DEPARTAMENTO DE INFORMÁTICA

ONTOLOGÍAS PARA SERVICIOS WEB SEMÁNTICOS DE INFORMACIÓN DE TRÁFICO: DESCRIPCIÓN Y HERRAMIENTAS DE EXPLOTACIÓN

JOSÉ JAVIER SAMPER ZAPATER

UNIVERSITAT DE VALENCIA
Servei de Publicacions
2005

Aquesta Tesi Doctoral va ser presentada a Valencia el día 5 de Juliol de 2005 davant un tribunal format per:

- D. Gregorio Martín Quetglás
- D. Jesús V. Albert Blanco
- D. José Llorens Sánchez
- D. Luis Amable García Fernández
- D. Martín Llamas Nistal

Va ser dirigida per:
D. Juan José Martínez Durá
D. Eduardo Carrillo Zambrabo

Depòsit legal:
I.S.B.N.:84-370-6270-5
Edita: Universitat de València
Servei de Publicacions
C/ Artes Gráficas, 13 bajo
46010 València
Spain
Telèfon: 963864115

UNIVERSITAT DE VALÈNCIA
DEPARTAMENT D´ INFORMÀTICA

ONTOLOGÍAS PARA SERVICIOS WEB SEMÁNTICOS DE INFORMACIÓN DE TRÁFICO: DESCRIPCIÓN Y HERRAMIENTAS DE EXPLOTACIÓN

TESIS DOCTORAL

Presentada por:
D. José Javier Samper Zapater

Dirigida por:
Dr. Juan José Martínez Durá
Dr. Eduardo Carrillo Zambrano
Valencia, 2005

A mis padres y resto de familia, en especial a Raquel y mi hijo Carlos,
por ese tiempo que no pude estar con ellos.
A mis amigos.

"El sueño detrás de la Web es un espacio común en el que nos comunicamos compartiendo información. Su universalidad es esencial: el hecho de que un enlace de hipertexto pueda apuntar a cualquier cosa, sea personal, local o global, sea un borrador o algo definitivo. Hay una segunda parte del sueño, que consiste en nuestras interacciones en línea, trabajar, vivir en sociedad y jugar. Podemos usar los ordenadores para analizar esas interacciones, encontrarle el sentido a lo que estamos haciendo, donde encajamos y como podemos trabajar mejor juntos. Con el flujo dramático de material de todo tipo que sufre la Web a partir de los 90's, la primera parte del sueño está cumplida. La segunda parte todavía no se cumple, pero hay signos y planes que nos hacen confiar. La gran necesidad de información acerca de información, para ayudarnos a categorizar, ordenar, pagar, y poseer información está derivando en lenguajes diseñados para que la Web sea procesada por máquinas, más que por gente. La Web de documentos legibles por humanos se está mezclando con la Web de datos entendibles por máquinas. El potencial de la mezcla de humanos y máquinas trabajando juntos puede ser inmenso."

("The World Wide Web: A very short personal history", Tim Berners-Lee, 1998)

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

AGRADECIMIENTOS

Me gustaría destacar por encima de todo, el gran esfuerzo, compañerismo y motivación que me han aportado varios de los miembros de mi grupo de investigación, durante las diferentes etapas de las que ha constado este trabajo.

Agradecer la gran labor de coordinación y metodología ejercida por mis codirectores Dr. Juan José Martínez Durá y Dr. Eduardo Carrillo Zambrano. Gracias por orientarme en aquellos momentos de dificultad.

Resaltar la figura de Eduardo Carrillo, ya no solo por su gran labor investigadora sino por su calidad como persona y por la gran amistad que a lo largo de varios años me ha sabido demostrar. Sin duda, su apoyo ha sido esencial para que esta investigación se hiciera realidad, dándome aliento cuando las fuerzas escaseaban. Gracias Eduardo.

No me gustaría olvidarme de agradecer a la persona de Gregorio Martín, por sus fructíferos comentarios, por expresarme sus inquietudes sobre la Web Semántica y los lenguajes de marcado y por servirme de guía en determinados momentos de mi investigación.

Gracias a Francisco Matas y Germán Herrero por ayudarme en la construcción del prototipo y en las diversas implementaciones desarrolladas. Ellos han supuesto una parte importante en esta realización, permitiéndome poner en práctica muchas de las ideas que iban surgiendo. Gracias, por aguantar mis ocurrencias y por hacer esas horas extra en algunos momentos.

Agradecer a Arturo Cervera (compañero de despacho), su enorme generosidad al ofrecerme su ayuda cuando lo necesitaba, por ayudarme en la implementación, y por darme ideas en determinados aspectos. Cuenta conmigo en tu etapa de doctorando.

Quisiera agradecer al doctorando y compañero V. Ramón Tomás por compartir conmigo sus ideas e inquietudes y por acogerme junto al Dr. Luís Amable García en su grupo de investigación *AIA-Applying Intelligent Agents* del Departamento de Ingeniería y Ciencia de los Computadores de la Universitat Jaume I de Castellón. Gracias a los dos.

Gracias a Leo van den Berg por hacer posible de igual forma, mi estancia en TRAIL (*The Netherlands Research School for Transport, Infrastructure and Logistics*) perteneciente a la *Technical University of Delft* (TU Delft). A su vez me gustaría agradecerle sus comentarios acerca de las tecnologías relacionadas con la Web Semántica y su visión de poder aplicarlas al tráfico.

A Manolo Cid y Vicente Lloréns por mostrarme el trabajo que estaban realizando en el proceso de reingeniería del Centro de Información de Tráfico de la DGT, y por atenderme en aquellos momentos de necesidad, incluso a veces telemáticamente.

Al LISITT y sus responsables por todas las facilidades dadas para poder llevar a cabo mi trabajo.

Estoy sumamente agradecido a todas las personas que he nombrado y a muchos que no he recordado al nombrar. A todos, gracias.

Javi.

RESUMEN

Esta tesis integra dos áreas científicas: por una parte la Ingeniería de Tráfico y por otra las Ciencias de Computación.

Dentro de la Ingeniería de Tráfico, forma parte de los denominados Sistemas Inteligentes de Tráfico (ITS) los cuales se basan en la aplicación de tecnologías de Telecomunicaciones e Informática (Telemática) a los sistemas de ayuda al tráfico vial, perteneciendo al dominio "Servicios de Información al Viajero" (TIS).

Como se pone de manifiesto en esta tesis, existen algunos problemas derivados del uso de este tipo de sistemas, desde el punto de vista del usuario, que dan lugar a que éste sea incapaz de obtener la información clara y precisa correspondiente a sus requerimientos.

Los principales problemas identificados han sido la falta de un vocabulario común de términos que haga uso de una semántica bien definida que permita obtener el significado (incluso aquél no explícito) de cada concepto de tráfico vial, así como la recuperación de información desde distintas fuentes heterogéneas, de una forma intuitiva y sencilla.

Varios aspectos describen la problemática actual relativos al acceso y distribución de la información:

- Un gran volumen de información de tráfico se distribuye entre varios sitios web. El principal problema para un usuario que necesita información de este tipo es encontrar estos sitios web y además tratar con los diferentes accesos a ésta, así como sus diferentes formas de presentación.
- Por otra parte, un usuario puede necesitar información de diferente naturaleza o tipo, y por lo tanto el almacenaje de toda esta información en un solo sitio web no es viable a nivel de costes de almacenamiento ni incluso a nivel de mantenimiento.
- El tratamiento de la información no permite realizar inferencias sobre ella de manera que pueda ser obtenido como resultado, información que a priori no estuviera explícitamente detallada.

Tras el descubrimiento de la problemática existente y la identificación de elementos que pudieran solventar esta situación, el presente trabajo se centra en el estudio de los sistemas ITS orientados a facilitar información de tráfico al usuario, y que a su vez permitan la gestión de la información, su tratamiento e intercambio, de manera eficaz entre los diferentes elementos que componen la arquitectura de servicios de información de tráfico, como los usuarios, aplicaciones, y proveedores de la información.

Para la consecución de los objetivos marcados han sido desarrollados diversos elementos como parte integrante de una arquitectura, que supusieron determinados aportes y/o resultados:

- **Construcción de una infraestructura ontológica cuyo dominio queda enmarcado en la información sobre tráfico vial**. Las ontologías construidas forman el núcleo principal del trabajo, ya que por una parte, constituyen la ontología de dominio utilizada como soporte en el emparejamiento de servicios, y a su vez, conforman un mecanismo de descripción de los diferentes servicios de información de tráfico, los cuales son especificados mediante las referencias a conceptos descritos en estas ontologías. La infraestructura ontológica creada puede considerarse una base sólida en el desarrollo de un futuro y completo vocabulario semántico a utilizar por las Administraciones de Tráfico.
- Se ha expuesto un **marco de trabajo para la conversión de portales Web convencionales de información en Servicios Web Semánticos (SWS)** de manera que la información aportada pueda ser almacenada con la adición de significado consiguiendo a su vez, potenciar nuevas capacidades que en un principio no existían.
- **Extensión a las metodologías de construcción de ontologías, previamente existentes, para plantear el proceso a seguir en la obtención de un modelo semántico formal a partir de un modelo de Entidad-Relación (ER).** En la presente aproximación, se han integrado diferentes aspectos de las metodologías actuales así como aquéllos surgidos de lecciones aprendidas de la experiencia. Este método fue aplicado en la construcción de la infraestructura ontológica creada, y puede servir como guía de desarrollo en construcciones ontológicas similares.
- Se ha diseñado, implementado y evaluado, un **algoritmo de emparejamiento de SWS**, que aprovecha al máximo las capacidades proporcionadas por las ontologías de descripción de servicios y que constituye una mejora de los ya existentes. En este algoritmo de emparejamiento se amplió el rango de grados de similitud, principalmente con la consideración de nuevas relaciones en la taxonomía de la ontología de dominio utilizada como soporte en el emparejamiento. Inclusión del **grado fraternal o de hermanos y otra serie de modificaciones**.

Se han mostrado una serie de propuestas que permiten crear y manejar **servicios web semánticos de información de tráfico vial**: nuevo **parámetro no funcional, ontología de categorización de servicios de tráfico y elección de factores de medida en la QoS.**

Finalmente las propuestas han sido validadas a partir del desarrollo de un prototipo software basado en la **arquitectura de integración** de **SWS de información sobre tráfico vial**, que ha permitido el ensamblaje de todos los elementos (**ontologías de tráfico**, **algoritmo de emparejamiento**, **servicios de información de tráfico**), obtenidos como resultado en cada una de las etapas de esta investigación. Haciendo uso del prototipo construido, se ha evaluado y verificado la funcionalidad de todo el conjunto mediante el estudio de las relaciones entre cada uno de los elementos integrantes y los resultados obtenidos.

SECCIÓN I: INTRODUCCIÓN

CAPÍTULO 1

INTRODUCCIÓN: APROXIMACIÓN AL PROBLEMA

1.1 INTRODUCCIÓN

Dentro de la amplia gama de información de interés para la sociedad se encuentra la información de tráfico, la cual comúnmente es organizada y presentada por diversos organismos de gestión de tráfico y a su vez divulgada por los diferentes medios: prensa, radio y televisión, portales web especializados, etc. Dichos organismos han generado contenidos accesibles a través de Internet, relacionados con este tipo de información, lo que ha posibilitado a los usuarios de ésta la planificación de itinerarios libres de incidencias. Este objetivo ha sido conseguido gracias al conocimiento del estado actual de las vías y las posibilidades de uso de rutas alternativas, así como la existencia de información meteorológica y otros rasgos condicionantes.

Hoy en día se ha convertido en algo esencial la capacidad de intercambio de información entre diferentes sistemas. La principal razón reside en la creciente demanda de nuevos servicios de información, los cuales a su vez son dependientes de sistemas con distintas características. Además, resulta importante poder usar aquella información que ya está almacenada en repositorios heterogéneos con objeto de desarrollar estos servicios.

Los Sistemas Inteligentes de Transporte y Servicios (ITS) se constituyen en un tipo de aplicación donde es realmente importante conseguir intercambiar información de manera fiable y con calidad. Estos sistemas se basan en la posibilidad de ofrecer información de tráfico a conductores y viajeros de manera que puedan planificar sus actividades antes o durante la realización de un viaje. Este trabajo forma parte de este tipo de aplicaciones y además dentro del dominio multimodal de los sistemas de transporte (tierra, mar, aire) se centra en el tráfico vial, tomando como punto de partida para delimitar el problema, la consideración del problema de la distribución y acceso a la información “previaje” y al manejo de “sucesos”[1] principalmente.

En 1997 la Comisión Europea encomendó por medio de la organización ERTICO (sociedad europea encargada de la puesta en práctica de los Sistemas

[1] Eventos que se producen en una localización geográfica y que implican o pueden implicar una alteración del estado de la vía o de la seguridad vial.

Inteligentes de Transporte y Servicios) un estudio económico para valorar el futuro de mercado de las aplicaciones ITS. Este estudio reveló que en ocho países europeos, el mercado anual previsto (en condiciones de saturación) para el sector de servicios y equipamientos de automóvil, sería en el futuro de 56.6 billones de euros por año, en Europa. De éstos, 26 billones estarían dedicados al mercado de servicios avanzados de información al viajero, en los que se incluyen los sistemas de navegación dinámica y la información de tráfico a bordo del vehículo [ERTICO98]. Algunos de los proyectos tienen como objetivo proporcionar al conductor toda la información que necesite, referente a condiciones climatológicas, posibles incidencias que puedan presentarse, aparcamientos disponibles, localización de hoteles y restaurantes etc.

1.2 MOTIVACIÓN

La participación en varias actividades relacionadas con el tráfico, me ha permitido conocer en profundidad la problemática existente en estos sistemas y la forma en que las administraciones europeas han asumido este reto, sus esfuerzos y sus deficiencias. A continuación se tratará de revisar todas aquellas experiencias que me han motivado de alguna forma en la realización de esta tesis.

El grupo LISITT (Laboratorio Integrado de Sistemas Inteligentes y Tecnologías de la Información de Tráfico) como parte integrante del Instituto de Robótica, persigue como metas el promocionar el uso de la Telemática en el Tráfico y Transporte y asesorar sobre su conveniencia, tanto a administraciones públicas como a empresas privadas y por otra parte, ayudar y colaborar a través de proyectos I+D y estudios con estas entidades. Por lo que mi participación en este grupo desde marzo de 1996 ha sido una constante fuente de conocimiento y experiencia en el área de la Telemática aplicada al transporte.
Durante este periodo he participado en los proyectos europeos ARTS (*Advanced Road Traffic in South-West*) y SERTI (*Southern European Road Telematics Implementation*) y mayoritariamente dentro del dominio denominado “Servicios de información al viajero”, en el subdominio 4.6 “Servicios basados en Internet y Telecomunicaciones” en actividades como las que a continuación cito:

- Servicio WAP de Información de tráfico para el Servei Català de Trànsit.
- Rediseño del Servicio WAP para la Dirección General de Tráfico.
- Desarrollo de un portal multilingüe de información de incidencias para PDAs.
- Desarrollo de un servicio de alertas de incidencias de tráfico basado en mensajes de voz, e-mail y SMS para ambas administraciones.

De igual manera, también ha supuesto la participación en proyectos nacionales como la CICYT: “Desarrollo de un sistema de intercambio de información entre camiones y dispositivos externos para control de mercancías mediante el uso de una infraestructura conceptual basada en ontologías”.

Por otra parte, otra fuente de motivación ha sido la participación como miembro experto en ISO TC204/WG1 *Architecture, Taxonomy & Terminology* y concretamente en “Guía de uso de XML (*eXtensible Markup Language*) en la normalización de temas relacionados con ITS”, ejerciendo tareas de supervisión en las distintas normas y

decisiones.

En cuanto a la posibilidad de realizar una investigación relacionada con otras áreas, la visión que sobre la Web Semántica (WS) estableció Tim Berners Lee [Ber01], fundador y director de W3C[2], los numerosos artículos y trabajos que han ido apareciendo después y la pertenencia del Instituto de Robótica como miembro del consorcio en noviembre de 2002, me han motivado especialmente en este campo y la evaluación de su posible aplicación al entorno de los ITS. La lectura de autores como Haustein, el cual asevera en [Hau02] la necesidad de desarrollar aplicaciones reales basadas en Web Semántica como realización necesaria para que esta tecnología prospere, reafirmaron mis ideas iniciales.

1.3 PROBLEMÁTICA EN TRÁFICO

Debido al vertiginoso ritmo de aparición y posicionamiento de las tecnologías, tanto en el nivel de las comunicaciones como en el de tráfico, se han cometido diversos errores en cuanto a la generación de los contenidos publicados en la Web, y la forma en la que son transmitidos a los usuarios. Estos errores han constituido duplicidad de esfuerzos al desarrollar diversas aplicaciones entre diferentes administradores, sin una estandarización como base y generalmente asociada con el desarrollo de contenidos basados en lenguajes de marcado como HTML. Esto ha dificultado procesos deseables como el compartir la información al tratar con vocabularios no comunes entre los diferentes elementos involucrados, y que al final dan como resultado la percepción por parte del usuario de información diversa y difusa.
A continuación se describen cada uno de los problemas que bajo mi perspectiva, caracterizan la problemática de la difusión de información de tráfico vía Web:

Problema 1: Ausencia de vocabularios comunes

En el desarrollo de las distintas aplicaciones de tráfico es característico la ausencia de vocabularios comunes que puedan ser elegidos como núcleo y punto de partida para desarrolladores de contenidos. Por ejemplo, en el caso de las aplicaciones sobre gestión de incidencias de tráfico, no existen tales vocabularios comunes que sean asociados a los descriptores o *metadatos*[3] de la información que deben de desarrollar las aplicaciones, con lo que los desarrolladores pueden llegar a describir la información (por ejemplo de los vehículos implicados en un accidente) de diferente forma, inclusive entre desarrolladores de una misma empresa de desarrollo y más aún entre diferentes administraciones y países.

Problema 2: Múltiples fuentes y formatos

Existe una distribución inherente de los datos e información repartidos entre fuentes muy diversas como bases de datos y portales web entre otras. Incluso los documentos públicos aportados por las diferentes entidades, aparecen en formatos tan

[2] El W3C (World Wide Web Consortium) se creó en octubre de 1994 con el fin de intentar dar el máximo potencial desarrollando protocolos que promuevan su evolución y aseguren su interoperabilidad. W3C tiene alrededor de 500 organizaciones miembro por todo el mundo y se ha ganado el reconocimiento internacional por sus contribuciones al crecimiento de la Web.
[3] Los metadatos son datos acerca de datos, y denotan cualquier tipo de conocimiento que puede usarse para conseguir información sobre la estructura y el contenido de una colección de documentos.

diversos como puedan ser HTML, XML, WML, VoiceXML etc. y a su vez dicha información debe ser tratada por diferentes usuarios como puedan ser las propias administraciones de tráfico o por los conductores o viajeros. En el dominio de los ITS el tratamiento de la información y las consecuencias de este tratamiento, como la toma de decisiones o la simple consulta de este tipo de información han sido hasta la fecha altamente dependientes del conocimiento humano.

Problema 3: Gran volumen de información distribuido

La difusión de información de tráfico abarca un gran abanico de posibilidades que hacen que el tratamiento de la información se deba abordar de manera eficaz. El usuario se ve obligado a tratar con un gran volumen de información que aparece a su vez distribuida por los diferentes *portales web* de administraciones o entidades privadas que la ofrecen. A su vez la aparición de múltiples dispositivos de usuario final ha dado lugar a que desde el punto de vista del usuario haya una gran dificultad para encontrar estos lugares y tratar con las diferentes representaciones y accesos a la información.

Problema 4: Imposibilidad de almacenaje de toda la información

Por otra parte, un usuario puede necesitar información de diferente naturaleza, y por lo tanto el almacenaje de toda esta información en un solo sitio web no es viable a nivel de costes de almacenamiento ni incluso a nivel de mantenimiento.

Problema 5: Carencia de descripciones semánticas y de servicios web

Tal y como se describe en el problema 1, los vocabularios o lenguajes que describen conceptos y estructuras de datos relacionados con tráfico suelen ser Ad Hoc para cada aplicación siendo además descripciones de tipo sintáctico, carentes de semántica. Hasta donde llegó la búsqueda y revisión bibliográfica de esta tesis, no se encontraron ontologías o vocabularios con valor semántico que aporten significado a los conceptos y sus relaciones. Tampoco se conoce de la existencia de Servicios Web (SW) de información de tráfico.

