estar sujeto a errores, o bien utilizar el editor gráfico, en donde se puede definir cada uno de los elementos que necesitemos. Podemos cambiar de forma de visualización seleccionado el fichero WSML y haciendo clic en el botón derecho del ratón seleccionar la opción "Open with" e indicar "WSML Text Editor" o "WSML Visualizer" para usar el editor de texto o el editor gráfico respectivamente.

La creación de los elementos desde el editor gráfico es muy simple y se basa en pulsar el botón derecho y en la opción "Add" indicar el elemento que queremos crear, los elementos que nos aparecen en esta opción depende del elemento que tengamos seleccionado en pantalla. Además, siempre tenemos visible en la pantalla del editor gráfico una leyenda en donde nos indica que significa los elementos que aparecen en la pantalla, pudiendo mostrar u ocultar aquellos que nos interese en un determinado momento.

Por tanto, teniendo en cuenta las restricciones que hemos indicado al comienzo de esta sección vemos que tendremos que crear varios conceptos, representados por los modelos y sus tipos, las monedas, las caracteristicas, las personas y los clientes. Además vemos, que los tipos de modelos y los modelos presentan una relación de herencia, lo que sucede con las personas y los clientes. Finalmente tenemos instancias concretas para los modelos, asi como para las personas y para los clientes.

Por tanto, los conceptos los modelizaremos como "concept", las instancias como "instance" y las relaciones de herencia como "subconceptOf".

Si hemos seguido las indicaciones anteriores habremos generado un fichero WSML que debe de tener el siguiente contenido:

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-core"
namespace { _"http://www.uoc.edu/tfc/webSemantica/ontologias/vendedor#"
,
    wsml _"http://www.wsmo.org/wsml/wsml-syntax#" }

ontology vendedor

concept caracteristicas
    tipo impliesType _string
    nombre impliesType _string
    direccion impliesType _string
    cp impliesType _string
    localidad impliesType _string
    provincia impliesType _string
    pais impliesType _string
    email impliesType _string
    web impliesType _string

concept clientes
    codigo impliesType _string

concept modelos
    marca impliesType _string
    modelo impliesType _string
    matricula impliesType _string
    fechaFabricacion impliesType _date
    matricula impliesType _string
    color impliesType _string
    numPuertas impliesType _int
    numPlazas impliesType _int
    precioVenta impliesType _float
    descuento impliesType _int
    deOferta impliesType _boolean

concept tipoModelos subConceptOf modelos

instance berlina memberOf tipoModelos
instance todoterreno memberOf tipoModelos
instance deportivo memberOf tipoModelos
instance monovolumen memberOf tipoModelos

concept moneda

instance euro memberOf moneda
instance dolar memberOf moneda
instance pesetas memberOf moneda

concept personas
concept clientes subConceptOf personas

instance santiago_marquez memberOf clientes
    nonFunctionalProperties
        titulo hasValue "D. Santiago Márquez Solís"
    endNonFunctionalProperties
    nombre hasValue "Santiago Márquez"
    sexo hasValue varon
    esposa hasValue marge_simpson
    hijos hasValue {sofia_marquez}

instance encarna_marquez memberOf personas
    nonFunctionalProperties
        titulo hasValue "Sr. Encarnacion Márquez Solís"
    endNonFunctionalProperties
    nombre hasValue "Encarnación Márquez"
    sexo hasValue mujer
    marido hasValue santiago_marquez
    hasChild hasValue {sofia_marquez }