Problema 6: Solamente es posible obtener conocimiento explícito

Este problema es derivado del anterior. El tratamiento de la información no permite realizar inferencias sobre ella de manera que pueda ser obtenida como resultado, información que a priori no estuviera explícitamente detallada.

1.4 UNA PRIMERA ALTERNATIVA

Con la aparición de Internet, el acceso a la información ha cobrado una especial importancia, especialmente debido a que las barreras existentes respecto a la distancia han tomado una nueva connotación, al permitir el acceso a la información desde casi prácticamente cualquier medio y parte del mundo a cualquier persona con bajo coste temporal y económico.

Con este auge y con las facilidades dadas por la Web, tal y como fue concebida en sus inicios por Tim Berners Lee a través de sus conceptos y herramientas relacionadas con la creación de la *World Wide Web* (WWW) y concretamente el lenguaje de marcado HTML para presentar información, se facilitó que cualquier persona fuese potencialmente usuario de este medio y permitiese tanto el acceso como la publicación de información, de tal forma que los conocimientos fueran compartidos entre todo un

universo de personas y medios.
En principio, el lenguaje usado fue un lenguaje basado en etiquetas predefinidas con las que el autor de la publicación generaba documentos presentados mediante un formato establecido por él. El propio documento contenía tanto la información que se pretendía mostrar como su formato. Esto creó un caos inicial que solamente unos años después fue percibido.

Según Ivan Herman [Her03] "la Web Semántica es una infraestructura basada en metadatos que permite razonar sobre los contenidos de la Web".
Tal y como aparece reflejado en [Mal02], está previsto que la WS sea un medio donde los datos puedan ser compartidos y procesados tanto por herramientas de manera automatizada como por humanos. La clave subyace en la automatización y la integración de los procesos a través de lenguajes legibles por máquinas. Con objeto de poder usar y enlazar la gran cantidad de información disponible en la Web, los agentes software[4] deben de ser capaces de comprender la información, es decir, los datos deben de estar escritos haciendo uso de una semántica. Por tanto, en los documentos XML o HTML, deberá de añadirse semántica adicional o metadatos para que los programas software puedan establecer el significado de las etiquetas de dichos documentos.

Para conseguir que los computadores sean mucho más útiles, la WS extiende la Web actual con conocimiento formalizado y datos que son procesados por ordenadores. Para ser capaz de buscar y procesar información relativa a alguna materia de interés, los programas necesitan información que haya sido modelada de una forma coherente. Una ontología modela todas las entidades y relaciones en un dominio. La ontología es necesaria para la representación del conocimiento. La clave de las ontologías es que pueden ser compartidas y por lo tanto incrementan en eficiencia e interoperabilidad.

La WS aporta otros elementos los cuales están ahora en fase de estudio y desarrollo:

Lógica: La posibilidad que brinda la lógica para que los ordenadores puedan razonar (mediante inferencia) haciendo uso de reglas.

Pruebas: Una vez construidos los sistemas que siguen la lógica, adquiere sentido usarlas para establecer pruebas. La gente alrededor del mundo escribirá sentencias lógicas, por lo que las máquinas podrán seguir los diferentes enlaces semánticos para construir pruebas.

Confianza: La idea de la WS es buena, pero bastante inútil si cualquiera puede decir algo. ¿Quién confiaría en tal sistema? Debido a la regla implícita de que "cualquier cosa puede decir cualquier cosa sobre cualquier cosa", esto se convierte en algo difícil de parar. Por este motivo se necesitan mecanismos que aseguren que lo dicho es cierto. Por ejemplo, el mecanismo de la firma digital, proporciona la prueba que cierta persona escribió (o está de acuerdo con) un documento o declaración. Otros mecanismos como la Web de Confianza, en la que se crean relaciones de confianza entre autores a través de la red es otra de las aportaciones a este campo.

Como ya ha sido puntualizado en la exposición de los problemas, actualmente

[4] Un **agente** es una entidad de software que funciona continua y autónomamente en un medio particular a menudo habitado por otros agentes y procesos, sin requerir de guía constante o intervención humana.

existen solo descripciones sintácticas y estructuras de datos relacionados con tráfico, carentes de semántica.

Por otra parte tampoco se encontraron SW de tráfico, solamente portales web. Generalmente lo que ofrecen las administraciones de tráfico en sus portales web no son *servicios web*[5] en sí, sino sitios web que en algunos casos poseen formularios para poder interactuar con una base de datos y que dependiendo de los valores de entrada devolverán una información u otra. Estos portales no están capacitados para recibir peticiones de invocación mediante protocolos de comunicación, por lo que deben de sufrir una adaptación y además necesitan del desarrollo de aplicaciones software que puedan extraer de ellos el conocimiento. La principal diferencia entre portal web y servicio web es el objeto de la interacción, ya que mientras en los primeros ésta se produce con los datos en las páginas, en los segundos se produce a través de aplicaciones.

1.5 PREGUNTAS DE INVESTIGACIÓN

Por todo lo expuesto anteriormente surgen algunas cuestiones interesantes como base de una investigación exhaustiva:

- ¿Qué vocabularios comunes pueden ser definidos para representar información de tráfico en lo relacionado con accidentes, medidas, localizaciones, tipos de vehículos etc.?

- ¿Qué principios, tecnologías y herramientas de representación del conocimiento y de definición de ontologías son las más apropiadas para abordar este problema?

- ¿Es posible obtener un marco de referencia que ayude en la descripción y posterior uso de información distribuida a través de páginas web tradicionales, para convertir éstas en SWS?

- ¿Qué mecanismos pueden utilizarse para permitir que cualquier usuario acceda de forma transparente o a través de interacciones sencillas a la información deseada por él, expresada según sus requerimientos?

1.6 ORGANIZACIÓN DE LA MEMORIA

Para dar respuesta a los objetivos previos la memoria ha sido dividida en cinco secciones principales:

[5] Un ***Servicio Web*** es un sistema de software diseñado para soportar la interacción entre dos máquinas a través de una red. Posee un interfaz descrito en un formato que puede ser procesado por una máquina (específicamente WSDL). Otros sistemas pueden interactuar con el Servicio Web en la manera prescrita por su descripción utilizando mensajes SOAP, típicamente transportados utilizando HTTP y serializados con XML, en conjunción con otros estándares relacionados con la Web. [W3C04]

En el capítulo 2 de la **presente sección**, se describen los principales objetivos a cubrir, mediante la exposición del objetivo general y de los diferentes objetivos específicos en los que podemos descomponer el primero. Por último se describe la metodología de investigación seguida en la realización de esta tesis.

En la **sección segunda** se describe el marco teórico, las tecnologías involucradas y el estado del arte.

El capítulo 3 incluye una visión global del concepto de ITS y sus efectos en el campo de transporte, para posteriormente hablar de los distintos objetivos a conseguir mediante el uso de las tecnologías ITS. Dentro de este capítulo, se detalla el dominio de aplicación en el que se enmarca esta tesis.

El capítulo 4 explica el concepto de representación de conocimiento, y el uso de ontologías que permitan compartir y reutilizar bases de conocimiento, haciendo uso de semántica. Se expone en que consiste un sistema terminológico y las principales diferencias con las bases de datos tradicionales. Posteriormente, se parte de la descripción de la propuesta del W3C denominada "Fundamentos de Ingeniería de la Web del Mañana", para a continuación presentar los conceptos básicos, lenguajes, herramientas existentes y estado actual de la WS, SW, Sistemas Multiagente y SWS. En relación a éstos últimos se presentan las principales tecnologías y arquitecturas, haciendo hincapié en el estudio de las propuestas de arquitecturas independientes de lenguaje para su desarrollo. Por último, se presentan los conceptos de matching (emparejamiento) semántico, grados de similitud o match así como los diferentes sistemas desarrollados que hacen uso de este tipo de emparejamiento.

En la **sección tercera**, se describen las principales propuestas y contribuciones.

En el capítulo 6 se explican los detalles de desarrollo de un esquema de representación del dominio particular "información de tráfico vial ofrecida al viajero a través de servicios basados en Internet", con una semántica bien definida. El uso de esta semántica nos permitirá obtener un conocimiento formalizado que hará posible el desarrollo de una arquitectura de integración de SWS de información de tráfico. También se detalla una propuesta de mejora de las metodologías existentes en la construcción de ontologías basadas en la experiencia.

En el capítulo 7 se describen las principales características del algoritmo propuesto para el emparejamiento de Servicios Web Semánticos. Finalmente, se describe su implementación así como las diferentes pruebas orientadas a la comprobación funcional del emparejador semántico y su comparación con un emparejador semántico de referencia en la mayoría de la literatura.

En el capítulo 8 se propone un marco de trabajo para la conversión de portales Web convencionales de información de tráfico en SWS. Para describir ésta propuesta se resume el proceso planteado mediante la transformación del sitio de información sobre el estado de la red de carreteras de España, perteneciente al portal de información sobre tráfico vial de la DGT. A su vez se proponen diferentes mejoras en las ontologías de descripción, como la adición de nuevos parámetros, una escala Ad Hoc de QoS para

este tipo de servicios, así como el desarrollo de una ontología de categorías de SW de tráfico que nos permite establecer una jerarquía de servicios atendiendo a su tipo.

La **sección cuarta** presenta otros resultados así como las pruebas utilizadas para validar las contribuciones generadas.

En el capítulo 9 se explica en que ha consistido el desarrollo y el uso de un entorno de edición de perfiles de servicio de tráfico, que adicionalmente integra la visualización y verificación de consistencia de ontologías que definen los conceptos con los cuales interactúa un servicio Web.

En el capítulo 10 se describe el sistema en general, entendido como un sistema multiagente orientado a facilitar, asistir, y optimizar los procesos de anuncio, descubrimiento e invocación de SW relacionados con el área de tráfico vial. Finalmente, se describen las pruebas realizadas que buscaban la comprobación de la integración del emparejador en la plataforma de agentes y su uso en un caso real.

En la **sección quinta** se detallan las conclusiones alcanzadas, aspectos de discusión y las recomendaciones para futuras investigaciones.

Finalmente se muestran las referencias bibliográficas, las publicaciones relacionadas y derivadas de la investigación llevada a cabo y anexos.

CAPÍTULO 2

OBJETIVOS Y METODOLOGÍA

Habiendo sido determinada la problemática referente a la difusión de información de tráfico vial vía Web, e identificando las cuestiones relativas a la posible solución a ésta, a continuación se describen los principales objetivos a cubrir, mediante la exposición del objetivo general y de los diferentes objetivos específicos en los que podemos descomponer el primero. Por último se describe la metodología de investigación seguida en la realización de esta tesis.

2.1 OBJETIVO GENERAL

El objetivo principal de esta tesis es proponer un modelo de integración basado en Web Semántica para el desarrollo de SW de información de tráfico, a partir del manejo de distintas fuentes de información (por ejemplo, diferentes administraciones u organismos) como recursos Web. Se tratará de establecer una descripción de los servicios aportados, así como hacer posible la interacción con dichos servicios de forma que el usuario satisfaga sus requerimientos y pueda interactuar con dichos recursos (previo tratamiento de la información) de manera relativamente sencilla mediante el uso de interfaces adaptables al servicio y sin preocuparse de donde pudiera encontrar ésta.

2.2 OBJETIVOS ESPECÍFICOS

Los objetivos específicos marcados fueron los siguientes:

- Construir ontologías de tráfico vial aplicadas a nuestro objetivo (descripción semántica de las capacidades de un servicio de información de tráfico), dada la carencia de éstas en la actualidad, que podamos tomar como base para nuestra investigación.

- Extender las metodologías de desarrollo de ontologías actuales con el fin de generar una propuesta metodológica para el diseño e implementación de SWS de información de tráfico, a partir de los modelos de datos semánticos ya existentes.

- Diseñar, implementar y evaluar a partir de un análisis previo, un algoritmo para el emparejamiento de SWS, que aproveche al máximo las capacidades proporcionadas por la ontología de descripción de servicios y que constituya una mejora de los ya existentes. El principal punto de investigación se centrará en los diferentes grados de similitud que permitirán emparejamientos flexibles, el

uso de parámetros no funcionales, así como la introducción de diversos filtros, algunos de los cuales no han sido considerados hasta la fecha, en trabajos de otros autores.

- Exponer un marco de trabajo para la conversión de portales Web convencionales de información de tráfico en SWS de manera que la información aportada por ellas pueda ser almacenada con la adición de significado y a su vez se consiga potenciar nuevas capacidades que en un principio no existían.

- Presentar y validar una arquitectura de integración de SWS de información sobre el tráfico vial mediante la construcción de un prototipo software capaz de hacer uso del algoritmo para el emparejamiento propuesto, así como la implementación de éste como un componente dentro de un sistema multiagente que permita la publicación, descubrimiento e invocación de los servicios.

2.3 METODOLOGÍA DE INVESTIGACIÓN

Esta investigación ha sido de tipo experimental, poniendo en práctica métodos así como herramientas que fueron aplicadas en distintos bancos de pruebas, mediante la incorporación de nuevas ideas.

La metodología principalmente siguió las fases que aparecen en la figura 1.1:

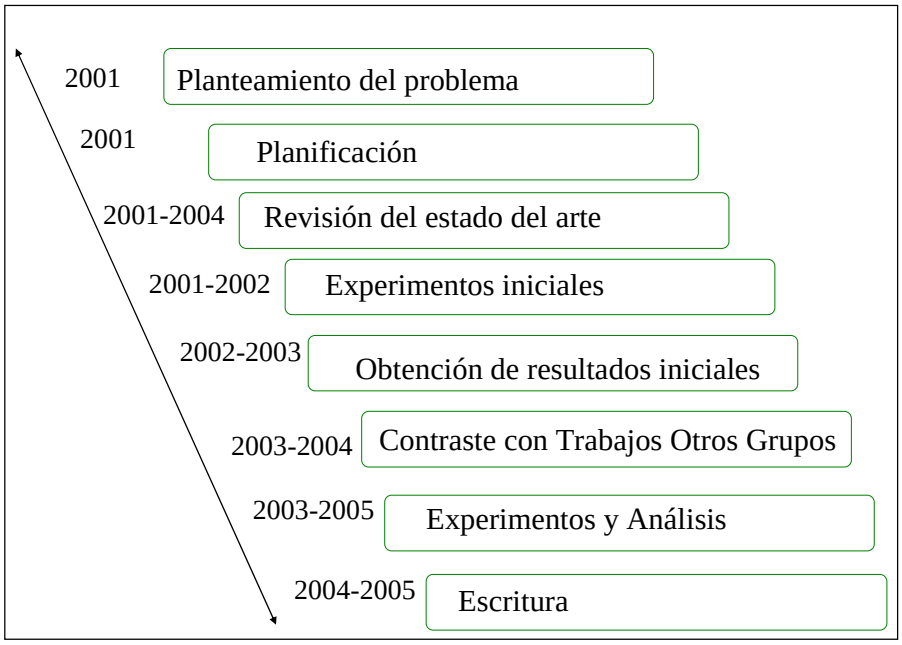


Figura 1.1 Metodología de Investigación

A partir del planteamiento y posterior planificación, la labor de investigación llevada a acabo ha tenido un carácter de revisión de todas aquellas tecnologías involucradas en el proceso, sobre todo aquéllas ligadas a la WS y concretamente a los SWS. En esta fase ha sido muy importante establecer un proceso de elaboración del estado del arte a través

de la búsqueda y la revisión bibliográfica. Este proceso fue llevado a cabo de forma permanente durante el resto de fases, y consistió principalmente en el estudio de recomendaciones W3C, libros, revistas, sitios web de congresos, revisión de proceedings de conferencias especializadas, asistencia a cursos de actualización como "Agentes Buscadores en Internet" en la Universidad Carlos III de Madrid, revisión de materiales del grupo ISO TC204 y material disponible en la Web proporcionado por investigadores.

Esta investigación fue llevada a cabo dentro del grupo de investigación LISITT del Instituto de Robótica. Fruto de este grupo de investigación se han definido distintas contribuciones que han dado pie a varias publicaciones, así como también a la finalización de varios proyectos final de carrera que constatan la validez de este trabajo.

El estudio, evaluación, utilización y prueba de tecnologías emergentes, dio lugar a la aparición de infinidad de problemas en cuanto a la estabilidad de las herramientas, lo que provocó muchas veces un avance en falso. En determinadas ocasiones muchas de las herramientas fueron optimizadas por nosotros mismos, y en otros se requirieron consultas con los autores de éstas, a través de e–mails y participación en foros (Sean Bechhofer, Alan Rector, Massimo Paolucci, David Martin, Sheila McIlraith, Ian Horrocks, Bijan Parsia etc.). A su vez, fueron desarrolladas nuevas herramientas Ad Hoc ante la incapacidad de adaptar las ya existentes, como fue el caso de la interfaz de cliente y parsers.

En cada parte del proceso se desarrollaron pruebas en distintos escenarios, ya que el sistema final requería que se cumpliesen diversos objetivos parciales bajo diferentes situaciones. Una vez realizado el estudio, el análisis y la evaluación de las diferentes alternativas se generó una propuesta de integración que fue depurada en diversas etapas.

La naturaleza del tema tratado, en constante evolución, hizo necesario que el trabajo realizado se ejecutase en un entorno que facilitase el intercambio de los conocimientos con otros grupos. En este sentido se buscó una estrategia basada en fomentar los siguientes aspectos:

1) La interacción con diferentes grupos de discusión ha sido una de las estrategias seguidas, para poder estar al día de cualquier avance en el campo objeto de investigación. Por ejemplo, en el desarrollo de ontologías y tras numerosos problemas en la edición de éstas, debido principalmente a limitaciones de las herramientas de edición, se vio la necesidad de establecer contacto con miembros de W3C y concretamente con el grupo de WS, para intentar solventar cada problema puntual que se presentaba. A continuación se citan algunos de los grupos o listas de discusión consultados:

    - **public-sws-ig@w3.org**: lista de correo de SWS del W3C, *Semantic Web Services Interest Group*.
    - **www-ws@w3.org**: lista de correo de Web Services en W3C (a pesar de ser de servicios web, son tratados temas concernientes a Web Semántica)
    - **semantic-web@w3.org**: lista de correo sobre Web Semántica en W3C
    - **www-rdf-interest@w3.org** : lista de interés en RDF del W3C

- **www-rdf-logic@w3.org**: foro para discusiones técnicas concernientes al diseño de lenguajes basados en lógica para su uso en la Web.
- **oiled-discussion@lists.manchester.ac.uk**: lista sobre el editor OilEd
- **protege-discussion**, **protege-users**. **protege-owl**: listas del editor Protégé
- **jena-dev@yahoogroups.com**: lista de discusión de Jena
- **webservices-latinos@yahoogroups.com**: lista de correo española sobre los SW y los SWS

2) En ésta misma línea, la lectura de artículos relacionados de revistas especializadas y congresos, junto con la publicación de los resultados alcanzados en cada fase del desarrollo e investigación y la participación en foros especializados, sirvió para contrastar y evaluar la idoneidad de nuestro trabajo. De esta forma teníamos las certeza de haber tomado decisiones acertadas o por el contrario de no haberlas tomado.

3) Por último, se utilizó como un elemento importante dentro del proceso, la participación en diversos grupos de investigación, lo que permitió contrastar y discutir ideas y planteamientos con diversos investigadores. Para abordar este aspecto se realizaron estancias de investigación en la Universidad Jaume I de Castellón como investigador visitante dentro del grupo *AIA-Applying Intelligent Agents* del Departamento de Ingeniería y Ciencia de los Computadores, así como en TRAIL (*The Netherlands Research School for Transport, Infrastructure and Logistics*) perteneciente a la *Technical University of Delft* (TU Delft). Estas estancias me permitieron consolidar mi trabajo de investigación en SWS y su extensión e integración de ontologías de tráfico vial usando sistemas multiagente para la gestión de SW de tráfico.

SECCIÓN II: FUNDAMENTOS TEÓRICOS, TECNOLOGÍAS Y ESTADO DEL ARTE

CAPÍTULO 3

SISTEMAS INTELIGENTES DE TRANSPORTE

3.1 RESUMEN

La evolución de las nuevas tecnologías de información en la última década ha permitido el desarrollo de los Sistemas Inteligentes de Transporte (ITS) que han influido notablemente en el tráfico vial.

Desde el punto de vista de los usuarios, el estar el menor tiempo en el coche para realizar un recorrido o conocer la información acerca del estado del tráfico tanto antes de salir o una vez en ruta es cada vez más importante.

En este capítulo se pretende dar una visión global del concepto de ITS y sus efectos en el campo de transporte, para posteriormente describir los distintos objetivos a conseguir mediante el uso de las tecnologías ITS. Tomando como base estos objetivos, a continuación se expone el trabajo efectuado por ISO en cuanto a una categorización de servicios de tráfico así como sus propuestas de hacer uso de XML y de SW en cualquier desarrollo ITS. Como ejemplos de uso de lenguajes de marcado en tráfico, son mostradas algunas de las iniciativas surgidas en diversos campos como la información, modelado etc. Por último se describe el programa MIP y sus diferentes proyectos euro-regionales que son aplicados en distintas áreas geográficas y dominios, así como los diferentes métodos o sistemas de difusión de información de tráfico vial.

3.2 CONCEPTO DE ITS

El conjunto de tecnologías conocidas internacionalmente como ITS están orientadas prioritariamente a mejorar la movilidad de personas y mercancías en las infraestructuras viarias existentes, a la vez que representan un ahorro importante a la hora de concebir y/o construir nuevas infraestructuras [Eur04], [ITS04]. Al mismo tiempo, en su aplicación se consiguen importantes beneficios para el conjunto de la sociedad.

Debido a los problemas que acontecen, así como los elevados costes externos que caracterizan la situación de tráfico, el mundo de las telecomunicaciones y la informática, y por tanto la integración de ambos campos (telemática), ha surgido durante la última década como posible remedio a estos inconvenientes. El sector de la telemática se considera una pieza clave en el puzzle que hace realidad la aplicación de nuevas medidas correctoras en los problemas provocados por el tráfico (urbano, aéreo, marítimo y ferroviario).

Con el fin de adoptar una política común en lo referente a las aplicaciones telemáticas, la Unión Europea pone en marcha la creación de los “Programa Marco” donde se

remarcan aspectos tales como el desarrollo sostenible, la interoperabilidad, la acción global europea y la necesidad de involucrar a todos, de manera que las autoridades competentes y los sectores públicos o privados de los diferentes países, caminen hacía el futuro con un mismo objetivo, el progreso común.
A pesar de que España no se encuentra a la cabeza en cuanto en lo que se refiere a la aplicación de sistemas telemáticos, es en lo referente a la investigación y desarrollo de sistemas inteligentes donde España si adquiere toda la atención dentro de la Unión Europea.

Los sistemas ITS persiguen una utilización más eficiente de las infraestructuras del transporte en todos los campos: el de la seguridad vial, el de la reducción de costes de congestión y tiempos de recorrido, el de la mejora medioambiental, el de la comodidad del usuario del transporte, etc., todo ello bajo un doble enfoque de utilidad: la individual y la colectiva.

Entre los objetivos que se ha propuesto la Unión Europea en la política de implantación de ITS se encuentra el aumentar los niveles de seguridad, eficiencia e interacción medioambiental de los sistemas de transporte, obtener beneficios económicos en forma de ahorros en congestión y accidentalidad, y crear nuevos mercados para los sistemas telemáticos. Es importante reseñar que se ha conseguido obtener beneficios contundentes sobre diversos aspectos en aquellos países en los que se lleva algún tiempo aplicando estas tecnologías, como EEUU o Japón por ejemplo, mayores cuanta más implicación en ITS ha habido. Dichos aspectos, entre otros, son los siguientes [ITS04]:

- Los accidentes se han reducido entre un 20 y un 50%.
- Los tiempos de viaje en los desplazamientos que se hacen en calles principales y carreteras urbanas se han reducido entre un 8 y un 48%.
- La capacidad vial se ha incrementado entre un 8 y un 32%.

Como factor realmente importante que ha hecho decantarse por el uso de ITS es que la implantación de estos sistemas puede ser hasta un 35 % más económica que la ampliación de la red viaria existente. Dos factores han incidido de manera directa en el abaratamiento de las tecnologías: el desarrollo fulgurante de las telecomunicaciones, lo que implica que la oferta de las tecnologías de información sea cada vez más amplia, y la tendencia de los gobiernos a liberalizar el sector de las telecomunicaciones [UPM00a].

En definitiva, con la aplicación de las tecnologías ITS se pretende conseguir mejoras en cinco puntos fundamentales, que representan una problemática común en todos los sistemas de transporte [ITS04]:

- Incrementar la eficiencia y la capacidad del sistema de transporte que permitan reducir las congestiones entre otros factores.
- Mejorar la movilidad personal y la conveniencia y comodidad del sistema de transporte.
- Mejorar la seguridad de todo el sistema de transporte.
- Reducir el consumo de combustible mediante y los costes ambientales, decrementando el impacto sobre el medio ambiente.

- Mejorar las condiciones de productividad económica presentes y futuras de todos los ciudadanos, las organizaciones y el sistema económico en general.

Sobre la base de estos cinco objetivos articulados como fundamentales, se ha establecido un conjunto de *User Services* (Servicios al Usuario) que pretenden ofrecer las soluciones a esas problemáticas planteadas. Cada país ha adoptado sus propios servicios al usuario con su nomenclatura correspondiente, siendo en definitiva muy similares de un país a otro. Además, esos *User Services* se agrupan en categorías, que pueden ir desde 5 a 8 según el país considerado.
Actualmente, el grupo de trabajo ISO TC204 WG1 *(Architecture, Taxonomy* & *Terminology* de la *International Organization for Standardization*) [ISO03] trabaja en la norma *"Transport Information and Control Systems (TICS) – Reference Model. Architecture(s) for the TICS Sector – Part 1: TICS Fundamental Services"* [ISO04], lo que representa la primera clasificación normalizada de servicios ITS, que vendrá a sustituir la actual diversificación en cada país.

3.3 NORMALIZACIÓN ISO

ISO es una federación de alcance mundial integrada por cuerpos de estandarización nacionales. Es una organización no gubernamental establecida en 1947. La misión de la ISO es promover el desarrollo de la estandarización y las actividades en el mundo relacionadas con ella con vistas a facilitar el intercambio de servicios y bienes, y promover la cooperación en la esfera de lo intelectual, científico, tecnológico y económico. Todos los trabajos realizados por la ISO resultan en acuerdos los cuales son publicados como estándares internacionales.

El trabajo de preparar estándares internacionales se realiza normalmente a través de los comités técnicos de la ISO. Cada miembro interesado en un tema para el cual se ha establecido un comité técnico, tiene derecho a ser representado en ese comité. Las organizaciones internacionales, gubernamentales y no gubernamentales, en alianza con la ISO, también participan en las tareas. ISO colabora con la Comisión Electrotécnica Internacional (IEC) en todas las materias de la estandarización electrotécnica. Los estándares internacionales se elaboran de acuerdo con las reglas dadas en las directivas de ISO/IEC, parte 3.
Una norma internacional es el resultado de un acuerdo entre los organismos miembros de ISO. Para su elaboración se llevan a cabo los siguientes pasos [UPM00b]:

1. Trabajos preliminares, denominados PWI, (recopilación de documentación, discusión sobre el contenido etc.) previos a la toma en consideración a una nueva iniciativa.
2. Redacción de un anteproyecto (CD) que se difunde para un estudio en el seno del comité técnico correspondiente.
3. Remisión del documento a la secretaría general cuando el CD ha obtenido un apoyo sustancial.
4. Conversión del CD en un proyecto de norma internacional (DIS)
5. Votación de aceptación del DIS entre todos los organismos miembros de ISO; si logra un 75% de apoyo.
6. Se eleva al Consejo de ISO para su aceptación como norma internacional.

ISO TC204 WG1 es un grupo de trabajo cuyo objetivo principal es la de suministrar un modelo de Arquitectura de Referencia Conceptual que muestre la estructura e interrelaciones del sector y suministre definiciones a tiempo y apropiadas de la terminología mediante glosarios y diccionarios, que expliquen en un lenguaje claro y con un mínimo de jerga, los términos usados en TICS, y desarrollen normas para AVI (*Automatic Vehicle Identification*) genéricos en el sector TICS.
Como resultado de la combinación de las principales contribuciones efectuadas, tales como las definiciones de los servicios de usuario en USA, Japón, Taiwán y Corea, junto con la definición europea de las necesidades del usuario, el grupo de trabajo (WG) está definiendo actualmente un conjunto de Grupos de Servicio ITS y Categorías las cuales pueden ser usadas en una variedad de combinaciones y configuraciones, para proporcionar una descripción del contorno de las diferentes aproximaciones de arquitecturas ITS. A su vez este grupo de trabajo está promoviendo el uso de *XML Schema* en dichas arquitecturas.

El informe técnico elaborado por los miembros de ISO y el cual ha sido tomado como referencia en esta investigación, establece una definición de los servicios primarios y áreas de aplicación provistas a un usuario de ITS. Éstos son descritos como “Grupos de servicio ITS y Categorías”.
Este informe técnico se está diseñando para asistir a la integración de servicios en una arquitectura cohesiva de referencia, ayudar a la interoperabilidad y la definición común de los datos. Específicamente, los servicios definidos dentro de los grupos de servicio serán la base para la definición de los casos del uso y de la funcionalidad resultante de la arquitectura de referencia, junto con la definición de datos aplicables dentro de diccionarios de los datos, así como estándares aplicables de las comunicaciones y del intercambio de datos.

3.3.1 Categorización de Servicios. Definiciones

La norma ISO TC204 [ISO04] establece algunas definiciones como: "*Un servicio ITS consiste en un producto o una actividad informativa proporcionada a un tipo específico de usuario ITS. Un grupo de servicio ITS consiste en unos o más servicios similares o complementarios proporcionados a usuarios ITS. Un dominio soporte o categoría (Support Domain) del servicio se refiere a un área de aplicación específica que abarque a uno o más grupos de servicio".*
Los servicios ITS no representan tecnología o funcionalidad interna de un sistema ITS. El nivel del detalle en el informe está en el nivel los grupos de servicio y las categorías, y no trata los servicios específicos.

Los grupos de servicio ITS y los propios servicios allí contenidos poseen una serie de características:

a) Cada grupo de servicio ITS está orientado a un grupo de servicios relacionados para la gestión de, o a la disposición de la información sobre la red del transporte. Puede ser dividida en servicios específicos si es necesario dirigirse a usuarios particulares o modos.

b) El nombre de cada grupo de servicio debe reflejar su área de aplicación (p.e. información de previaje)
c) Cada servicio dentro del grupo de servicio debe referirse a la actividad del grupo de servicio y a la naturaleza de los usuarios o de los modos abarcados por el servicio (ejemplo, información de previaje - transporte público)
d) Cada grupo de servicio debe ser lógicamente constante y de un tamaño manejable, y no debe referirse a usuarios específicos o a la cuestión de los modos (los servicios dentro de los grupos, por otra parte, pueden referirse a usuarios específicos o a modos).
e) Cada nivel de la jerarquía debe estar en un nivel equivalente de granularidad.

Usuarios ITS

ISO TC204/WG1 define a un usuario ITS como sigue: "*aquél que interactúa con proveedores de servicios ITS para beneficiarse de un servicio ITS".*
Además distinguen dos clases claramente diferenciadas:

- Usuarios de ITS externos: Aquéllos que interactúan con el sistema ITS a través de interfaces externos ITS. Los interfaces y usuarios externos definen el límite del sector ITS. Pueden utilizar simultáneamente servicios de sistemas que no se consideren ITS.

- Usuarios Internos: Aquéllos que interactúan con el sistema ITS a través de las interfaces internas del sistema. Es una entidad de un sistema ITS que recibe, directa o indirectamente, o provee a, la transacción de otro servicio ITS.

Algunos ejemplos de posibles usuarios de ITS son: viajeros, conductores particulares, conductores de autobús, turistas, proveedores de servicio, peatones, ciclistas, motoristas, transportistas, agencias de policía, operadores comerciales, operadores de transporte público, inspectores de vehículos de mercancías, servicios de emergencia, operadores de flota, operadores de control de peaje etc.

Categorías y Grupos de Servicios:

La categorización de las actividades ITS es uno de los primeros pasos en la definición del universo de actividades soportada por la Arquitectura de Referencia. Sirve para delimitar los diferentes sectores de la industria ITS. En la tabla 3.1 puede verse un sumario de la clasificación de servicios ITS agrupados en once categorías[6].

Categoría del Servicio	Nº	Nombre del Servicio
1. Traveller Information The service groups within the 'Traveller Information' domain address the provision of both static and dynamic information about the transport network and services for users prior to and during the trip, and provide tools for transport professionals to collect, archive, and manage information for future transport planning activities.	1.1	Pre-trip Information
	1.2	On-trip Information
	1.3	Travel Services Information
	1.4	Route Guidance & Navigation-Pre Trip
	1.5	Route Guidance & Navigation-On Trip
	1.6	Trip Planning Support

[6] Esta norma se encuentra en sus inicios y aún no ha sido traducida oficialmente al español, por lo que su contenido se muestra en inglés.

2. Traffic Management and Operations Service Groups The service groups under the "Traffic Management and Operations" domain specifically address maintaining the movement of people, goods and vehicles throughout the transportation network, and include both automated monitoring and control activities as well as decision-making processes (both automated and manual) that address real-time incidents and other disturbances on the transportation network, as well as managing travel demand as needed to maintain overall mobility.	2.1	Traffic Control
	2.2	Transport Related Incident Management
	2.3	Demand Management
	2.4	Transport Infrastructure Maintenance Management
3. Vehicle Services Groups The service groups in the "Vehicle Services" domain focus on specific services that improve the operational safety of vehicles, and are contained within the vehicle itself.	3.1	Transport Related Vision Enhancement
	3.2	Automated Vehicle Operation
	3.3	Collision Avoidance
	3.4	Safety Readiness
	3.5	Pre-crash Restraint Deployment
4. Freight Transport Services Groups The service groups addressed in the "Freight Transport" domain specifically address activities that facilitate both commercial vehicle operations and multi-modal logistics, including inter-jurisdictional coordination.	4.1	Commercial Vehicle Pre-clearance
	4.2	Commercial Vehicle Administrative Processes
	4.3	Automated Roadside Safety Inspection
	4.4	Commercial Vehicle On-board Safety Monitoring
	4.5	Freight Transport Fleet Management
	4.6	Intermodal Information Management
	4.7	Management and Control of Intermodal Centres
	4.8	Management of Dangerous Freight
5. Public Transport Service Groups The service groups within this domain describe activities that result in more timely and efficient operation of public transport services and provision of operational information to the operator and passenger.	5.1	Public Transport Management
	5.2	Demand Responsive and Shared Public Transport
6. Emergency Service Groups The service groups in this domain describe activities that permit emergency services to be more quickly initiated and expedited throughout the transportation network	6.1	Transport Related Emergency Notification and Personal Security
	6.2	Emergency Vehicle Management
	6.3	Hazardous Materials & Incident Notification
7. Transport Related Electronic Payment Service Groups This domain addresses activities that permit revenues for transportation services and facilities to be collected through non-cash and non-stop payment.	7.1	Transport Related Electronic Financial Transactions
	7.2	Integration of Transport Related Electronic Payment Services
8. Transport Related Personal Safety The service groups in this domain describe activities that protect the personal safety of pedestrians and individuals using road transportation facilities	8.1	Public Travel Security
	8.2	Safety Enhancements for Vulnerable Road Users
	8.3	Safety Enhancements for Disabled Road Users
	8.4	Intelligent Junctions and Links
9. Weather and Environmental Conditions Monitoring Service Groups The service groups in this domain describe activities that monitor weather and environmental conditions that have an impact upon the transport network and its users.	9.1	Weather Monitoring
	9.2	Environmental Conditions Monitoring
10. Disaster Response Management The service groups in this domain describe ITS activities that manage resources from multiple jurisdictions in their response to natural disasters, civil disturbances, or terrorism.	10.1	Disaster Data Management
	10.2	Disaster Response Management
	10.3	Coordination with Emergency Agencies
11.National Security The service groups in this domain describe activities that directly protect or mitigate physical or operational harm to persons and facilities due to natural disasters, civil disturbances, or terror attacks.	11.1	Monitoring and Control of Suspicious Vehicles
	11.2	Utility or Pipeline Monitoring

Tabla 3.1: Sumario de jerarquía de servicios. Arquitectura de referencia ITS según ISO TC 204 [ISO04]

3.3.2 Uso de XML en los estándares ITS.

Para alcanzar los objetivos marcados por los ITS, se deben de cumplir ciertas condiciones relacionadas con el intercambio de de información entre organizaciones pertenecientes a distintos países, diferentes áreas funcionales de ITS, sistemas heterogéneos e incluso a través de varias redes de comunicaciones. Desde el punto de vista de la tecnología de información de los ITS, se requieren algunos elementos, de entre los que destacan los siguientes:

- Método formal para definir vocabularios ITS expandirlos y reutilizarlos;
- Registro, gestión y mantenimiento de reglas y herramientas para componentes XML (Registros y Diccionarios de Datos);
- Método formal para definir diálogos y mensajes;
- Generación automática de esquemas XML desde UML; y
- Reglas para la transformación automática entre ASN.1 a/desde esquemas XML.

Lo que quiere decir que ISO está promoviendo el uso de diferentes lenguajes como UML, ASN.1 y principalmente XML Schemas para describir las estructuras de información y mensajes. El grupo de trabajo TC204 WG1 sugiere finalmente el uso de la recomendación[7] W3C de XML Schema [XMLS04] como estándar internacional XML, para de esta forma conseguir interoperabilidad entre aplicaciones [PWI04]. Hay que hacer constar que el grupo de trabajo ISO TC204 WG1 sigue manteniendo este campo de trabajo como objeto de discusión en reuniones recientemente celebradas [ISO05]. Existen varios ejemplos del uso de XML en áreas de ITS, como describiremos en la siguiente sección, a través de los diferentes lenguajes de marcado concernientes a tráfico vial.

3.3.3 Estrategia para una estandarización de Servicios Web.

Actualmente TC204 WG1 está planteando la necesidad de hacer uso de aplicaciones basadas en SW en determinados campos como los sistemas de información al viajero o servicios de emergencia. Actualmente hay un trabajo preliminar (*PWI*) denominado "*Using Web Servicies (machine-machine delivery) for ITS service delivery*" [PWI05].

3.4 LENGUAJES DE MARCADO EN EL ÁMBITO DE TRÁFICO VIAL

Desde hace unos años se han venido planteando la necesidad de uso de lenguajes de marcado que permitieran un intercambio de información de manera fiable, haciendo hincapié en el uso del estándares como HTML primero y XML después, pero hoy en día sabemos que estas arquitecturas pueden resultar ineficaces si el objetivo va más allá de un simple intercambio de información. Sin embargo, hay que hacer constar que el uso de XML sigue siendo una de las herramientas más importantes en el área de distribución de información en Internet, ya que este se ha utilizado para definir casi

[7] Una Recomendación del Consorcio World Wide Web (W3C) se entiende por la industria y por la comunidad Web en general como un estándar Web. Cada Recomendación es una especificación estable desarrollada por un Grupo de Trabajo del W3C y revisada por los Miembros del W3C.

todos las nuevos lenguajes que se utilizan para intercambiar datos sobre la Web. Por ejemplo, los lenguajes de contenido semántico han sido serializados[8] en él.

Actualmente, existen vocabularios o lenguajes que describen conceptos y estructuras de datos relacionados con tráfico, pero son solo descripciones sintácticas, carentes de semántica. Por lo tanto, aunque existen numerosos trabajos y desarrollos que hacen uso de lenguajes de marcado XML (difusión de información, intercambio de información, modelado de tráfico), no se conocen especificaciones semánticas como las propuestas en este trabajo.

Por ejemplo, en cuanto a la difusión de información de tráfico por parte de un centro gestor, existen algunas iniciativas que hacen uso de lenguajes de marcado para la difusión de información como CARS (*Condition Acquisition and Reporting System*) [CARS03] en Estados Unidos, en la que se hace uso de XML y de diccionario de datos TMDD (*Traffic Management Data Dictionary*) y el proyecto del Departamento de Investigación y Desarrollo del Instituto Hokkaido Development Bureau dentro del programa *ITS/Win Research*, en el cual proponen un lenguaje que usa tecnología XML denominado RWML (*Road Web Markup Language*) para manejar la información de tráfico. Este lenguaje consiste en la descripción de un vocabulario que permite representar la información relacionada con las carreteras, información meteorológica, desastres naturales, regiones geográficas etc. [RWML03]

Experiencias en el desarrollo de lenguajes de marcado para el intercambio de datos de información de tráfico usando XML destacan: TDML (*Traffic Data Murkup Language*), donde se define y se hace uso de XML Schema Data en lugar de DTDs [TDML03], y en Europa TPEG (*Transport Protocol Experts Group*) con *Road Traffic Message Application* ML (Tpeg-rtml v0.4) [RTML03].