instance sofia_marquez memberOf personas
```

```
nonFunctionalProperties
    titulo hasValue "Sr. Encarnacion Márquez Solís"
endNonFunctionalProperties
nombre hasValue "Encarnación Márquez"
sexo hasValue mujer
marido hasValue santiago_marquez
hasChild hasValue {sofia_marquez }
```

Y que gráficamente se expresa del siguiente modo:

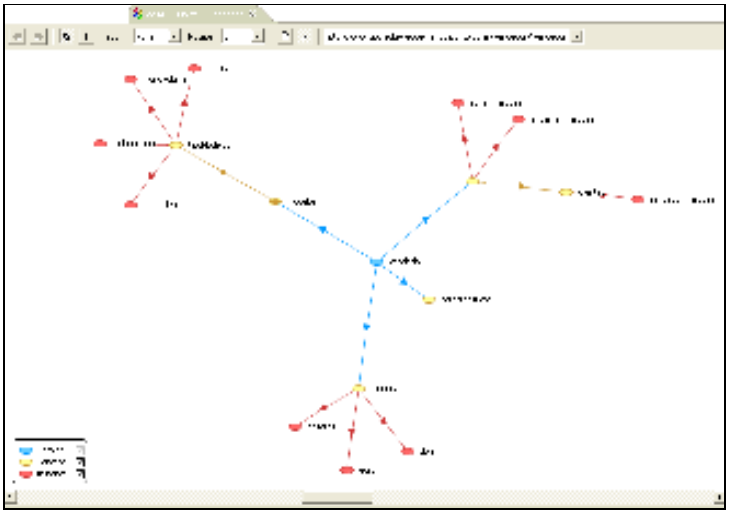


Figura 30 - Vista gráfica de la Ontología Vendedor

Como vemos resulta relativamente sencillo, extender estas ideas a la entidad "Comprador" e incluso a la entidad "Organismo Oficial" añadiendo la semántica que consideremos necesaria a los mismos.

5.4.2 Diseño de un Servicio Web

Tres son los Servicios Web asociados a los servicios que hemos definido en el enunciado de nuestro problema, por tanto, necesitamos un Servicio Web denominado "Vendedor", otro "Comprador" y otro "Organismo Oficial". Igual que hicimos en el caso de la modelización de ontologías solamente vamos a ver como modelizar uno de ellos y seguiremos con el mismo ejemplo que antes, es decir con la entidad "Vendedor".

Para crear el servicio asociado con el "Vendedor" debemos seleccionar "File > New > Web Service", y en el asistente que nos aparece en pantalla debemos de indicar los siguientes valores de información:

- Nombre del Servicio Web: Indicaremos "VendedorWS".
- Namespace: Se corresponde con el espacio de nombre que queremos dar a cada uno de nuestros servicios, en nuestro caso daremos el siguiente valor:

  http://www.uoc.edu/tfc/webSemantica/vendedorWS#
- Web Service ID: Valor identificativo único asociado a nuestro servicios. Daremos en este caso el siguiente valor "VendedorWSID"
- Variante WSML que usaremos: Igual que antes, se refiere al sublenguaje WSML que usaremos para modelizar nuestro problema, y que volverá a ser WSML-Core

La herramienta permite que definamos los elementos de los servicios de dos formas distintas (igual que cuando definíamos las ontologías), bien escribiendo el documento directamente, siguiendo las reglas de WSML, lo que no deja de estar sujeto a errores, o bien utilizar el editor gráfico, en donde se puede definir cada uno de los elementos que necesitemos. Podemos cambiar de forma de visualización seleccionado el fichero WSML y haciendo clic en el botón derecho del ratón seleccionar la opción “Open with” e indicar “WSML Text Editor” o “WSML Visualizer” para usar el editor de texto o el editor gráfico respectivamente.

La creación de los elementos desde el editor gráfico es muy simple y se basa en pulsar el botón derecho y en la opción “Add” indicar el elemento que queremos crear, los elementos que nos aparecen en esta opción depende del elemento que tengamos seleccionado en pantalla. Además, siempre tenemos visible en la pantalla del editor gráfico una leyenda en donde nos indica que significa los elementos que aparecen en la pantalla, pudiendo mostrar u ocultar aquellos que nos interese en un determinado momento.

En este ejemplo, vamos a modelizar como sería quedaría la operación de consulta de las ofertas por parte de la entidad “Vendedor”. Lo primero que debemos de hacer es indicar en las propiedades no funcionales del Servicio Web, los valores asociados al autor, versión y cualquier dato adicional que consideremos importante indicar.

Por otro lado, definimos una capacidad (capability) para proporcionar todas las características que nuestro servicio va a poner a disposición de los clientes y que denominaremos “vendedorWSCapability”, en este caso nos centramos solamente en una operación de “consulta de vehículos de oferta”. Al definir la capability debemos indicar tanto las poscondiciones como las precondiciones, de cada una de las acciones que la capability proporcione, en nuestro ejemplo la entrada o precondición viene determinada por una variable denominada “request” que es miembro de “itemSearchRequest” que equivale a las características (o preferencias) que un “Comprador” tiene por un determinado vehículo.