Otras iniciativas que han intentado englobar ambas áreas (difusión de tráfico e intercambio) han sido:

- TRIDENT (*TRansport Intermodality Data sharing and Exchange NeTworks*) [TRIDENT02], proyecto europeo (ERTICO) finalizado en julio de 2002. Enfocado en el intercambio y difusión de información multimodal de viaje (transportes públicos, privados, etc.). Como punto de partida, para la especificación tomaron DATEX[9], al ser uno de los sistemas que se estaban utilizando en ese momento. El lenguaje utilizado para el intercambio fue XML.

- OTAP (*Open Travel data Access Protocol)* [OTAP05]. Se basa en disponibilidad de la información proveniente de los CIT (Centros de Información de Tráfico) por medio de Internet, haciendo uso de estándares tecnológicos y formatos comunes. Cada centro es responsable de su área geográfica y de la calidad de servicio. Cualquier proveedor de servicios de información accederá a ésta para su posterior manejo. Para el intercambio de

[8] Serialización es entendido como el proceso de transformación a un formato serial como lo es XML.
[9] DATEX es un estándar para el intercambio de información de viaje y tráfico entre centros de gestión de tráfico.

información también hacen uso de XML y DATEX. En marzo de 2003 ERTICO empezó a participar en este proyecto.

- DTX-ML (DaTa eXchange Markup Language). Hace uso de igual forma de XML y DATEX. Es una iniciativa nacida en el seno del grupo de investigación LISITT (Instituto de Robótica), que está siendo utilizada en la actualidad en proyectos conjuntos con la DGT y RNE.

Por otra parte, en modelado de tráfico se han desarrollado varias especificaciones en el *Transportation Research Center* de la Universidad de Florida: TMML (*Traffic Model Markup Language*) es un lenguaje de marcas desarrollado para compartir los datos más fácilmente entre los diferentes productos de software para modelado de tráfico. Proponen una especificación que cubra todos los datos comúnmente usados por un conjunto de productos de software que manejen información relativa a intersecciones señalizadas y vías. El uso de TSDD (*Traffic Software Data Dictionary*) sirve como referencia para el vocabulario y etiquetas usados, identificando clases y atributos [TMML03].

Otros trabajos centran su interés en el desarrollo de vocabularios específicos para accidentes como CRML (*Crash Records Markup Language*), transporte y logística como TranXML [TranXML03] o información de viaje como el desarrollado por Mitretek donde la *Society of Automotive Engineers* (SAE) ha desarrollado un vocabulario XML para ATIS (*Advanced Traveler Information Systems*) denominado *Traveler Information Markup Language* (TIML) [TIML03], basado en el diccionario de datos *SAE ATIS* y los conjuntos de mensajes estándar (J2353 y J2354) definidos usando la notación ASN. 1. El resultado final es un lenguaje de marcas estándar documentado mediante XML Schema.

3.5 PROGRAMA TEMPO. PROGRAMA ITS, DOMINIOS DE APLICACIÓN.

Entre 1995 y 2000, la Comisión Europea promovió el desarrollo de ITS sobre la red TEN-T (***T**rans-**E**uropean **N**etwork for **T**ransport*) contribuyendo con 125 millones de Euros al financiamiento de los proyectos de la gestión de tráfico en carretera [Eur04]. Cinco de estos proyectos eran grandes proyectos de índole Euro-Regional los cuales implicaban varios Estados miembros. A su vez, también se desarrollaron diversos proyectos nacionales pero con importancia europea. Posteriormente, la Comisión Europea lanzó el programa TEMPO (**T**rans-**E**uropean intelligent transport syste**M**s **P**r**O**jects) en 2001. La obtención de beneficios de ITS a escala europea requiere un esfuerzo coordinado que involucra a todos los participantes. Antes de 2001, los proyectos individuales se ocuparon de varias implementaciones ITS, pero la coordinación entre los distintos participantes era realmente escasa. Ahora, sin embargo, con el programa TEMPO, se logra realizar un despliegue armonizado y sincronizado de los sistemas y servicios inteligentes de transporte en la red europea de transporte por carretera. El programa anual de financiamiento de TEN-T fue remplazado por el MIP (**M**ulti-Annual **I**ndicative **P**rogramme). Este programa multi-anual permite a la Comisión tomar decisiones indicativas sobre los presupuestos de ITS para los siguientes

años, permitiendo realizar una planificación a largo plazo y dar soporte a los proyectos a medio y largo plazo. Este programa está financiado en parte por la Dirección General para la Energía y el Transporte de la Comisión Europea y en parte por los Estados miembros (TEN-T Expert Group on ITS). Juntos han identificado un número limitado de objetivos claves así como las aplicaciones prioritarias del programa.

El programa se apoya en tres objetivos principales:

- Optimización del uso de la capacidad de la carretera y flujos de tráfico de pasajeros y mercancías.
- Ahorro en términos de seguridad en carretera, debido a la reducción de accidentes y su impacto.
- Alivio del daño ambiental con la reducción de la congestión del tráfico.

En orden a reflejar las acciones prioritarias definidas en el nivel europeo, todos los proyectos han sido organizados siguiendo siete dominios de aplicación más dos dominios adicionales (Project Management y Horizontal Issues), tal y como veremos al hablar de las actividades en los proyectos ARTS y SERTI.

Los proyectos financiados inicialmente dentro del programa TEMPO 2001-2006 son: ARTS (*Advanced Road Traffic in South-west*), CENTRICO (*Central European Region Transport Telematics Implementation Project*), CORVETTE (*Co-ordination and Validation of the Deployment of Advanced Transport Telematics in the Alpine Area*), SERTI (*Southern European Road Telematics Implementation*), VIKING y STREETWISE (*Seamless TRavel Environment for Efficient Transport in the Western ISles of Europe*). Cada uno de estos proyectos actúa en una región diferente de Europa.

A continuación se describen los dos proyectos en los que se encuentran involucradas las diferentes administraciones de tráfico de España, y sus dominios de aplicación.

3.6 PROYECTOS EURO-REGIONALES: ARTS Y SERTI

3.6.1 Antecedentes de los proyectos ARTS y SERTI

El proyecto ARTS [UV03] está constituido por cinco administraciones públicas (Dirección General de Tráfico, Dirección de Tráfico del País Vasco, Instituto das Estradas de Portugal, Direcçao Geral de Viaçao y la Direction de la Sécurité et de la Circulation Routiéres) junto con varias concesionarias de autopistas de peaje (Autoroutes du Sud de la France, Cofiroute, y French Toll motorway association ASFA) y varias compañías consultoras de España, Francia y Portugal. El área ARTS está enmarcada dentro de la zona TERN (*Trans-European Road Network*) del Sur-Oeste de Europa y principalmente del eje sur-Atlántico.

ARTS es principalmente un proyecto gestionado por autoridades públicas de tráfico, aunque en los últimos años se ha incrementado la participación de las compañías de peaje.

El proyecto SERTI [UV03b] está constituido por siete administraciones públicas (Ministerio Francés de Transportes, Dirección General de Tráfico, Servicio Catalán de Tráfico, Ministerio de Transportes del lander alemán de Baden-Wurttemberg,

Ministerio Italiano de Obras Públicas, Agencia de Movilidad de Andorra y la Administración Suiza de Tráfico) junto con varias concesionarias de autopistas de peaje francesas e italianas y varias compañías consultoras de España, Francia e Italia. El área SERTI está enmarcada dentro de la zona TERN del Sur-Este de Europa y principalmente del eje Mediterráneo y centro Europeo.

Las administraciones de tráfico y compañías constitutivas de los proyectos han estado trabajando juntas desde finales de 1997 en varios estudios preliminares sobre la factibilidad de mejorar la calidad y continuidad del servicio que están recibiendo los usuarios de carreteras dentro de la zona TERN mediante la ejecución coordinada de proyectos de instalación de sistemas telemáticos de tráfico, tanto a nivel regional, bilateral como multilateral.

3.6.2 Actividades de trabajo

Las actividades de trabajo de las propuestas para ambos proyectos se organizaron siguiendo los dominios de aplicación definidos por el grupo de expertos en Sistemas Inteligentes de Transporte para la gestión de tráfico por carretera, incluyéndose las recomendaciones propuestas dentro de los proyectos Euro-regionales MIP, sobre los que se han sustentado los trabajos efectuados [Eur04]:

- Dominio 1.- *Infraestructuras de Monitorización* (RMI).
- *Dominio 2.- Centros de Gestión de Tráfico (TIC).*
- *Dominio 3.- Control y Gestión de Tráfico (TMC).*
- *Dominio 4.- Servicios de Información al Viajero (TIS).*
- *Dominio 5.- Gestión de Flotas y Mercancías (FFM).*
- *Dominio 7.- Gestión de Incidentes y Emergencias (IEH).*
- *Dominio 8.- Temas Horizontales (HI.)*
- *Dominio 9.- Gestión del Proyecto (PM).*

A continuación se describe más detalladamente el cuarto dominio, debido a que la investigación llevada a cabo en esta tesis queda enmarcada en él:

El dominio 4 incluye todas las actividades relacionadas con la provisión de servicios de información al viajero, tanto “pre-viaje” (pre-trip) como “en-viaje” (on-trip). Los objetivos de las tareas incluidas en el mismo se agrupan en 4 subdominios:

· 4.3 Cálculo de tiempos de viaje. Este subdominio se centra en los estudios, experimentaciones e implementaciones de diferentes tecnologías para el cálculo de tiempos de viaje. Se valida la precisión de la información proporcionada y cómo esta información es percibida por los usuarios.

· 4.4 Información en carretera y puntos fijos de información. Se promociona la instalación de Paneles de Mensajes Variables (PMV) en carretera, con objeto de crear una red adecuada para la distribución de información a los usuarios. A su vez se analiza la simbología e interpretación de los mensajes utilizados, fomentándose el trabajo de harmonización a nivel europeo de los mismos.

· 4.5 Información en el vehículo y sistemas de navegación. Los trabajos se centran en la mejora y evaluación del sistema de difusión de información *Radio Data System-Traffic Message Channel* (RDS-TMC) y su integración dentro de los sistemas de navegación. Se fomentan las actuaciones en radio digital, tanto terrestre como vía satélite.

· 4.6 Estudios y servicios basados en Internet y telecomunicaciones. Se trata de ofrecer a los usuarios finales servicios de información de tráfico y viaje a través de Internet o telefonía móvil. Se mejoran los servicios ofrecidos incluyendo la difusión de la información traduciéndola a diversos lenguajes e incrementando la diversidad y calidad de la información distribuida.

3.7 DIFUSIÓN DE INFORMACIÓN DE TRÁFICO. EL USUARIO DE LA RED VIARIA.

Tomando como punto de referencia el dominio 4 o TIS, se dará una visión de las tecnologías que han ido apareciendo o se han utilizado en ese campo.
El concepto de TIS engloba aquellas aplicaciones de tipo telemático que proporcionan información del estado del tráfico u otras informaciones relacionadas con él.

El análisis de nuevas tecnologías para la difusión de información de tráfico se hace indispensable, debido a la necesidad de mantener a los conductores debidamente informados.

Los proyectos europeos que en la actualidad se desarrollan con el objetivo de proporcionar al conductor toda la información que necesite tienen por fin el suministrar en tiempo real referencias sobre las condiciones climáticas, opciones alternativas, las posibles incidencias que puedan presentarse, aparcamientos disponibles, localización de hoteles y restaurantes etc. Tres son las claves técnicas en las que se trabaja para la implantación de estos sistemas: información de tráfico de mayor calidad, mejoras en la recepción de información para usuarios de telefonía móvil y mejoras en servicios de guía dinámica de ruta [UPM00c].

Son diversos los medios utilizados para el establecimiento de una comunicación entre los conductores o usuarios de ésta y los distintos centros emisores. A veces esta comunicación es unilateral y en el mejor de los casos bidireccional, aportándose en este último caso un cierto grado de interactividad que permite que la comunicación fluya en ambos sentidos. Podemos a su vez hacer una distinción en cuanto al ámbito de difusión de la información, y atendiendo a este factor establecer que existen aplicaciones de difusión o "*broadcast*" y aplicaciones "*unicast*". Sin lugar a dudas cada uno de éstos establece una correspondencia unívoca con el medio y el uso al que va dirigido. Aplicaciones como los Paneles de Mensajes Variables (PMV) o la radio son claramente aplicaciones de difusión y aunque fueron y son realmente importantes, carecen del sentido de comunicación bidireccional y por tanto, no aportan la deseable característica de la personalización.

A continuación se presentarán los sistemas que si bien forman parte de los centros de gestión de tráfico, ya que son ellos los encargados de su puesta en funcionamiento, mantenimiento y su actualización diaria, son utilizados por los usuarios para conocer el estado del tráfico [Tom04].

Estos sistemas se pueden clasificar principalmente en dos tipos atendiendo al momento en que el usuario recibe la información: Sistemas *Pre-Trip,* donde el usuario consulta la información necesaria antes de haber iniciado el viaje y sistemas *On-Trip*, donde el viaje se ha iniciado y el usuario quiere conocer cual es la situación del tráfico (colas, accidentes, rutas alternativas, etc.). En la figura 3.1 se puede observar un esquema con los diferentes sistemas de difusión de información sobre tráfico vial.

Figura 3.1 Difusión de información de tráfico

Hasta hace muy poco tiempo la idea de encontrar información sobre tráfico estaba concebida para o limitada por la obtención de la información de tipo Pre-Trip (antes de efectuar el viaje); el receptor de ésta sólo era capaz de recibir información previa a su posible desplazamiento por la red de carreteras. Posteriormente, el avance en telemática permitió el desarrollo de diversas tecnologías como RDS-TMC, que veremos más adelante, que originaron un gran cambio en cuanto a la concepción de la difusión de información. El usuario y receptor, puede hoy en día recibir información On-Trip (en ruta). Hay que hacer constar que los servicios Pre-trip y On-trip pueden involucrar el mismo tipo de información para el usuario y en algunos casos llegar hasta él mediante el uso de las mismas tecnologías. Por esta razón, ambas tecnologías pertenecen a la misma categoría de “Servicios de Información al Viajero”. El advenimiento de los desarrollos en comunicaciones móviles está reduciendo la necesidad de de distinguir entre ambos tipos de información [TEN-T00]. A modo de ejemplo, destacaremos que aunque Internet o el teléfono representan servicios del primer tipo, sin embargo el abastecimiento en las autopistas de puntos de acceso o incluso el uso de nuevas tecnologías como el caso de alertas de voz permiten tratarlas en determinadas situaciones como servicios On-trip.

A continuación se presentan los dos sistemas Pre-Trip más importantes:

- *Radio-TV-teletexto:* estos medios de difusión de información, junto al VideoText ya obsoleto, fueron los primeros sistemas para difundir información de tráfico. No permiten interactividad, aunque sus contenidos lógicamente son variables. El receptor recibe mucha más información de la que en realidad necesita y en muchos casos no desea.

- *Internet:* La aparición de Internet abrió un nuevo campo de trabajo para la difusión de la información en general y del tráfico en particular. Un gran número de entidades privadas y administraciones públicas pusieron en marcha diversas iniciativas basadas en la creación de portales que suministrasen información sobre tráfico. Estos portales ofrecen mapas y callejeros, cálculo de itinerarios, situación del estado del tráfico en tiempo real (obras, incidentes, etc.), información del tiempo etc. Éstos siguen siendo todavía un importante medio para distribuir información de forma rápida y eficiente. La principal novedad que aportaron los portales, fue sin lugar a dudas la posibilidad de interactuar a través de su interfaz.

Una vez visto los sistemas Pre-Trip, nos vamos a centrar en los sistemas On-Trip en los que se puede apreciar más claramente la importancia de las nuevas tecnologías en este tipo de sistemas.

- *RDS-TMC:* El Canal de Mensajes de Tráfico (TMC) es una aplicación específica del *Sistema de Datos de Radio* (RDS) de FM usado para transmitir en tiempo real información de tráfico. Los mensajes de tráfico se reciben de forma silenciosa (no afectan al programa de radio que se esté escuchando), se decodifican por un equipo de radio con TMC o un navegador dinámico, y se muestran al usuario visualmente o de forma hablada en el lenguaje seleccionado por el usuario. Los mensajes TMC se pueden filtrar de modo que solo se muestren aquéllos relevantes para el viaje, además un sistema de navegación puede ofrecer guiado dinámico – alertando del problema en la ruta planeada y calculando una ruta alternativa para evitar el incidente.

- *DAB*: El sistema DAB (*Digital Audio Broadcasting*) nació en 1987 como un proyecto europeo denominado Eureka 147. El objetivo era especificar un sistema de radiodifusión digital válido para comunicaciones terrestres y por satélite. El DAB aporta una gran novedad técnica a la radiodifusión, que posiblemente sea más importante que la calidad digital de audio que proporciona y que la gran capacidad del canal. Esta novedad consiste en el uso de una técnica llamada OFDM (*Orthogonal Frequency Division Multiplex*) que permite el establecimiento de redes de frecuencia única y elimina prácticamente todo el problema de las interferencias que sufren las transmisiones convencionales. Entre las posibles aplicaciones de ITS de tráfico destacan el poder ofrecer información de tráfico y la transmisión de mapas digitales.

- *WAP:* La tecnología WAP (*Wireless Access Protocol*) o también conocida como "Internet aplicada a teléfonos móviles" se basa en la unión de Internet y la telefonía celular [Sam01]. Numerosas administraciones decidieron ofrecer el mismo servicio de información que el que se puede obtener mediante el servidor de Web, a través del teléfono móvil. Así, los usuarios disponen de la posibilidad de consultar antes y durante sus desplazamientos, una información fiable, rápida y actualizada que les conduzca a planificar su viaje por el recorrido más adecuado y en el tiempo estimado. Una de las ventajas aportadas por la telefonía inalámbrica fue la incorporación del concepto "en cualquier instante y en cualquier lugar". La telefonía móvil nos brinda una cierta autonomía en cuanto a la disposición de la información deseada en cualquier sitio. Esta circunstancia se convierte en esencial cuando pensamos en la movilidad de los conductores.

- Sistemas de *alertas de incidencias por voz, e-mail o Short Message Service (SMS)* [Sam03]. El acceso a la información por medio de la voz es el método más natural e inherente al ser humano, ya que la voz es la interfaz de comunicación entre las personas. Este es el tipo de acceso apropiado cuando se está conduciendo un vehículo. Un servicio de alertas de tráfico mediante voz incluye por tanto dos importantes premisas: el acceso a la información por cualquier persona, en cualquier sitio e instante y no interferir en la capacidad de conducción del usuario. El sistema, cuando aparece una incidencia nueva, realiza una búsqueda en la base de datos para obtener aquellos usuarios interesados en la misma. El usuario recibirá una llamada en su teléfono móvil en la que se le informará de la carretera, punto kilométrico y motivo de la incidencia. Existe también la posibilidad que el usuario reciba la información mediante mensajes SMS o por correo electrónico si lo prefiere. Ver figura 3.2.

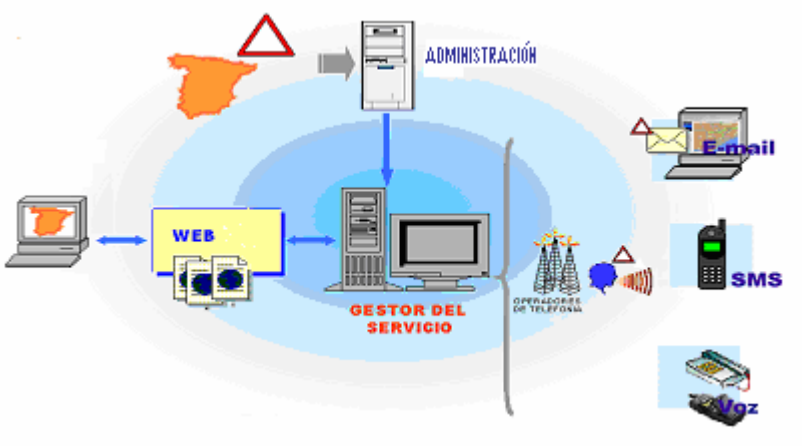


Figura 3.2 Arquitectura de un sistema de alertas

- *Portales de voz* [Sam02]*:* Los portales de voz son el resultado de la convergencia entre el mundo de la telefonía e Internet. La idea es sencilla: poder comunicar mediante voz los contenidos que actualmente son accesibles a través de Internet. Mediante esta idea, lo que se intenta es extender los conceptos de buscador, portal, navegador, etc. hasta ahora propiedad de la Internet tradicional, al mundo telefónico. La interacción mediante la voz con los sistemas informáticos permite conectar a los usuarios con bases de datos en cualquier lugar y en cualquier instante utilizando algo tan natural para el ser humano como el habla. Navegar por Internet debería ser algo tan sencillo como el mantener una conversación con otra persona, interactuando mediante órdenes vocales. Mediante un portal de voz, el conductor de un vehículo puede consultar en cualquier momento el estado de las carreteras, obtener información de las incidencias ocurridas y conocer el nivel de servicio de la vía. Esta consulta puede realizarla durante la conducción ya que únicamente interviene la voz en todo el proceso.

Otras tecnologías:

La combinación de un sistema de posicionamiento vía satélite para localización y seguimiento del vehículo y un sistema de telefonía móvil para realizar la transacción

ofrecen la mínima infraestructura de carretera necesaria. La implantación de ambos sistemas está muy desarrollada y alcanza casi cualquier punto de la geografía nacional e internacional (en algunos sistemas de posicionamiento, como el *Global Positioning System* o GPS, la cobertura es a nivel mundial). Estos sistemas fueron y son actualmente muy importantes en cuanto a la dotación de un dato relevante: la posición del individuo que requiere una información determinada. Este dato es en suma, muy interesante a la par que importante porque mediante él, cualquier sistema nos permitirá determinar, con mayor o menor exactitud, la posición del usuario y esto dotará a las aplicaciones de un mayor grado de interactividad y personalización en el servicio. Recordemos la máxima de "dar al usuario lo que necesita según sus requerimientos y gustos". A su vez nos permite ofrecer información local a la posición del usuario o móvil.

Por otra parte, la participación en proyectos europeos como ARTS y SERTI, mayoritariamente dentro del dominio denominado "Servicios de información al viajero", en el subdominio 4.6 "Servicios basados en Internet y Telecomunicaciones" me ha permitido realizar diversos aportes a este área en una fase previa, así como durante el desarrollo de mi tesis. Han sido efectuadas numerosas contribuciones en este campo, y se han diseñado, desarrollado y probado diversas arquitecturas que toman como base estas tecnologías. A continuación se nombran las principales actividades llevadas a cabo:

- **Implantación de un Servicio WAP de Información de tráfico para el Servei Català de Trànsit (SCT)**. [Sam01], [San02]: Este sistema se constituyó en el primer servicio de información de tráfico para dispositivos móviles tipo WAP desarrollado en catalán y fue uno de los primeros de Europa.

- **Actualización del servicio WAP para la Dirección General de Tráfico (DGT)**: En este proyecto se propuso una arquitectura para desarrollo de aplicaciones multidispositivo basada en XML.

- **Desarrollo de un portal multilingüe de información de incidencias para PDA´s para la DGT**: En este trabajo se desarrolló un marco para desarrollo de aplicaciones multilingües de información de tráfico que podía ser accedido a través de diversos dispositivos.

- **Desarrollo de una plataforma de voz para ofrecer información sobre incidencias de tráfico en ambas administraciones**. Estudio de factibilidad de portales de voz [Sam02] y posterior desarrollo de un servicio de alertas de incidencias de tráfico basado en mensajes de voz, e-mail y SMS para ambas administraciones [Sam03], [Sam03b]. Esta aportación se basó en la exploración de la utilización de interfaces de voz en acceso a información de tráfico vía Web.

- **Evaluación de los diferentes sistemas que ofrecen información sobre restricciones de tráfico a vehículos pesados.** En este estudio se trató de evaluar las distintas fuentes y medios por los que actualmente este tipo de información es canalizada y distribuida, analizando posteriormente la integración de todos aquellos conceptos, en un principio aislados, para abordar en conjunto toda aquella información relativa a las restricciones del tráfico pesado, mediante

portales WWW, y hacer llegar ésta a los interesados por distintos medios electrónicos.

- **Estudio de las diferentes tecnologías necesarias para el desarrollo de oficinas móviles mediante GSM/GPRS.** En este trabajo se hizo un análisis de las posibilidades y costos que ofrecía la migración de soluciones móviles basadas en GSM (Global System for Mobile), así como el estudio de las posibilidades de nuevos servicios que podían ser ofrecidos haciendo uso de la plataforma GPRS (General Packet Radio Service).

- **Estudio sobre los servicios de localización móvil en los ITS.** En este estudio se hizo un análisis de las posibilidades que se tenía de desarrollar contenidos que variaran de acuerdo con la ubicación geográfica del usuario.

3.8 CONCLUSIONES

En este capítulo se ha puesto de manifiesto la ayuda y la mejora que supone la aplicación de tecnologías emergentes en el área de transporte, reflejadas en factores tan importantes como la reducción del número de accidentes. Se ha podido constatar el hecho de que el sector de la telemática aplicado a los ITS, ha sido introducido de tal manera que apenas nos altera el ritmo de vida, pero que nos facilita enormemente las actividades cotidianas de transporte.

Las iniciativas llevadas a cabo por la Unión Europea como el programa TEMPO, han logrado conjuntar esfuerzos y planificar la puesta en marcha de los ITS en la red de transporte por carretera de todos los países miembros. Cada uno de los países y a través de las administraciones que los representan, participan en distintos proyectos euro-regionales que permiten la obtención de resultados compartidos por todos sus participantes.

A mi juicio, esta unión de esfuerzos para resolver problemas de índole común ha sido uno de los mayores logros alcanzados, en cuanto a que permiten simplificar el proceso de encontrar soluciones ante problemas que no afectan a un único país. Pensemos que la red de carreteras no termina en el límite de las fronteras sino que ésta se extiende más allá del territorio de una única nación.

En cuanto a la difusión de información, ante la necesidad de presentar ésta de forma clara y comprensible, haciendo que la respuesta del usuario se produzca de una manera intuitiva y natural y el hecho de querer obtener esta información en cualquier lugar e instante, ha dado lugar a numerosas alternativas, como las presentadas en este capítulo. Como ha quedado patente, la evolución de las tecnologías marca el desarrollo de los diferentes campos de trabajo y a veces es determinante en la aparición y desaparición de éstos.

De entre estas tecnologías, destaca Internet como herramienta de gran alcance usada para diseminar la información entre muchos individuos. Esta tecnología ha dado lugar a una comunicación "ubicua". Los usuarios, las corporaciones y las entidades del gobierno utilizan Internet extensivamente para compartir la información y realizar

diferentes operaciones. Las administraciones de tráfico han adoptado el uso del Internet y del WWW como parte de su programa operacional.

Desde el análisis del estado del arte, se ha podido determinar que existen algunas aproximaciones que hacen uso de vocabularios o lenguajes que describen conceptos y estructuras de datos relacionados con información de tráfico vial, pero son solo descripciones sintácticas, carentes de semántica.

Por otra parte, tampoco se encontraron SW de tráfico, solamente portales Web. Generalmente lo que ofrecen las administraciones de tráfico en sus portales Web no son *SW* en sí, sino *sitios Web* que en algunos casos poseen formularios para poder interactuar con una base de datos y que dependiendo de los valores de entrada devolverán una información u otra. Estos portales no están capacitados para recibir peticiones de invocación mediante protocolos de comunicación, por lo que deben de sufrir una adaptación y además necesitan del desarrollo de aplicaciones software que puedan extraer de ellos el conocimiento. La principal diferencia entre portal web y servicio Web es el objeto de la interacción, ya que mientras en los primeros ésta se produce con los datos en las páginas, en los segundos se produce a través de aplicaciones.

A modo de ejemplo, las administraciones de tráfico españolas están haciendo un gran esfuerzo para adaptar sus sistemas de distribución de información a estas nuevas tecnologías. La participación de la Universidad de Valencia a través de dichas administraciones en los proyectos relacionados con estas áreas, ha hecho posible mi contribución de forma activa en el desarrollo de arquitecturas orientadas a facilitar la distribución y posterior recepción de la información de tráfico vial mediante diferentes tipos de dispositivos. Este tipo de actividades me ha permitido ser consciente de la realidad de los servicios de información de tráfico actuales y la posibilidad de orientar este *background* de conocimiento hacia las nuevas corrientes de la WS y otras investigaciones asociadas. Ante la reticencia inicial de las administraciones por participar abiertamente en este tipo de investigaciones, mi grupo de investigación, liderado por el Dr. Juan José Martínez Durá solicitó en diciembre de 2003 una convocatoria de ayudas de Proyectos de Investigación Científica y Desarrollo Tecnológico al Ministerio de Ciencias y Tecnología, la cual fue aceptada (ref: TRA2004-06276). En este proyecto se está consiguiendo poner en práctica todo el conocimiento adquirido en las diversas actividades en las que se ha participado, así como en el desarrollo de esta tesis.

De igual manera, cabe considerar la estandarización de las diferentes arquitecturas relacionadas con esta área como un método previo necesario para hacer las cosas bien hechas, ya que el uso de estándares permitirá unificar criterios. La labor que está realizando el grupo de trabajo TC204 de ISO, sirve como marco de referencia para el desarrollo y despliegue de aplicaciones ITS basado en la definición de los requisitos y de los estándares apropiados y necesarios. Por otra parte, aunque la consideración de establecer como requerimiento el uso de XML *Schema* y de SW y la posterior definición de reglas para ser utilizadas por los desarrolladores de ITS, es un paso importante en la estandarización y como consecuencia en la interoperabilidad entre sistemas, todavía no se han planteado la introducción de la semántica tal y como se propone en esta tesis.

CAPÍTULO 4

RUTA CONCEPTUAL HACIA EL DESARROLLO DE SERVICIOS WEB SEMÁNTICOS

4.1 RESUMEN

En este capítulo se parte de la descripción de la propuesta del W3C denominada "Fundamentos de Ingeniería de la Web del Mañana", la cual sienta las bases para el entendimiento del papel de diferentes tecnologías relacionadas con la evolución de la Web, y dentro de la que un componente fundamental son los SWS.

Posteriormente se describe el estudio y estado del actual de dos componentes básicos de la Inteligencia Artificial que influyen notablemente en el desarrollo de la Web y los Servicios Web: la representación de conocimiento (y como parte de ella las ontologías y los mecanismos de inferencia), y los sistemas de agentes, como componentes software que permiten realizar inferencias sobre el conocimiento previamente representado.

Adicionalmente se presentan los conceptos básicos, lenguajes, herramientas existentes y estado actual de la Web Semántica y de los Servicios Web, cuya intersección apoyada por los sistemas multiagente constituye los SWS.

Finalmente, se presentan las principales tecnologías y arquitecturas de SWS. Primero se describe el concepto, así como las tareas que se espera sean desarrolladas por una ontología que permita agregar anotaciones semánticas a los SW. Posteriormente se aborda el estudio de las propuestas de arquitecturas independientes de lenguaje para su desarrollo, así como la descripción detallada de los componentes de las ontologías DAML-S y OWL-S. Por último, se presentan los conceptos de *matching* (emparejamiento) semántico, grados de similitud o *match* así como los diferentes sistemas desarrollados que hacen uso de este tipo de emparejamiento.

4.2 INTRODUCCIÓN

La *WWW* se ha convertido en un instrumento de uso cotidiano para el intercambio de información en nuestra sociedad. Desde que a principios de los años 90, Tim Berners-Lee presentase su proyecto de "World Wide Web" en el CERN (Suiza), ha llegado a superar a medios como la radio o televisión y se ha constituido junto al papel en uno de los medios de publicación más importante en la sociedad actual.

La arquitectura inicial de la Web estaba basada en tres pilares fundamentales: HTML, URIs y HTTP. El W3C ha presentado recientemente la arquitectura de la Web del mañana, sobre la sólida base proporcionada por 3 tecnologías: URI, HTTP 1.1 y XML. (Ver figura 4.1)

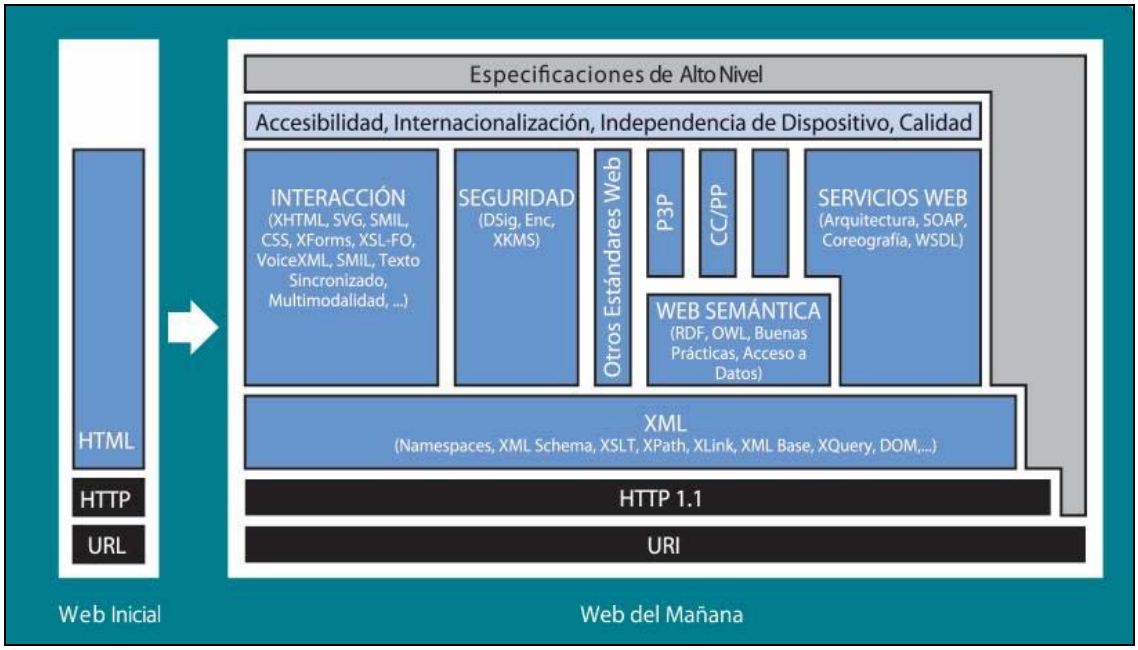


Figura 4.1 Web del mañana. [Mar05b]

En este proceso de evolución, una primera etapa se basó en el uso de tecnologías de simple publicación de contenidos (basada mayoritariamente en HTML, así como información textual con enlace a contenidos gráficos o multimedia), donde el centro de atención era la interacción con *datos* diseñada principalmente para consumo *humano*. Dentro de los aspectos fundamentales de la Web del mañana se encuentran la búsqueda de significado y enriquecimiento de los datos, así como la interacción entre *máquinas* donde un componente importante es la interacción entre *aplicaciones*.

A partir del análisis de la figura anterior podemos observar que por encima de la capa base XML encontramos una capa horizontal compuesta, entre otras, por las siguientes tecnologías base:

Interacción: Este componente está relacionado con las tecnologías que forman parte de la interfaz de usuario para la Web, cuyo lenguaje de marcado básico es XHTML. Adicionalmente forman parte de esta capa los lenguajes de la segunda generación en la Web (CSS, MathML, SMIL y SVG). Finalmente, en esta capa también se desarrollan los lenguajes que formarán parte de la próxima generación de las interfaces web (VoiceXML, Interacción Multimodal y XForms).

Seguridad: Basada en la necesidad de garantizar la integridad, la confidencialidad y la autenticidad de los datos que fluyen a través de la Web se ha convertido en un requisito esencial. Dos estándares incluidos dentro de este grupo son XML *Encryption* y XML *Signature*.

Web Semántica: Iniciativa nacida del mundo académico que pone énfasis en los datos y en el significado de éstos.

Servicios Web: Iniciativa nacida del mundo empresarial enfocada en la comunicación entre aplicaciones.

Finalmente, podemos observar que dentro del desarrollo de los SW existe la posibilidad de crear una intersección entre Web Semántica y los primeros, lo que da origen a los SWS.

A continuación se presenta el camino conceptual que lleva el desarrollo de la visión de la Web del futuro, entendida como una Web más inteligente, apta no solo para el consumo humano sino también para la interacción entre máquinas. La conceptualización parte de las bases computacionales de la representación de conocimiento y sus componentes, para posteriormente describir las tecnologías base que constituyen los SWS.

4.3 REPRESENTACIÓN DEL CONOCIMIENTO

La representación del conocimiento se enfoca en el diseño de formalismos epistemológica y computacionalmente apropiados para expresar el conocimiento en un área particular.

El conocimiento se ha representado en los sistemas de información utilizando diversos formalismos. En las bases de datos se han utilizado diagramas entidad-relación para definir los conceptos y sus relaciones en un determinado universo. En programación se han utilizado gramáticas y estructuras de datos como clases y objetos. En Ingeniería del Software se ha propuesto el uso de lenguajes de modelado como UML en donde también es posible definir clases y sus relaciones.

Algunas de las formas que se han utilizado para representar el conocimiento en Inteligencia Artificial son la lógica matemática y las estructuras de datos.

En la lógica matemática destacan los enfoques basados en lógica y los basados en reglas.

En los sistemas basados en lógica utilizan fórmulas lógicas para representar relaciones complejas. Dentro de ellos las lógicas de orden mayor tienen el poder expresivo más alto, sin embargo, este mecanismo tiene diversas desventajas dentro de las que se encuentran su ineficiencia para manejo de grandes combinaciones de conceptos. Se entiende por lógica de mayor orden un lenguaje en el que las variables pueden aparecer en cualquier parte donde los predicados y/o funciones lo hagan. Si no se requiere una semántica de orden mayor, ésta puede ser simplificada en lógica de primer orden (FOL). La Lógica de Predicados es un ejemplo donde la sintaxis y la semántica son amba*s* de primer orden. Finalmente, realizar razonamiento sobre lógica de FOL es computacionalmente intratable para grandes cantidades de datos.

Por otra parte los sistemas basados en reglas permiten definir el conocimiento en forma de cláusulas IF-THEN u otras condiciones de acción. Estas reglas son posteriormente utilizadas por motores de razonamiento para inferir conocimiento a partir de las reglas definidas inicialmente.

En relación con los sistemas de representación del conocimiento basados en estructuras de datos, se destacan los siguientes:

- **Redes semánticas** [Qui68] consistentes en un conjunto de nodos los cuales representan objetos, conceptos o situaciones y enlaces que representan las relaciones existentes entre los nodos.
- **Marcos** (*frames*) [Min74] los cuales representan conceptos denominados clases y relaciones llamados *slots.* Los esquemas de representación basados en marcos insisten en una organización jerárquica de éstos, mientras que las redes semánticas no requieren de tal organización.
- **Redes de herencia estructurales** [Bra78] desarrolladas para subsanar las ambigüedades de las dos anteriores y cuya puesta en práctica se realizó en el sistema KL-ONE [Bra85].
- **Sistemas terminológicos** o *Descripciones Lógicas,* las cuales son un tipo de lenguaje de representación basado en lógica que ha sido diseñado para *razonar* sobre redes semánticas y *frames.*
- **Grafos, Redes de Petri, Mapas Tópicos,** son estructuras de representación de más bajo nivel que constituyen la base formal de otros mecanismos recientes de representación del conocimiento.

Las dos formas generales de razonar sobre las redes semánticas y los marcos son el emparejamiento (por ejemplo, identificación de objetos que tienen propiedades comunes) y la herencia de propiedades, en la cual las propiedades son inferidas para una subclase.

John F. Sowa afirma que "*la representación de conocimiento es un área multidisciplinar que aplica teorías y técnicas de los campos de la Lógica, Ontologías y la Computación*". Para este autor, la lógica provee la estructura formal y las reglas de inferencia, y sin ella no existirán criterios para determinar si hay sentencias contradictorias o redundantes. Las ontologías definen los tipos de cosas que existen en el dominio de aplicación, permitiendo que los términos y símbolos estén bien definidos y no den lugar a confusión. Por último, los modelos de computación permitirán implementar las dos primeras disciplinas en programas de aplicación [Sow00].

4.3.1 Ontologías

4.3.1.1. De metadatos a ontologías

Los metadatos son datos acerca de datos, y denotan cualquier tipo de conocimiento que puede usarse para conseguir información sobre la estructura y el contenido de una colección de documentos. Por ejemplo, estableciendo una analogía con una videoteca, los datos serian las cintas de video o dvd´s, mientras que los metadatos serían la información contenida en las fichas, es decir, el título del libro, director, protagonistas

etc. Una de las facetas más importantes en el entorno de los metadatos es la interoperatividad entre diferentes entidades que no tienen porque compartir el mismo conocimiento y tecnología. “Los metadatos describen el contenido, calidad, condición, y otras características de los datos. Describen el quién, qué, cuándo, dónde, porqué, y el cómo sobre un conjunto de datos” [NOAA04].