Si hemos seguido las indicaciones anteriores habremos generado un fichero WSML que debe de tener el siguiente contenido:

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml-syntax/wsml-core"
namespace { _"http://www.uoc.edu/tfc/webSemantica/vendedorWS#",
  dc _"http://purl.org/dc/elements/1.1/",
  dt _"http://www.wsmo.org/ontologies/dateTime#",
  loc _"http://www.wsmo.org/ontologies/location#",
  xsd _"http://www.w3.org/2001/XMLSchema#",
  wsml _"http://www.wsmo.org/wsml/wsml-syntax#"
}

webService _"http://www.uoc.edu/tfc/webSemantica/vendedorWS"
  nonFunctionalProperties
    dc#title hasValue "Servicio Web Semantico para el Servicio Vendedor"
    dc#creator hasValue "Santiago Marquez Solis"
    dc#subject hasValue {"Vendedor", "Busqueda de Vehiculos", "Compra de Vehiculos"}
    dc#description hasValue "Servicio Web Semantico para la busqueda y compra de Vehiculos de Oferta "
    dc#publisher hasValue "UOC TFC Web Semantica"
dc#date hasValue "2006-12-10"
    dc#type hasValue _"http://www.wsmo.org/2004/d2#webservice"
    dc#format hasValue "text/html"
    dc#identifier hasValue _"http://localhost:8080/webServices/vendedorWS"
    dc#source hasValue _"http://localhost:8080/webServices/vendedorWS"
    dc#language hasValue "es-ES"
    wsml#version hasValue "$Revision: 1.00 $"
  endNonFunctionalProperties

capability vendedorWSCapability
  nonFunctionalProperties
    dc#description hasValue "Esta capacidad proporciona una descripcion comprensible
         de todo el Servicio Web del Vendedor. Todas las entradas y salidas de las operaciones,
       se muestran en esta descripcion"
  endNonFunctionalProperties
```

```
 sharedVariables {?request}

 precondition
     nonFunctionalProperties
       dc#description hasValue "El servicio Vendedor manipula peticiones para:
        - Consulta de vehiculos en oferta"
     endNonFunctionalProperties

     definedBy
      //Busqueda de vehiculos por caracteristicas
      ?request memberOf itemSearchRequest.

postcondition
     nonFunctionalProperties
      dc#description hasValue "El Servicio puede devolver:
        - una lista con los vehiculos de oferta
        - nulo"
     endNonFunctionalProperties

     definedBy
      // El resultado de la petición de busqueda de vehiculos de oferta es un contenedor (items)
      // de todos los vehiculos en oferta que contiene el vendedor:
      (?request[vehiculos hasValue ?vehiculos] memberOf busquedaVehiculos implies
        exists ?container (?container memberOf itemContainer and
                                                    forall ?item (?container[items hasValue ?item]
                                                    implies ?item[vehiculos hasValue ?vehiculos]    memberOf
vehiculosVendedor))).

 interface vendedorWSInterface
  choreography
    stateSignature
     importsOntology _http://localhost:8080/ontologias/vendedor/OntologiaVendedor
    in itemSearchRequest withGrounding _http://localhost:8080/VendedorWS/2006-12-
10#wsdl.interfaceMessageReference(VendedorWSPortType/ItemSearch/In)

     out listContainer withGrounding {
        _"http://localhost:8080/VendedorWS/2006-12-
10#wsdl.interfaceMessageReference(VendedorWSPortType/ListSearch/Out)",
        _"http://localhost:8080/VendedorWS/2006-12-
10#wsdl.interfaceMessageReference(VendedorWSPortType/ListLookup/Out)"
       }
    transitionRules
     if (?ItemSearchRequest memberOf itemSearchRequest) then
        add(_# memberOf itemSearchResponse)
     endIf
```

Y que gráficamente se expresa del siguiente modo:

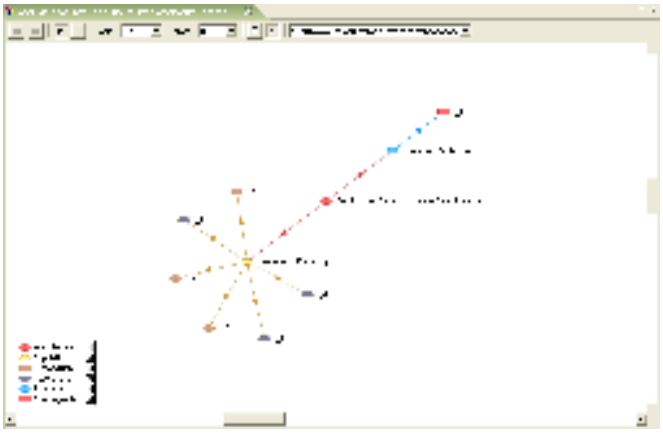


Figura 31 - Representación gráfica de un servicio

Conclusiones

A lo largo de este documento hemos visto multitud de aspectos que han intentado poner un poco de luz en todos los aspectos relacionados con la Web Semántica y tecnologías relacionadas con la misma.

De todos los conceptos vistos parece razonable establecer las siguientes conclusiones finales sobre la Web Semántica:

- La Web Semántica es un framework construido sobre la web actual para conseguir que las máquinas sean capaces de procesar la información que hay en Internet.
- Las características más interesantes de la Web Semántica son su flexibilidad y la facilidad que ofrece para que cualquiera pueda añadir información y relacionarla con otra ya existente, de forma similar a como se hace con la actual web. Para ello se hace necesario la utilización de ontologías que permitan añadir significado.
- Existen numerosas formas de modelizar ontologías, de todas ellas se puede decir que RDF es el más extendido a pesar de sus limitaciones. RDF es un modelo de datos que se puede poner por encima de XML para darle semántica. En RDF la información se describe en forma de triplas constituidas por un sujeto, una propiedad y un objeto. Todos los elementos de RDF son o recursos, identificados mediante una URI, o tipos de datos básicos.
- Dado que la definición mediante RDF no es suficiente, es necesario algún mecanismo que permita ontologías que sirvan para definir los conceptos y propiedades que se utilizan en RDF. La ontología los definirá mediante el establecimiento de relaciones entre ellos. El estándar para especificar ontologías del W3C es OWL y está fundamentado en las lógicas descriptivas.
- Sin embargo, OWL tiene algunas limitaciones expresivas. Unas son propias de la especificación del lenguaje y pueden ser subsanadas en futuras versiones.
- En un nivel superior, para el desarrollo de la Web Semántica se hace necesario la utilización de Servicios Web, los cuales posibilitan la creación de Arquitecturas Orientadas a Servicios (SOA) en donde lo que importa son las funcionalidades que a través de la interface de acceso del servicio web se expone a los clientes.
- La interacción entre varios Servicios Web se puede describir desde dos puntos de vista. Uno de ellos es como un observador externo a la interacción, en este caso, describiríamos la secuencia de operaciones y mensajes que

intercambian varios servicios web para llevar a cabo una interacción. A esto lo denominamos coreografía. El otro punto de vista es desde dentro de uno de los servicios web que interviene en la interacción y que va recibiendo y realizando invocaciones a los restantes elementos de la misma. A esto lo denominamos composición u orquestación

- Finalmente es posible añadir semántica a los Servicios Web del mismo modo que se hace con la web tradicional, para ello haremos uso del estándar WSMO (Web Service Modeling Ontology) que mediante la definición de cuatro elementos: Goals, Web Services, Ontologies y Mediator, permiten que un Servicio Web quede semánticamente descrito. Los conceptos expresados en WSMO se expresan en otros lenguajes como es WSML, en donde las reglas definidas en WSMO cobran vida.