Podemos resumir las aportaciones de los metadatos en dos aspectos:

1. Información descriptiva sobre un objeto o recurso,

DATOS	METADATOS
Javier Samper	Nombre
Polígono La Coma S/N	Dirección
Paterna	Ciudad
Valencia	Provincia

Tabla 4.1: Descripción de un recurso

2. Permiten el etiquetado o catalogado.

Pero los metadatos solo van a estructurar los contenidos, por lo que necesitaremos algo que nos permita estructurar la semántica de un recurso, ese algo se denomina ontología (ver figura 4.2). La principal motivación de las ontologías es la de que ellas van a permitir compartir y reutilizar bases de conocimiento de manera computacional.

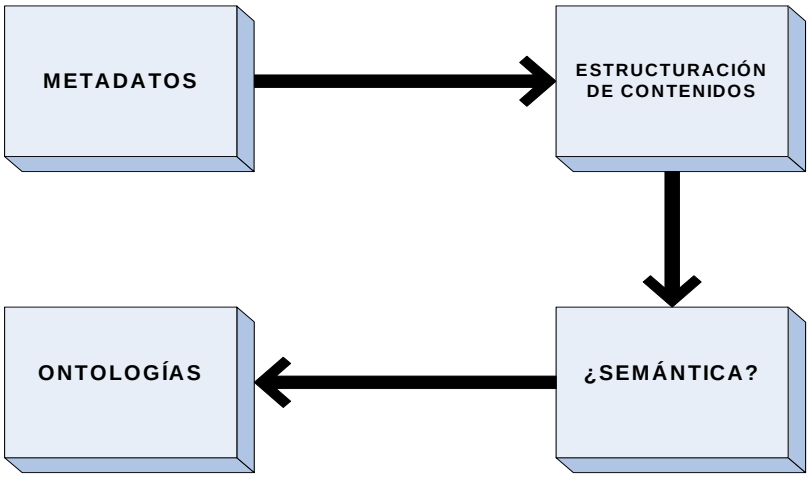


Figura 4.2 De metadatos a ontologías

El uso de ontologías proporciona una forma de representar y compartir conocimiento haciendo uso de un vocabulario común. Mediante esta representación permiten utilizar un formato de intercambio de este conocimiento, y a su vez brindan la posibilidad de ampliar, integrar otras ontologías o reutilizarlas en la aplicación de otros dominios. Separan el conocimiento del dominio del conocimiento operacional, haciendo independientes las técnicas y algoritmos, así como hacen que nuestras suposiciones sobre el dominio se hagan explícitas, lo que facilita replantearse éstas y ayuda a que

otros puedan entender su descripción. Por último, permiten analizar el conocimiento del dominio utilizando métodos formales. [Fri01], [Hon02]

Pero, ¿Qué queremos decir con semántica? Para responder a esta pregunta pensemos en lo que ocurre cuando leemos un texto y encontramos en ella símbolos. Estos símbolos serán interpretados (su significado) por nosotros con respecto a un modelo mental. Es decir, nosotros poseemos el significado (semántica) de (alguna parte de) del mundo en nuestras mentes. Cualquier otro individuo realizará su interpretación según su propio modelo mental que obviamente no tiene porque ser común al resto de individuos. Por tanto podemos asegurar que no hay un conocimiento en estos documentos si alguien o algo que interprete su semántica. Si nosotros deseamos diseminar conocimiento embebido en los documentos, necesitaremos, al menos parcialmente, automatizar el proceso de interpretación semántica. Necesitamos describir y representar computacionalmente una porción de nuestros modelos mentales sobre dominios específicos. Las ontologías son una herramienta fundamental para lograr este objetivo. Las ontologías intentan limitar las posibles interpretaciones a un solo modelo mental (el que nosotros queremos expresar). Ningún otro modelo como puedan ser las taxonomías, bases de datos etc. es capaz de hacerlo. Los computadores de esta forma suplantarán a los humanos en la interpretación de los diferentes vocabularios, mediante un proceso de inferencia que se asemejará toscamente al proceso de interpretación o razonamiento humano. [Dac03]

Un lenguaje de representación del conocimiento incluye una sintaxis del lenguaje (describe la configuración que puede constituir sentencias) y Semántica (determina los hechos y significado basado en las sentencias).

4.3.1.2. Definiciones

En filosofía, una ontología es una teoría que trata de la naturaleza y organización de la realidad, es decir de lo que "existe". Conocimiento del ser (del griego onto: ser y logos: conocimiento).

Platón trató con la cuestión de dar un apropiado nombre a las cosas, en su opinión esto era de suma importancia para que cualquiera pudiera unívocamente identificarlas. Aristóteles más allá de la cuestión de los nombres, se interesó por las definiciones. Una definición significaba explicar claramente lo que una cosa era mediante la existencia de una declaración esencial de la entidad. Por lo tanto, él creyó que para decir lo que algo era, siempre requería decir por qué era ese algo. Él ignoró las limitaciones inevitables de comunicar el significado vía un lenguaje y las ambigüedades creadas por el cambio implícito de los sentidos diferentes de significado. [Mae02]

Gottlob Frege introdujo una distinción entre dos tipos de significado: el concepto y el referente. La interpretación gráfica de esta distinción es comúnmente referida como el “triángulo del significado” (*meaning triangle*) y fue introducida por Ogden y Richards en 1923 [Ogd23]. El “*meaning triangle*” (figura 4.3) define la interacción entre símbolos o palabras, pensamientos y cosas del mundo real.

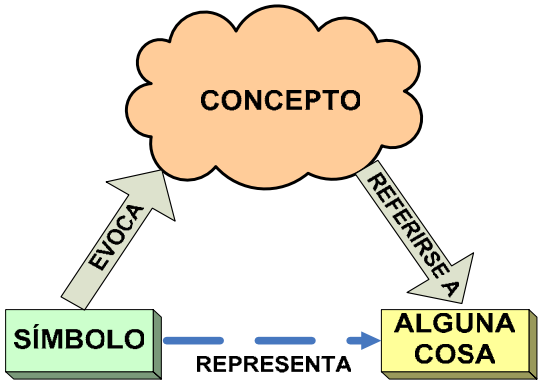


Figura 4.3 “Triángulo del Significado”. Desde metadatos a ontologías [Ogd23]

El diagrama ilustra que aunque los símbolos no pueden completamente capturar la esencia de una referencia (o concepto) o de un referente (o cosa), hay una correspondencia entre ellas. La relación entre una palabra y una cosa es indirecta. El enlace puede solamente ser completado, cuando un intérprete previamente procesa la palabra, la cual invoca un correspondiente concepto, para posteriormente enlazar este concepto a una cosa en el mundo. Hay una cierta correspondencia entre este triángulo de significado y el aspecto que cubren cada uno de sus elementos. Por ejemplo, utilizaremos los términos o símbolos “Javier” + “Samper” (sintaxis: símbolos) para dar un sentido o evocar al concepto <Javier Samper> y este concepto referenciará o denotará (mediante su semántica: significado) a un referente en el mundo real o posible (aspecto pragmático: uso). Mientras que la representación de los símbolos puede ser llevada a cabo mediante diferentes modelos, la construcción de las ontologías se convierte en esencial para establecer los otros dos vértices del triángulo y sus relaciones, gracias a su fuerte semántica lógico-conceptual.

Los investigadores de la Inteligencia artificial se han apropiado del término de “Ontología” y lo han incorporado a su jerga. En el campo de la Inteligencia Artificial "lo que existe es aquello que puede ser representado".
Una de las primeras definiciones en el área de la ciencia de la información la hizo Neches [Nec91] y su equipo de trabajo: “una ontología define los términos básicos y relaciones incluyendo el vocabulario de un área así como las reglas para la combinación de términos y relaciones para definir ampliaciones de un vocabulario”. Se puede decir que esta definición aporta unas líneas a seguir para crear una ontología: identificar términos básicos y relaciones entre los términos, identificar reglas para combinar las relaciones, aportar definiciones de los términos y las relaciones. Así que según esta definición, una ontología no incluye solo los términos que son explícitamente definidos en ella, sino que también los términos que pueden ser deducidos usando reglas.

En 1993, Gruber [Gru93a] dio una de las definiciones más empleadas: “las ontologías se definen como una especificación explícita de una conceptualización”. En 1997, Borst [Bor97] modificó ligeramente la definición de Gruber diciendo que, “las ontologías se definen como una especificación formal de una conceptualización compartida”. Estas dos definiciones fueron ampliamente explicadas por Studer y su equipo de trabajo en [Stu98]. Para Guarino [Gua98] “una ontología es una fuerte

estructura semántica que codifica reglas implícitas restringiendo la estructura de una porción de la realidad".

Otras definiciones surgen como consecuencia de cómo sus autores construyen y usan las ontologías: Bernaras [Ber96] y Swarout [Swa97].

Una ontología puede tomar una gran variedad de formas, pero necesariamente incluirá un *vocabulario de términos*, y alguna *especificación de su significado*. Esto incluye definiciones y una indicación de cómo los conceptos se interrelacionan lo que colectivamente impone una estructura sobre el dominio y restringe las posibles interpretaciones de los términos [Wei97] [Usc98], [Fri01].

Con las definiciones dadas, se puede ver que una ontología puede ser, una teoría lógica, una descripción formal de semántica, el vocabulario de una teoría lógica y una especificación de una conceptualización.

4.3.1.3 Componentes de las ontologías

Las ontologías tienen los siguientes componentes que servirán para representar el conocimiento de algún dominio [Gru93b]:

- **Conceptos**: son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc.
- **Relaciones**: representa la interacción y enlace entre los conceptos del dominio. Suelen formar la taxonomía del dominio.
- **Funciones**: son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología.
- **Instancias**: se utilizan para representar objetos determinados de un concepto.
- **Axiomas**: son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Permiten junto al mecanismo de la herencia de conceptos, inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos.

4.3.1.4 Tipos de ontologías

Van Heijst [Van96], propone una clasificación de las ontologías de acuerdo con la cantidad y tipo de la conceptualización. Así se pueden diferenciar los siguientes tipos de ontologías:

- **Terminológicas:** especifican los términos que son usados para representar conocimiento en el universo de discurso. Suelen ser usadas para unificar vocabulario en un dominio determinado.
- **De Información:** especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información.

- **De modelado del conocimiento:** Especifican conceptualizaciones del conocimiento. Contienen una rica estructura interna y suelen estar ajustadas al uso particular del conocimiento que describen.

Hay otras posibles clasificaciones de ontologías atendiendo a diversos criterios, por ejemplo si se atiende al asunto que conceptualizan, se distinguen tres tipos fundamentales de ontologías [Ste98]:

- **Ontologías de un dominio**, en las que se representa el conocimiento especializado pertinente de un dominio o subdominio, como la medicina, las aplicaciones militares, tráfico etc.
- **Ontologías genéricas**, en las que se representan conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos o los tipos de objetos, independientes de un dominio en particular.
- **Ontologías representacionales**, en las que se especifican las conceptualizaciones que subyacen a los formalismos de representación del conocimiento, por lo que también se denominan *meta-ontologías* (*meta-level* o *top-level ontologies*).

A estos tres tipos, Guarino [Gua98] añade las ontologías que han sido creadas para una actividad o tarea específica (denominadas ***task ontologies***), como por ejemplo la venta de productos o el diagnóstico de una enfermedad y las ontologías creadas para una aplicación específica. En dicho artículo, se pone de manifiesto la posibilidad de desarrollar diferentes tipos de ontologías teniendo en cuenta el nivel de generalidad. Ver figura 4.4.

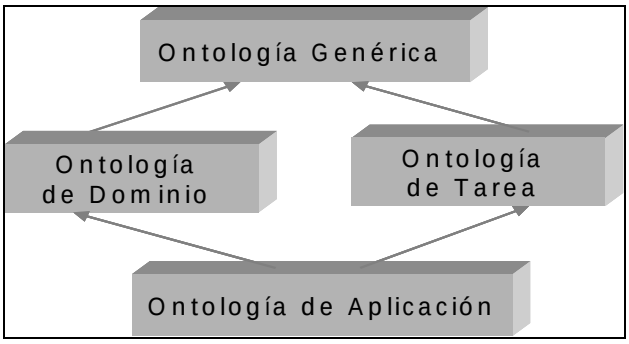


Figura 4.4 Tipos de ontologías, acorde a su nivel de dependencia de una tarea particular o punto de vista [Gua98]

Por último, se puede establecer una clasificación de las ontologías en función de su uso y reutilización (figura 4.5):

Figura 4.5 Clasificación de ontologías según uso y reutilización [Hon02]

Gruninger y Uschold [Grun02] argumentan que hay muchos tipos de cosas a las que la gente llama ontologías. Podemos pasar de tener solo términos aislados (con menor significado) a tener descripciones de términos dentro de una lógica formalizada (con mucha menor ambigüedad de los términos). Una ontología formal estaría formada por una lista de términos con definiciones, relaciones complejas entre éstos, reglas que controlen esas relaciones y valores potenciales para cada término.

4.3.1.5 Construcción de ontologías. Metodologías y métodos

En el diseño de una ontología se debe tener en cuenta que la ontología resultante cumpla ciertas características. Si cumple estas características tendremos la garantía de una ontología eficiente en las aplicaciones que la empleemos. Para especificar una ontología Gruber [Gru93a] propone cinco criterios que más tarde fueron ampliados a once por otros autores [Van96]. Los autores afirman que no es necesario el cumplimiento de todos ellos, pero que se debe realizar un balance de cuales son los convenientes para cada caso en particular.

Diversos autores han aportado guías de desarrollo de ontologías que establecen todos aquellos pasos que se consideran necesarios y los cuales se han convertido en un estándar metodológico [Fri01] y [Rec02].

El proceso de desarrollo de ontologías se refiere a las tareas que hay que llevar a cabo para construirlas. Adaptando el estándar establecido por la IEEE 1074-1995 [IEEE96] para el desarrollo de software general, las tareas identificadas para el desarrollo de ontologías se clasifican en 3 categorías:

a) Actividades ligadas al manejo del proyecto: Planificación, control del seguimiento de la planificación y actividades que aseguren la calidad del producto.

b) Actividades orientadas al desarrollo de la ontología: Especificación, conceptualización, formalización, implementación y mantenimiento.
c) Actividades integrales (necesarias para un buen desarrollo de la ontología): Adquisición de conocimiento, integración con otras ontologías, evaluación y documentación.

En general, las metodologías proporcionan un conjunto de directrices que indican cómo hay que llevar a cabo las actividades identificadas en el proceso de desarrollo, qué técnicas son las más apropiadas en cada actividad y qué produce cada una de ellas.

Hay dos métodos[10] principales que nos permite diferenciar dos tipos de ontologías según su construcción:

- ***Kactus***: es un método de construcción de ontologías que se basa en tomar una base de conocimiento y a partir de ésta, determinar y conceptualizar cuales son los términos y relaciones más importantes que representaran a la ontología. [Ber96].
- ***Sensus***: es un método que representa a las ontologías construidas a partir de una rama de una ontología más general y que es especializada para obtener una ontología nueva [Swa97]. Es decir consiste en crear ontologías específicas de dominio a partir una ontología más general.

Existen tipos de metodologías que construyen la ontología orientada al objetivo de la aplicación o proceso en el que se utilice. De este tipo de metodología destacaremos ***Methontology*** [Gom96b], recomendada por *FIPA* [FIPA04b], que lleva a cabo su cometido mediante las tareas de especificación, adquisición de conocimiento, conceptualización, integración, implementación, evaluación y documentación de las ontologías.

Otra metodología de este tipo (orientada a proceso) es ***On-To-Knowledge*** **(OTK)** [Sta01], incluye una identificación de metas, objetivos, que son logrados con herramientas de soporte para el manejo de conocimiento. Las distintas fases que promueven son estudio de factibilidad, fase inicial o *Kickoff*, refinamiento, inferencia, evaluación, y aplicación-evolución.

Otros métodos son ***Cyc KB*** [Len90] el cual consiste en la codificación de información extraída a mano y la adición posterior de más información usando herramientas de soporte y formalización. ***Uschold y King*** [Usc95] el cual se basa en identificar el propósito para posteriormente construir, evaluar y documentar las ontologías y por último, la metodología de ***Grüninger y Fox*** [Grun95] basada en la identificación de escenarios y la formulación de preguntas de competencia (*Competency Questions*)**,** extracción de conceptos y relaciones relevantes y la formalización en Lógica de Primer Orden.

[10] En la definición dada por IEEE, los métodos y técnicas son parte de las metodologías ya que son usados para llevar a cabo las tareas dentro de los diferentes procesos en los que consiste una metodología. [Gom04].

En [Fer99], [Cor02] y más tarde en [Gom04], han sido comparados los diferentes métodos y metodologías para el proceso de construcción de ontologías, atendiendo a diversos criterios en cuanto al soporte total o parcial de tipo tecnológico (herramientas usadas) y la estrategia seguida en la construcción como la propuesta de un ciclo de vida, grado de dependencia con la aplicación que la usa, estrategia para identificar conceptos (*bottom-up, top-down o middle-out*) o el uso de ontologías base como punto de partida en el desarrollo del dominio (Ver tabla 4.2).

Característica	CYC	Uschold & King	Grüninger & Fox	KACTUS	METHON-TOLOGY	SENSUS	OTK
Ciclo de vida propuesto	Prototipos de desarrollo	No propuesta	Prototipos de desarrollo o incremental?	Prototipos de desarrollo	Prototipos de desarrollo	No propuesto	Incremental y cíclica con Prototipos de desarrollo
Estrategia con respecto a la aplicación	Independiente de aplicación	Independiente de aplicación	Semi-dependiente de aplicación	Dependiente de aplicación	Independiente de aplicación	Semi-dependiente de aplicación	Dependiente de aplicación
Estrategia para identificar conceptos	No especificada	Middle-out	Middle-out	Top-down	Middle-out	No especificada	Top-down, bottom-up, middle-out depende de la aplicación
Uso de una ontología base	Si	No	No	No	No	Si	Depende de los recursos disponibles para el Proyecto
Herramientas que dan soporte	Cyc	Ninguna específica	Ninguna específica	Ninguna específica	ODE WebODE OntoEdit Protégé2000	Ninguna específica (Usualmente Ontosaurus)	OntoEdit con sus plugins

Tabla 4.2: Comparativa de las diferentes metodologías de construcción de ontologías [Gom04]

Aunque no existe una metodología estándar para la creación de ontologías, pueden distinguirse 4 pasos básicos en el proceso de creación [Cec02].

1. Identificación del propósito y del alcance: Se trata de especificar el contexto de aplicación y el modelado del punto de vista que queremos describir. El contexto de la aplicación describe el dominio, los objetos de interés y las tareas que van a realizarse. El modelado del punto de vista describe el tipo de modelo, p.e., dinámico-estático, funcional-causal, etc.
2. Construcción de la ontología: Podemos distinguir las siguientes etapas: Captura, Codificación (representación explícita de la conceptualización en un lenguaje formal) e Integración de las ontologías existentes (determinar si se va a reutilizar y cómo alguna de las ontologías existentes).
3. Evaluación: Evaluación del diseño definitivo. Se tendrán en cuenta aspectos como la posible reutilización de la ontología construida.

4. Documentación y reutilización. Como es habitual, la documentación debe desarrollarse paralela a la realización de las etapas anteriores. Debe incluirse la justificación de las decisiones tomadas, la evaluación realizada, el conocimiento adicional para usarla, etc. También ha de ser indexada y colocada con las ontologías existentes para su posible reutilización.