Como hemos venido indicando, la Web Semántica es algo que promete. Hasta ahora en informática se ha planteado problemas sobre cómo almacenar mucha información o cómo ponerla a disposición de mucha gente. Estos problemas parece que ya han sido superados. En Internet disponemos de cantidades ingentes de información, incluso nuestros propios discos duros se han transformado en grandes almacenes de datos, por tanto el siguiente problema una vez resuelto como almacenar la información es como acceder a la misma, de modo que podamos organizar y procesar esa información para que al usuario le resulte fácil acceder a ella sin que tenga que perder el tiempo buscándola.

Y es en la resolución de esta cuestión hacia donde se encamina la Web Semántica. Sin embargo y aunque el futuro es ciertamente prometedor no debemos olvidar que también para su desarrollo se plantean algunos inconvenientes importantes, y es que hasta ahora los sistemas de razonamiento automático sólo han tenido éxito en áreas muy restringidas de la informática y el hecho de que la Web Semántica requiera del razonamiento automático para desarrollar algunas de sus características la hace poco atractiva para algunos. Por otro lado, los problemas de mantenimiento de las ontologías y la necesidad de que la información se publique en RDF (o similares) además de en HTML plantea más problemas aún para su desarrollo.

Trabajo a futuro

Debido a la cantidad tan variada de aspectos relacionados con los Servicios Web que toca el proyecto tratar líneas de trabajo futuro sobre todos estos aspectos podría resultar interminable. Por eso, vamos a dar unas pinceladas sobre líneas de trabajo en los distintos temas que toca el proyecto. Las líneas de trabajo generales sobre Web Semántica y Servicios Web Semánticos son las siguientes:

- Es necesario estandarizar un lenguaje para definir coreografías de la misma forma que se ha estandarizado WSDL para las interfaces.
- BPEL4WS debe de poder integrarse mejor con WS-TRANSACTION y WSCOORDINATION, así como incrementar el soporte para los procesos de negocio abstractos. Quizás integrándolo con el lenguaje de coreografías.
- Es necesario trabajar para dotar de semántica los Servicios Web. Los primeros pasos deberían consistir en dar semántica a sus interfaces y lograr repositorios de servicios con semántica, esto es, dotar de semántica a UDDI. Además, la bicefalia que existe entre WSMO y lenguajes que implementan los conceptos que expone, desde nuestro punto de vista no resulta más que una traba que puede dar lugar a diferentes lenguajes como está sucediendo con WSML.
- Hace falta establecer un framework que permita el descubrimiento de otros servicios y el negociado de contratos electrónicos basándose en criterios de calidad.
- Es necesario desarrollar sistemas que permitan la arbitración de conflictos entre dos partes debido a un incumplimiento del acuerdo.
- Se puede trabajar en mejorar los sistemas de aprovisionamiento automá-tico para los proveedores de servicio para que actúen en función de los contratos que el proveedor tenga firmados con los clientes.

Obviamente, estos son sólo algunos puntos muy generales, las líneas de investigación sobre todos estos temas son muchas y muy variadas.