Como hemos visto, hay aproximaciones que no proponen ningún modelo de ciclo de vida (ver tabla 4.2). Podemos ver un esquema de ciclo de vida en la figura 4.6:

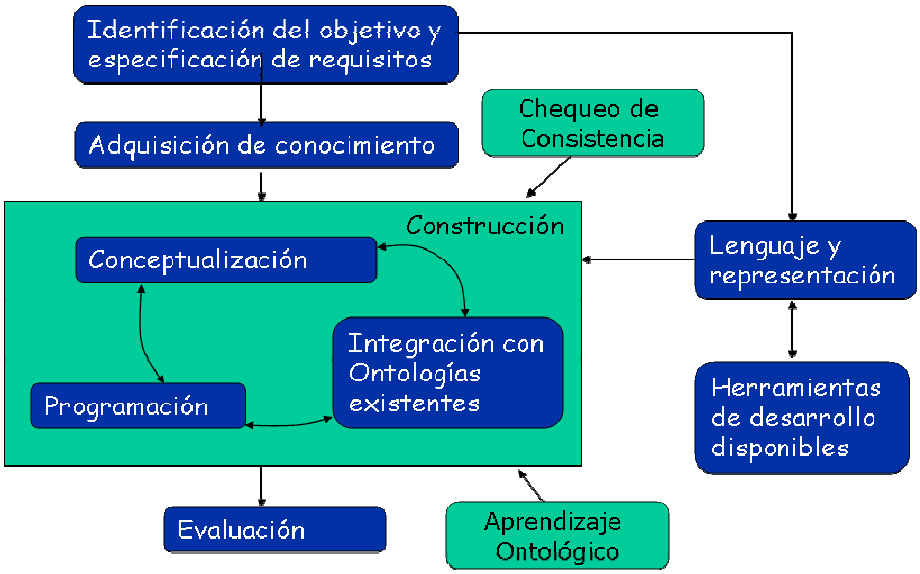


Figura 4.6 Ingeniería y Adquisición Ontológica [Gob02]

Los métodos y metodologías no han sido creados únicamente para construir ontologías desde cero. Cuando se (re)utiliza una ontología puede suceder que ésta esté implementada en un lenguaje con un paradigma de representación de conocimiento subyacente, diferente a las convenciones de representación usadas por la ontología que la reutiliza, que tenga diferentes enfoques etc. Para resolver algunos de estos problemas METHONTOLOGY incluye un método de **reingeniería** basado en las actividades de **reingeniería inversa,** consistente en obtener el modelo conceptual desde el código de implementación y posteriormente la actividad de **reestructuración** del modelo obtenido. [Gom04]

Sin embargo todos estos métodos y metodologías no permitían una construcción de la ontología de forma distribuida y colaborativa. Los métodos que incluyen la propuesta de este tipo de construcción son CO4 [Euz96] y $(KA)^2$.

Finalmente, todos estos métodos y metodologías descritos, fueron propuestos para la construcción de ontologías, pero hay muchos métodos y metodologías que no solo se encargan de esta parte del diseño sino abarcan temas como: la combinación o fusión de ontologías [Fri99], el aprendizaje [Mae02] y la evaluación [Gom01].

4.3.1.6 Lenguajes para el desarrollo de ontologías

Veamos ahora el universo de lenguajes de ontologías que se han empleado y las que a día de hoy se emplean.

A pesar de que es una tecnología reciente, son varios los lenguajes que se han utilizado para desarrollar ontologías y que con el paso del tiempo se han ido depurando para dar lenguajes más completos para su especificación.

Hay diferentes estudios que establecen comparativas entre los diferentes lenguajes, como los que aparecen en [Cor00], [Gil02], [Ont02b]. En esta sección se dan unas nociones básicas sobre estos lenguajes dando sus características principales y se analiza de forma más amplia las diferencias entre los principales más recientes.

Los lenguajes de especificación de ontologías los podemos dividir en dos grupos cronológicos: primero surgieron los lenguajes tradicionales de especificación de ontologías empleados en los sistemas de representación de conocimiento y después han surgido los lenguajes de especificación basados en Web.

En este tipo de lenguajes existen dos rasgos diferenciadores: la expresividad (¿qué puede ser dicho?) y la inferencia (¿qué puede ser obtenido de la información?).

4.3.1.6.1 Lenguajes de ontología tradicionales

Los lenguajes de ontologías tradicionales también se distinguen según la forma en la que se basan para representar el conocimiento: basado en *frames*, lógica descriptiva, predicados de primer y segundo orden, o los orientados a objeto.

A continuación se exponen los lenguajes más representativos [Gom04]:

Ontolingua [Ontol04] es un lenguaje de ontologías basado en *KIF* y en *Frame Ontology* (FO) empleado en el *Ontolingua Server*. *KIF* (*Knowledge Interchange Format*) desarrollado para resolver problemas de los lenguajes de representación del conocimiento, permite definir objetos, relaciones y funciones y utiliza predicados de lógica de primer orden. Como *KIF* es un lenguaje no orientado a construir ontologías, sino para intercambio de conocimiento, *Ontolingua* emplea FO para permitir la descripción de ontologías utilizando los paradigmas de *frames*, y con el que empezaron a surgir términos como clase, instancia, subclase. FO no permitía axiomas, pero al surgir a partir de *KIF*, si permite que se incluyan axiomas de *KIF* dentro de sus definiciones.

OKBC Protocol [OKBC95] (*Open Knowledge Base Connectivity Protocol*), como indica su nombre no es un lenguaje sino un protocolo basado en el *GFP* (*Generic Frame Protocol*) es decir, basado en *frames*. Es utilizado como complemento de lenguajes de representación de conocimiento, y permite, como otros sistemas basados en *frames*, clases, constantes, atributos, *frames* e instancias que sirven de base de conocimiento. También implementa una interfaz para acceder al conocimiento al igual

que funciones utilizadas para acceder a través de la red a un conocimiento compartido. Fue empleado junto a *Ontolingua*.

OCML [OCML99] (*Operational Conceptual Modeling Language*) formó parte del proyecto *VITAL* [Sha93]. Es un lenguaje basado también en *frames* que en su sintaxis permite declarar relaciones, funciones, reglas, clases e instancias. A este lenguaje se le añadió un módulo de mecanismos de lógica y una interfaz para poder interactuar con el conocimiento. Una de las ventajas que tiene este lenguaje es que es compatible con estándares como Ontolingua.

FLogic (Frame Logic) [Flog04] es otro lenguaje basado en *frames* pero que también hace uso de lógica de primer orden. Tiene una estructura con aspectos parecidos a los lenguajes orientados a objeto como la herencia, objetos complejos, tipos polimórficos etc. Se considera la misma relación (paradigma) entre *FLogic* y la programación orientado a objeto como la que hay entre el cálculo de predicados y la programación relacional. Con *FLogic* se han creado desde bases de datos a ontologías y se puede combinar con otros programas de lógica para interactuar con la información almacenada en la ontología.

LOOM [LOOM99] es un lenguaje de programación orientado a construir sistemas expertos y aplicaciones que hagan uso de la inteligencia artificial. Basado en lógica descriptiva (DL), está formado por la mezcla de los paradigmas basados en *frames* y los basados en reglas. Permite la descripción de objetos y relaciones, además, acepta que se le hagan restricciones a los objetos y relaciones, expresándolo en forma de afirmaciones, al igual que también permite realizar estas últimas sobre las instancias. También tiene un clasificador, por lo que permite inferencia y razonamiento. Este lenguaje destaca sobre los anteriores por estar basado en lógica descriptiva, no en *frames*, lo que permite realizar descripciones sobre objetos que podemos emplear en el clasificador que incorpora, ya que el clasificador se encarga de colocar los conceptos nuevos dentro de la ontología en el lugar adecuado, atendiendo a sus definiciones.

4.3.1.6.2 Lenguajes de ontología basados en Web y estándares

Estos lenguajes serán presentados dentro del apartado Web Semántica.

4.3.2 Inferencia y mecanismos de razonamiento

En conjunción con sistemas de inferencia, el razonamiento mediante lenguajes formales, como DL, sobre el conocimiento representado dentro de una ontología, permite que las máquinas compartan un marco común de referencia. Esta habilidad permite a las máquinas coleccionar información sobre el dominio de o presentarlo a humanos o a otras máquinas.

El marco de investigación de la DL también es conocido con el nombre de Sistemas Terminológicos haciendo hincapié en el hecho de que las clases y las relaciones definen los términos utilizados para representar un determinado dominio de conocimiento.

4.3.2.1 Lógica Descriptiva

Lógica Descriptiva es un lenguaje de representación de conocimiento adaptado para expresar conocimiento sobre conceptos y jerarquías entre éstos, mediante la creación de ontologías formales. Este tipo de lenguajes es apropiado para estructurar la información. Es un formalismo lógico cuyo origen son los fundamentos de otros formalismos como redes semánticas o frames. Sus primeros lenguajes de implementación y sistemas fueron: KL-ONE, Krypton, Classic, LOOM y Kris.

Hay una gran variedad de DL, siendo especificada cualquier lógica por tres conjuntos de características:

- Operadores para formar conceptos, tales como conceptos primitivos, conjunción y cuantificadores existenciales.
- Operadores de formación de propiedades, tales como primitivas e inversas.
- Axiomas soportados, tales como implicación de conceptos igualdad o implicación en las relaciones.

4.3.2.1.1 Servicios de inferencia en Lógica Descriptiva

La arquitectura de un Sistema Terminológico está formado por tres subsistemas (figura 4.7): la base de conocimiento, un mecanismo o sistema de inferencia y un interfaz. En la base de conocimiento se almacena información sobre el dominio organizada utilizando una jerarquía de clases y relaciones entre ellas. En ella se distinguen claramente dos partes bien diferenciadas: TBox y ABox. La primera de ellas contiene el conocimiento normativo (terminológico) en forma de una terminología y es construido a través de declaraciones que describen propiedades generales de conceptos (definiciones intensionales de conceptos y roles). El ABox contiene conocimiento extensional o factual (aserciones sobre objetos), específico para los individuales del dominio de discurso [Baa03]. En otras palabras, los TBox contienen las definiciones de los conceptos y roles, mientras que los Abox contienen las definiciones de los individuales (instancias)
El sistema de inferencia se utiliza para poder razonar sobre la base de conocimiento. El conjunto de razonamientos que se pueden realizar está directamente relacionado con la expresividad de la DL que se esté utilizando y finalmente el interfaz que permite la interoperabilidad con el Sistema Terminológico.

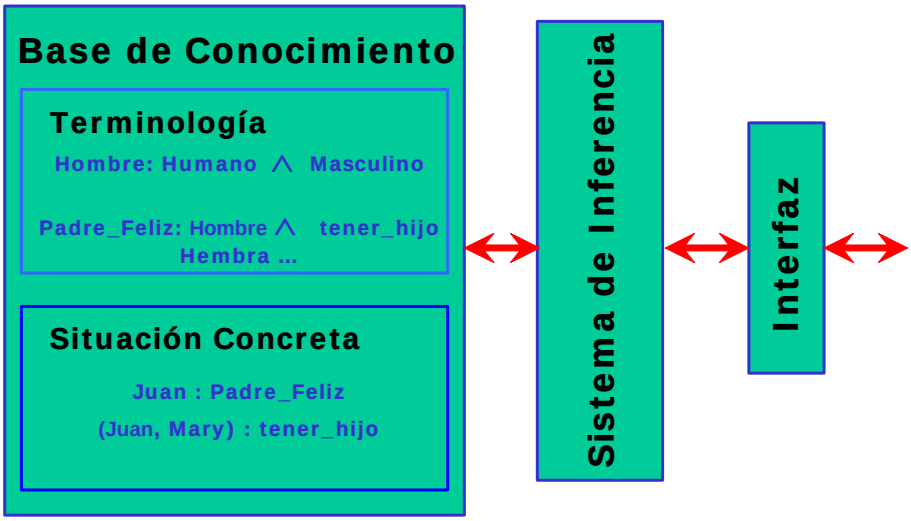


Figura 4.7 Arquitectura de un Sistema Terminológico [Hor01].

Un Sistema Terminológico permite razonar sobre el dominio de conocimiento. Los principales problemas de inferencia que se plantean resolver utilizando un Sistema de Razonamiento suelen ser similares en todos los dominios de aplicación e independientes de la DL utilizada.
La representación de conocimiento y el sistema de razonamiento basado en DL provee un número determinado de facilidades de razonamiento que puede ser separado en dos grupos: razonamiento terminológico (sobre el TBox) y razonamiento sobre las instancias (ABox).

4.3.2.1.1.1 Razonamiento Terminológico.

Las cuatro tareas siguientes son la base del razonamiento terminológico, implementadas en la mayoría de los sistemas:

- **Subsumción:** Consiste en comprobar la posición que ocupa una clase dentro de la jerarquía de la base de conocimiento. Comprobar si una clase es subclase de otra;
- **Consistencia**: Consiste en chequear si el modelo de conocimiento que se ha construido es consistente, es decir, si la estructura jerárquica generada es consistente con el conocimiento almacenado. Comprobar si existen definiciones de clases inconsistentes;
- **Construcción taxonómica:** Procesar todas las relaciones de subclase (incluyendo aquéllas que no son explícitamente especificadas pero que son implicadas por las definiciones dadas);
- **Clasificación**: Determina las clases que inmediatamente son más generales que otras o aquéllas que son incluidas (menos generales) que una clase dada.

4.3.2.1.1.2 Razonamiento Instancial.

Las tareas básicas que involucran los individuales o instancias son:

- **Realización**: Dada una descripción parcial de un individual, encontrar el concepto más específico que lo describe;
- **Comprobación de instancias**: Dada una descripción parcial de un individual, y una descripción de una clase averiguar si la clase describe la instancia. Consiste en chequear si un individuo es instancia de una determinada clase cumpliendo las características de ésta;
- **Recuperación de instancias**: Encontrar todos los individuales que son descritos por un concepto dado. Consiste en obtener todas las instancias de una determinada clase.
- **Equivalencia**: Hace referencia a comprobar si el conjunto de instancias de una determinada clase es el mismo al de otra.

4.3.2.1.2 DL y FOL

DL es una cuidadosa selección de parte de la lógica de predicados de primer orden (FOL). Aunque se trata de un subconjunto de FOL, sin embargo entre ellos hay algunas diferencias:

- En lugar de predicados unarios como en FOL, en DL conceptos (o clases).
- FOL orientada a razonamientos y a verdad o falsedad, mientras que en DL orientadas a conceptos y a pertenencia de objetos.
- En lugar de predicados binarios como en FOL, en DL roles (o propiedades).
- En DL existen constructores de conceptos para formar descripciones.
- Los axiomas de Subclass/property de DL se corresponden con implicaciones en FOL.

La razón por la cual FOL no es utilizada directamente para representar conocimiento sin restricciones adicionales es debida principalmente a la no existencia de un equilibrio entre la expresividad (constructores) y la complejidad del razonamiento. El poder expresivo es demasiado alto, lo que hace imposible obtener problemas de inferencia eficientes y decidibles. Por el contrario, DL es decidible con la asunción de mundo abierto. Por lo tanto, para cualquier base de conocimiento y cualquier consulta, hay siempre una forma de responder ésta en un número finito de pasos.

4.3.3 Bases de Conocimiento DL vs. Bases de Datos Relacionales

Las bases de conocimiento (KB) son la evolución lógica de los sistemas de bases de datos tradicionales (BD), en un intento de plasmar no ya cantidades ingentes de datos, sino elementos de conocimiento (normalmente en forma de hechos y reglas) así como la manera en que éste ha de ser utilizado. También se les trata de dotar de conocimiento sobre sí mismas, es decir, una KB ha de "saber lo que sabe" [Mor00].

Una BD consiste en dos partes: el esquema (estructura de los datos) y los datos que van a ser almacenados. En una KB, la parte en la que aparecen los términos y relaciones (TBox) puede ser comparada al esquema de las BDs y la parte instancial (ABox) a los datos. Sin embargo la semántica de los ABoxes difiere de la interpretación semántica de la instancia de los datos de las BDs. La ausencia de información en las BD es interpretado como información negativa, mientras que la ausencia de información en ABox únicamente indica una falta de conocimiento. Por ejemplo, si la única aserción sobre JAVIER es tenerHijo(JAVIER, CARLOS), entonces en el lenguaje de las BDs, esto es interpretado como el hecho de que JAVIER tiene exactamente un hijo, llamado CARLOS. Por el contrario en ABox, solo se puede decir que CARLOS es un hijo de JAVIER, pero no podemos afirmar nada sobre el hecho de que JAVIER tenga más hijos. Las BDs consideran su conocimiento como "**mundo cerrado**", el cual asume que un hecho es falso a no ser que haya sido explícitamente declarado como cierto, al contrario que las DL que lo consideran "**mundo abierto**". Esto da lugar a que las consultas en DL requieran un razonamiento no trivial, y no sea suficiente por ejemplo, con chequear instancia por instancia como en el caso de las BDs.

Hay un número de formas en las cuales las DL y las BDs pueden interactuar [NI03]:

- Durante el proceso de diseño de las BD, en el cual los sistemas de inferencia pueden ayudar a detectar inconsistencias y relaciones extra en los datos que están siendo modelados.

- Las BD relacionales pueden se usadas para almacenar las instancias de los datos para el esquema aportado por el sistema de DL.

- El sistema de inferencia puede ser usado para almacenar parte o todas las instancias de datos para una aplicación en particular.

La principal diferencia entre ambos sistemas, se centra en que mientras que las bases de datos manipulan grandes y persistentes modelos de relativamente datos simples, las bases de conocimiento proveen un mayor soporte para la inferencia (encontrando respuestas sobre el modelo las cuales no habían sido dadas explícitamente) e involucrando menor número de datos pero con una mayor complejidad [Borg03].

4.3.3.1 Beneficios de una semántica formal.

En [Kle00], los autores encuentran que la relación existente entre una ontología y un XML Schema es equivalente a la existente entre el modelo Entidad-Relación (ER) y el esquema relacional de una base de datos.
El modelo relacional provee una descripción de las bases de datos orientada a implementación, en tanto el modelo ER provee un marco para modelar orígenes de información requeridos para una aplicación. Podemos considerar que para el dominio que nos ocupa las descripciones dadas en un modelado de ER podrán ser representadas de igual forma y sin pérdida de significado, mediante un lenguaje ontológico basado en DL, ya que este puede capturar la semántica de muchas metodologías de modelado conceptual (DLR n-ary). [Hor01]

DL puede aportarnos algunos beneficios importantes entre los que destaca el uso de servicios de razonamiento DL que soporten el desarrollo y mantenimiento de modelos correctos. Además, dado que las DL son a menudo más expresivas, es posible sugerir extensiones a los modelos de bases de datos que permitan mayor información sobre la estructura de los datos que han de ser capturados.

Los modelos semánticos tradicionales como ER no proveen ciertas características que podrían ser útiles con el fin de representar dependencias complejas entre los datos [Baa03]:

- Constructores arbitrarios Boléanos. En ER solo son permitidas relaciones de tipos IS-A, mientras que en DL son permitidas relaciones como las expresadas mediante constructores que permiten por ejemplo establecer que una entidad es la unión disjunta de otras entidades o que la participación en ciertas relaciones está restringida etc.

- Refinamiento de propiedades de entidades, más que la mera adición de atributos, como por ejemplo las restricciones de cardinalidad pueden ser refinadas mediante la restricción en el rango de valores.

- Definiciones de clases por medio de propiedades complejas, permitiendo establecer condiciones necesarias y suficientes, en contraste con las condiciones únicamente necesarias que las instancias de las entidades deben de satisfacer en los modelos semánticos de datos.

- Tal y como afirman Calvanese et al. [Cal98] estableciendo una formalización de los esquemas ER en términos de DL, permitirá varias formas de razonamiento sobre éstos: satisfactibilidad en las entidades y relaciones, consistencia del esquema, detectar posibles redundancias, establecimiento de restricciones más fuertes mediante el uso de cardinalidad y relaciones de subsumción tanto en las entidades como las relaciones.

4.4 SISTEMAS MULTIAGENTE (SMA)

4.4.1 Conceptos Básicos

FIPA (*Foundation for Intelligent Physical Agents*) define un agente como "Un proceso computacional que implementa una funcionalidad comunicativa autónoma en una aplicación" [FIPA02]. Sin embargo, no existe consenso en la definición exacta de agente.

Organizaciones como ARPA, y su proyecto KSE (*Knowledge Sharing Effort*), OMG (*Object Management Group*) con su proyecto MASIF (*Mobile Agent System Interoperabilities Facility*) y Agent Society han definido estándares para la construcción de agentes (arquitecturas, protocolos de comunicación, aplicaciones etc.), pero es FIPA la que tiene más aceptación en estos momentos.

Varios autores han realizado clasificaciones de los distintos tipos de agentes [Bru91], [Fran96], [Cag97], [Hua00]. Una de las más aceptadas ha sido propuesta por Wooldridge [Woo97] que los clasifica en: de Información (Internet), de Interfaz, de Colaboración, Móviles, Reactivos, Híbridos y Heterogéneos.