Glosario

- **B2B (Bussiness to bussiness)**: Operación comercial que se produce entre dos empresas, en contraposición a aquella que se produce entre un cliente y una empresa.
- **BPEL4WS** (Business Process Execution Language forWeb Services): Lenguaje para describir composiciones de servicios web.
- **BPWS4J** (BPEL4WS for Java): Infraestructura para la utilización de BPEL4WS en JAVA.
- **CDL** (Choreography Description Language): Lenguaje para la descripción de coreografías.
- **DAML-S** : Ontología para modelar servicios web semánticos. Es la antecesora de OWL-S.
- **DTD** – Document Type Definition o Definición de Tipo de Documento
- **EDI** – Electronic Data Interchange (Intercambio de datos electrónico)
- **ERP** – Enterprise Resource Planning
- **GXA** (Global XML Architecture): Conjunto de protocolos para ampliar la funcionalidad de SOAP, WSDL y UDDI. Actualmente son más conocidos por WS-*.
- **METEOR-S**: Framework de servicios web semánticos desarrollado en la Universidad de Georgia.
- **NCSA** – National Center for Supercomputing Application (Centro Nacional de supercomputación de Aplicaciones)
- **OASIS** – Organization for the Advancement of Structure Information Standards
- **OWL** (Ontology Web Language): Lenguaje para representar ontologías fundamentadas en lógica descriptiva. Hay tres versiones: OWL Lite, OWL DL y OWL Full.
- **OWL-S** : Ontología para modelar servicios web semánticos.
- **Partner Process** – Haremos uso de esta expresión para referirnos a procesos que se montan en una empresa para ser utilizados por otra
- **QRL** (Quality Requirements Language): Lenguaje para especificar requisitos de calidad.
- **RDF** (Resource Description Framework): Lenguaje para la descripción de recursos en Internet.
- **RFC** : Request For Comment
- **SEKT** – Semantically Enabled Knowledge Technology, se engloba dentro del programa FP6 o Sixth Framwork Programme de la Unión Europea, que tiene por objeto promover la investigación y el desarrollo tecnológico.
- **SGML :** Standard Generalized Markup Language o Lenguaje de Marcado Generalizado
- **SOA** (Service Oriented Architecture): La arquitectura orientada a servicios se basa en descubrir automáticamente servicios software negociar para utilizarlos y componerlos, enlazarlos y ejecutarlos.
- **SOAP** (Simple Object Access Protocol): Protocolo para el envío de mensajes entre servicios web. Es independiente del protocolo de transporte y muy extensible.

- **SWRL** (Semantic Web Rule Language): Lenguaje de reglas para utilizar conjuntamente con OWL.
- **Unicode** – Asigna enteros no negativos a los caracteres escritos de cada idioma y establece la codificación binaria de esos números, por ejemplo UTF-8 es un ejemplo de codificación Unicode
- **URI** (Universal Resource Identi er): Sirve para identi car un recurso de forma única. Las URLs son un tipo de URI.
- **WSDL** (Web Service Description Language): Lenguaje de de nición de interfaces utilizado habitualmente para de nir interfaces de servicios web.
- **WSMO** (Web Service Modeling Ontology): Ontología para el modelado de servicios web semánticos.
- **WSLA** (Web Service Level Agreement): Lenguaje para especi car acuerdos de nivel de servicio.
- **XML Schema** : Lenguaje para especificar documentos XML.

Referencias

Sobre RSS

- http://www.universia.com.ar/portada/actualidad/noticia_actualidad.jsp?noticia=14748
- http://www.w3schools.com/rss/default.asp
- http://www.eevl.ac.uk/rss_primer/
- http://en.wikipedia.org/wiki/RSS_%28file_format%29
- http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html
- http://www.webreference.com/authoring/languages/xml/rss/intro/
- http://www.rssgeneral.com/

Sobre RDF

- http://www.xml.com/pub/a/98/06/rdf.html (BRAY, Tim. RDF and Metadata)
- http://www.openrdf.org/ (openRDF.org)
- http://planet.rdfhack.com/ (Planet RDF)
- http://www.w3.org/PICS/ (W3C. Platform for Internet Content Selection)
- http://www.w3.org/TR/rdf-pics (W3C. PICS Rating Vocabularies in XML/RDF)
- http://www.w3.org/TR/rdf-dawg-uc/ (W3C. RDF Data Access Use Cases and http://www.w3.org/RDF/ (W3C. Resource Description Framework (RDF))
- http://www.w3.org/2001/11/IsaViz/ (W3C. IsaViz: A Visual Authoring Tool for RDF)
- http://www.w3schools.com/rdf/ (W3School. RDF Tutorial)

Sobre OWL

- http://www.w3.org/2004/OWL/
- http://www.hipertexto.info/documentos/owl.htm
- http://www.w3.org/TR/owl-guide/ (W3C. OWL Web Ontology Language Guide)
- http://www.w3.org/TR/owl-ref/ (W3C. OWL Web Ontology Language. Reference)
- http://www.w3.org/TR/owl-test/ (W3C. OWL Web Ontology Language Test Cases)

Sobre Web Semántica

- http://www.w3.org/2000/10/swap/Primer (Conociendo RDF y la Web Semántica con N3)
- http://www.w3.org/DesignIssues/Semantic (Hoja de Ruta de la Web Semántica)
- http://purl.org/swag/whatIsSW (Que es la Web Semántica)

- http://uwimp.com/eo.htm (Semantic Web Primer)
- http://logicerror.com/semanticWeb-long (Introducción a la Web Semántica - Extenso)
- http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html (SciAm: The

Semantic Web)

- http://www.xml.com/pub/a/2001/03/07/buildingsw.html (Construyendo la Web Semántica)
- http://infomesh.net/2001/06/swform/ (The Semantic Web, Taking Form)
- http://www.w3.org/2001/sw/Activity (SW Activity Statement)
- http://www.w3.org/2000/01/sw/ (SWAD)

Sobre Servicios Web, WSML, SOAP y Tecnología Asociada

- http://webservices.xml.com/pub/a/ws/2001/04/04/soap.html (BOX, Don. A brief history of SOAP)
- http://www.semantic-conference.com/semanticwave.html (MILLS, Davis (dir). Semantic Wave 2006. SemTech 2006)
- http://www.brics.dk/~amoeller/WWW/ (MOLLER, Anders. SCHAWARTZBACH, Michael I. Interactive Web Services with Java, JSP, Servlets, JWIG, SOAP, WSDL, UDDI)
- http://www.brics.dk/ixwt/IXWT_C04c.pdf (MOLLER, Anders. SCHWARTZBACH, Michael I. "Schema Languages")
- http://www.brics.dk/ixwt/webservices.pdf (MOLLER, Anders. SCHWARTZBACH, Michael I. "Chapter 11. Web Services")
- http://www-128.ibm.com/developerworks/library/ws-soap/?dwzone=ws (OGBUJI, Uche. Using WDSL in SOAP applications: An introduction to WDSL for programmers)
- http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/index.html (VASUDEVAN, Venu. A Web Services Primer)
- http://www.verisign.com/rsc/wp/xml/xmloverview/xmloverview_wp.pdf (VeriSign. XML Trust Services Overview)
- http://www.w3c.es/Divulgacion/Guiasbreves/ServiciosWeb (W3C.es Guía breve de Servicios Web)
- http://www.w3.org/2001/sw/ (W3C. Semantic Web Activity)
- http://www.w3.org/TR/soap12-part1/ (W3C. Simple SOAP)
- http://www.w3.org/TR/ws-addr-core/ (W3C. Web Services Addredsing (WS-Addressing))
- http://www.w3.org/TR/wsdl20/ (W3C. Web Services Description Language (WSDL))
- http://www.w3.org/TR/ws-cdl-10/ (W3C. Web Services Choreography Description Language (WS-C DL))
- http://www.w3.org/2005/Talks/0712-hh-ec/ (W3C. Web Services, Semantic and Standarization: An Overview)

Sobre WSMO

- http://sourceforge.net/projects/wsmx/.
- http://www.wsmostudio.org/
- http://wsmo4j.sourceforge.net/
- http://www.w3.org/submission/2005/06
- http://www.wsmo.org/
- http://www.wsmo.org/wsml/
- http://www.wsmx.org/