Wooldridge [Woo99] define un agente de software como: "*Un sistema computacional capaz de realizar acciones independientes en beneficio de su usuario o dueño.*"

Los ***agentes de información*** son agentes software que tienen acceso a múltiples fuentes de información heterogéneas geográficamente distribuidas.
Algunas de sus características más importantes son:

- Intentan resolver los problemas asociados al manejo de información en Internet.
- Pueden asistir al usuario en la búsqueda y filtrado de información relevante, informar de la disposición de nuevos datos de interés etc.
- Ayudan al usuario en la ejecución de tareas, generalmente mediante el uso de las preferencias de este.
- Además, deberían ser capaces de actuar adecuadamente ante situaciones no previstas, es decir deberían tener capacidad de aprendizaje.

Los agentes pueden llevar a cabo sus tareas de manera independiente (agentes no cooperativos o individuales) o trabajar de manera coordinada con otros agentes (agentes cooperativos o sistemas multiagente).

El término ***sistema multiagente*** es usado para definir todos los tipos de sistemas compuestos por múltiples componentes autónomos que poseen las siguientes características [Jen98]:

- Cada agente tiene capacidad para solucionar parcialmente el problema.
- No hay un sistema global de control.
- Los datos no están centralizados.
- La computación es asíncrona.

Podemos entender un sistema multiagente como aquél que opera gracias a la interacción entre agentes de software, ya sea cooperando o compitiendo a fin de cumplir con los objetivos (posiblemente contradictorios) de sus usuarios. Los agentes del sistema interactúan entre sí mediante el envío de mensajes. De esta manera pueden coordinarse, cooperar o negociar entre sí, según los requerimientos del sistema en cuestión [Woo99].

4.4.2 Componentes de un sistema multiagente

El modelo de referencia de administración de agentes [FIPA04] establece los elementos básicos que deberían formar parte de un sistema multiagente. Según este modelo los agentes serán procesos computacionales que se comunican mediante un lenguaje de comunicación de agentes (ACL).

Las entidades contenidas en este modelo pueden observarse en la figura 4.8. A continuación se dará una breve descripción de cada uno de ellos.

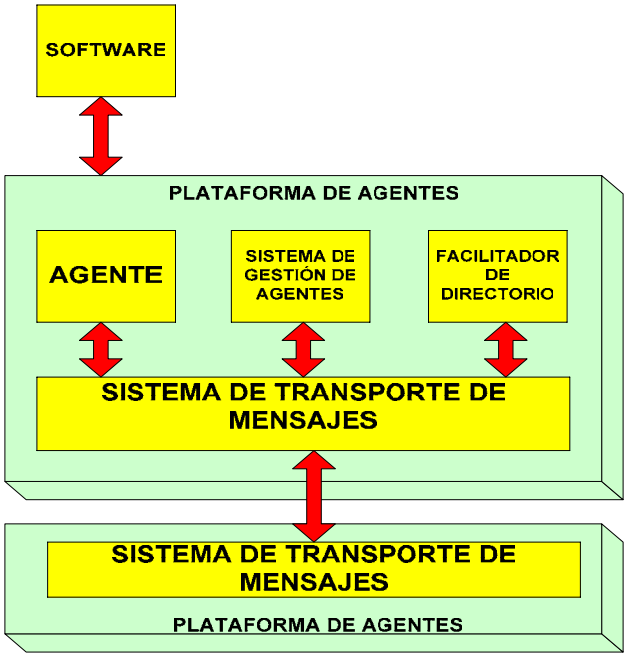


Figura 4.8 Modelo de referencia de administración de agentes [FIPA04]

Agente

Será el principal actor en la plataforma y combinará una o más capacidades de servicios, publicados en un descriptor de servicios. Deberá tener un dueño, y deberá implementar por lo menos alguna noción de identidad, denominada *identificador del agente* (AID), que etiquetará el agente para que pueda ser distinguido dentro de su universo.

Facilitador de directorio (DF)

Según [FIPA04] aunque este es un componente opcional, su existencia será muy útil ya que cuando está presente puede proporcionar servicio de páginas amarillas a otros agentes. Los agentes podrán registrar en él los servicios ofrecidos, preguntarle por el servicio ofrecido por un agente o qué agente ofrece un servicio determinado, cancelar un registro o modificar los datos de este.

Sistema de gestión de agentes (AMS)

Es un componente obligatorio que supervisará el acceso y uso de la Plataforma de Agentes (AP). Solo existirá un AMS en la AP, y se encargará de mantener un directorio de AIDs que contengan, entre otras cosas, direcciones de transporte de los agentes registrados en él. Cada agente deberá por tanto registrarse en el AMS para obtener un AID. Desarrollará tareas como la creación de agentes, su destrucción y su migración a/o de la plataforma. Mantendrá un índice de todos los agentes que residan en la AP. Deberá tener operaciones necesarias para registrar agentes, cancelar o modificar su registro, buscar sobre todos los registrados y obtener su descripción. Además podrá ordenar al AP suspender un agente, terminarlo, crearlo, reanudar su ejecución, invocarlo o ejecutarlo.

Sistema de transporte de mensajes (MTS)

Es el método de comunicación por defecto entre agentes en distintas APs.

Plataforma de agentes (AP)

Proporciona la infraestructura física en la que se desenvolverán los agentes. La AP consistirá en la máquina o máquinas, el sistema operativo, el software de soporte, el DF, AMS y MTS mencionados anteriormente y los agentes.

Software

Describe todo lo que no tiene que ver directamente con el agente, aunque es una colección de instrucciones ejecutables y accesibles por ellos para, por ejemplo, añadir nuevos servicios o adquirir nuevos protocolos de comunicación.

Los agentes FIPA existen físicamente en una AP y utilizan las facilidades que ofrece para llevar a cabo sus funciones. En este contexto, un agente, como un proceso software físico, tiene un ciclo de vida que debe ser administrado por la AP. Los estados que se creen necesarios pueden observarse en la figura 4.9.

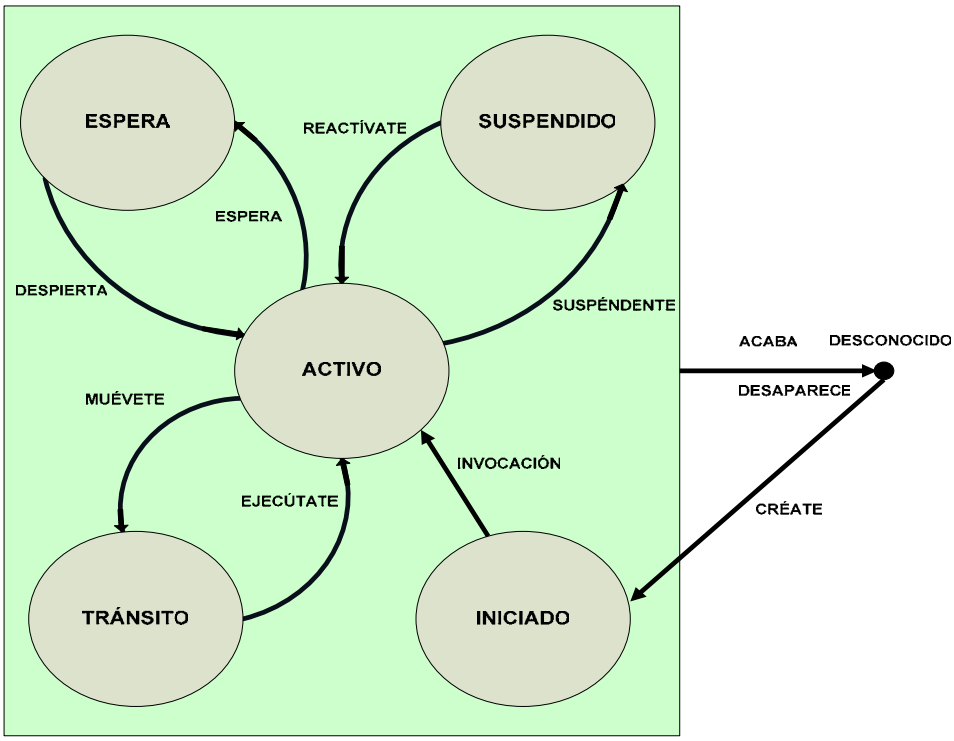


Figura 4.9 Ciclo de Vida de un Agente [FIPA04]

4.4.3 Clasificación de los agentes en un sistema de agentes cooperativos (SMA)

Se pueden distinguir tres tipos de agentes:

- **Agentes proveedores**, los cuales constituyen la base de la cadena de consumo de la información y servicios. Son agentes productores, que proporcionan capacidades, como por ejemplo servicios de búsqueda de información a sus usuarios y a otros agentes.
- **Agentes solicitantes**. Éstos consumen información y servicios ofertados por agentes proveedores en el sistema.

- **Agentes intermediarios o intermedios**, cuya misión es mediar para que pueda tener lugar una correcta comunicación entre solicitantes y proveedores. En [Dec97] definen una agente intermedio como aquél que ayuda a otros a localizar y conectar con agentes proveedores de servicios. Es necesario un mecanismo para avisar, encontrar, fusionar, usar, presentar, gestionar y actualizar los servicios y la información de los agentes. Se propuso la noción de este tipo de agentes para proporcionar estas necesidades. Los agentes intermedios son entidades a las que otros agentes comunican sus capacidades y que ni proveen ni solicitan servicios desde el punto de vista de la transacción que se está considerando en ese momento. La ventaja de los agentes intermedios es que permiten a los SMA operar de una forma robusta a pesar de tener que afrontar la aparición, desaparición y movilidad de los agentes [Flo99]. Para que pueda llevarse a cabo una correcta mediación, los proveedores tienen que registrar sus capacidades ante uno o varios agentes intermedios. Los solicitantes o consumidores pueden solicitar a un agente intermedio quién de los posibles proveedores puede llevar a cabo un determinado servicio, o la intermediación del mediador para la realización del servicio.

Hay varios tipos de agentes que entran dentro la definición de agentes intermedios. Estos tipos de agentes, descritos a continuación, están definidos tan vagamente que a veces es difícil hacer una diferenciación clara entre ellos.

- **Facilitadores**: agentes a los cuales otros agentes subordinan su autonomía a cambio de sus servicios. Los facilitadores pueden coordinar las actividades de los agentes y pueden satisfacer peticiones en beneficio de sus agentes subordinados.
- **Mediadores**: agentes que explotan el conocimiento codificado para crear servicios de más alto nivel para las aplicaciones.
- **Pizarras**: Repositorio de agentes que recibe y trata peticiones de proceso para otros agentes.
- **Brokers**: agentes que reciben peticiones y realizan acciones usando servicios de otros agentes en conjunción con sus propios recursos.
- **Emparejadores (Matchmakers)** y **páginas amarillas**: agentes que asisten a los que solicitan un servicio para buscar un proveedor de servicios, basándose en las capacidades comunicadas anteriormente.

Los dos últimos tipos de agentes tienen su correspondencia en los modelos de intermediación denominados "*brokering*" y "*matchmaking*" respectivamente [Syc01]. El primero de ellos se caracteriza por la inexistencia de una comunicación directa entre el proveedor y el solicitante. Las tareas del agente *broker* son las de contactar con el proveedor, negociar, controlar la transacción y devolver los resultados al solicitante. Por el contrario, en los modelos de *matchmaking,* el resultado devuelto por el agente *matchmaker* es una lista o referencia de proveedores que pueden proporcionar el servicio, y en estos casos, es el propio solicitante el encargado de contactar y negociar con el proveedor del servicio (Ver figura 4.10).

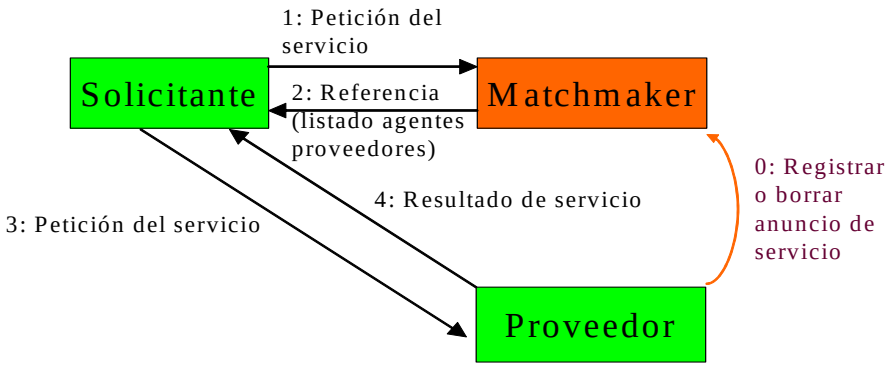


Figura 4.10 Intermediación de tipo Matchmaking [Syc01]

4.4.4 Estado actual de las tecnologías de agentes

La aplicación del paradigma de agentes al desarrollo de diferentes aplicaciones requiere el desarrollo de metodologías cuyo enfoque no solo sean los componentes internos de los agentes que intervienen sino que además contengan modelos y métodos para todo tipo de actividades a través del ciclo de vida del software.

Debido a las similitudes entre agentes y objetos se ha planteado comúnmente que las metodologías de orientación a objetos pueden ser útiles en el desarrollo de sistemas basados en agentes. Sin embargo, los agentes poseen características que no son plenamente cubiertas por metodologías O.O. [Omi00]

Una de las principales diferencias entre agentes y objetos es la autonomía, la cual se refiere al control que los agentes tienen sobre si mismos, es decir, la capacidad de éstos para decidir cuando desarrollar o no una acción solicitada. Por su parte, los objetos no tienen control sobre si mismos ya que una vez un método público es invocado, se desarrollan las correspondientes acciones.

En los últimos años han aparecido diferentes aproximaciones que tratan de presentar una metodología para el desarrollo de sistemas multiagente. En [Bot03] se presenta una visión general de esas propuestas, tales como AUML (*Agent Unified Modeling Language*), GAIA, MASSIVE (*Multi-Agent Systems Interactive View Engineering*), MAS-CommonKADS Tropos, MaSE (*Multiagent System Engineering*), MESSAGE

(*Methodology for Engineering Systems of Software Agents*), DESIRE, BDI y Burmeister.

Sin embargo, según Dignum et al. [Dig03] ninguno de las metodologías de ingeniería de software orientada a agentes existentes actualmente cumple los requisitos y consideraciones requeridas para este tipo de metodologías.

Un estudio reciente que resume los protocolos de interacción necesarios para la comunicación entre diferentes agentes, arquitecturas de desarrollo y aplicaciones de agentes se encuentra disponible en [Man04]. Por otra parte, hay muchos entornos de desarrollo de plataformas multiagente, tanto no comerciales como comerciales: JADE, ZEUS, Tryllian, Aglets Software Development Kit, Ajanta, FIPA-OS, Grasshopper etc. A su vez numerosos autores han establecido comparativas entre este tipo aplicaciones como las propuestas en [Ise99] y [Ngu02].

De entre las plataformas no comerciales destaca por sus características el entorno JADE [Bell03].

4.4.5 JADE (Java Agent Development Framework)

JADE incluye dos elementos principales: una plataforma de agentes que cumple los estándares FIPA y un paquete para desarrollar agentes Java.

Desde una perspectiva general se describe como "un marco de trabajo para el desarrollo de software, orientado al despliegue de sistemas multiagentes y aplicaciones relacionadas con los estándares del FIPA para agentes inteligentes" [JADE04]. De acuerdo con Rimaza [Rim03], desde su concepción se buscó implementar las propiedades señaladas por Wooldridge [Woo99] y cumplir con las especificaciones de FIPA.

JADE ofrece una plataforma distribuida para aplicaciones basadas en agentes, además de herramientas visuales para la administración de los agentes, depuración, movilidad y ejecución. Se dice que es distribuida ya que permite ejecutar una plataforma de agentes en varias máquinas virtuales de Java, cada una ejecutando un contenedor de agentes. JADE implementa los estándares FIPA como:

- Sistema de administración de agentes (AMS),
- Facilitador de directorios (DF),
- Canal de comunicación entre agentes (ACC),
- Agentes JADE

El diseño de los agentes en JADE, además de cumplir la especificación FIPA busca modelar las propiedades teóricas de los agentes.

Siguiendo la propuesta de la FIPA para el diseño de una arquitectura de SMA, los agentes residen en un entorno predefinido que se llama plataforma. Cada plataforma está dividida en contenedores y cada uno contiene agentes. Los contenedores pueden entenderse como dominios de agentes. Por ejemplo, si hablásemos de centros de control de tráfico, podríamos pensar que cada contenedor podría ser una ciudad que mantiene una serie de agentes que son centros de control de tráfico.

En la figura 4.11 se pueden observar cómo diferentes *hosts* tienen sus propios elementos y comparten el entorno de comunicación. En cada *host* se ejecuta una máquina virtual de Java (JVM) denominada contenedor (*container*). Cada contenedor provee un entorno de ejecución completo que permite a los agentes ejecutarse concurrentemente. En el contenedor principal (*Main Container*) es donde residen el AMS y DF y donde el registro RMI (*Remote Method Invocation*) es creado, el cual es usado internamente por JADE. El resto de contenedores de agentes conectan con el contenedor principal. La plataforma JADE proporciona un entorno seguro que permite que los agentes se conozcan y se puedan comunicar con un lenguaje de comunicación (FIPA-ACL).

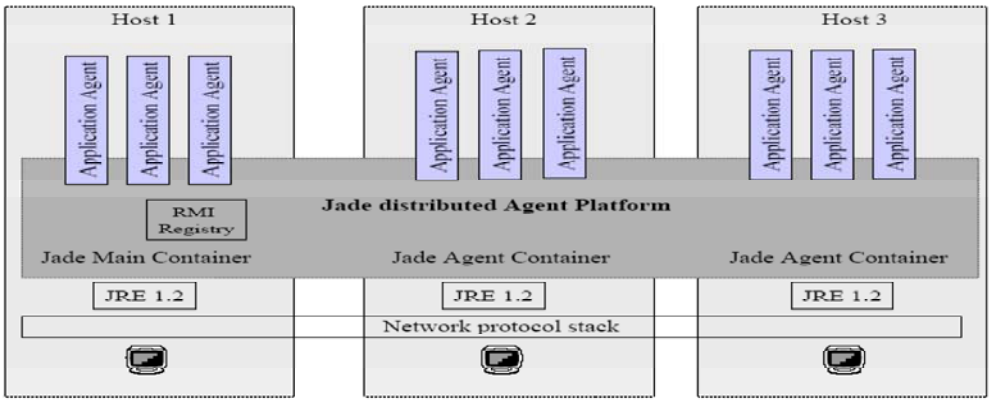


Figura 4.11 Plataforma de agentes JADE distribuida sobre varios contenedores [JADE04]

4.5 WEB SEMÁNTICA

La Web Semántica *"es una extensión de la Web actual en la qué se proporciona la información con un significado bien definido, y se mejora la forma en la que las máquinas y las personas trabajan en cooperación"* [Ber01]. Otras definiciones encontradas en la literatura pueden ser examinadas en [Her03] y [W3C01].

La Web Semántica propone superar las limitaciones de la Web actual mediante la introducción de descripciones explícitas del significado, la estructura interna y la estructura global de los contenidos y servicios disponibles en la WWW. Tiene como visión la búsqueda de una Web más inteligente en la que se pueda conseguir una comunicación efectiva entre ordenadores y centra sus esfuerzos principalmente en la búsqueda de descripciones enriquecidas semánticamente para los *datos* en la Web. En este sentido, se promueven descripciones que incluyan no solo las estructuras de datos, sino también las relaciones existentes con otros conceptos, las restricciones, reglas que permitan realizar inferencia, etc. Así mismo se promueve la definición y reutilización de vocabularios u ontologías de conceptos que faciliten el procesamiento por parte de las máquinas [Ber01].

Tal y como aparece reflejado en [Mal02], está previsto que la Web Semántica sea un lugar donde los datos puedan ser compartidos y procesados tanto por herramientas de manera automatizada como por la gente. La clave subyace en la automatización y la

integración de los procesos a través de lenguajes legibles por máquinas. En orden a poder usar y enlazar la gran cantidad de información disponible en la Web, los agentes software deben de ser capaces de comprender la información, es decir, los datos deben de estar escritos haciendo uso de una semántica legible y entendible por las máquinas. Por tanto, en los documentos XML, deberá de añadirse semántica adicional para que los programas software puedan establecer el significado de las etiquetas de dichos documentos.
Tim Berners Lee, en el artículo [Ber01] menciona cuatro componentes o características básicas necesarias para la evolución de la Web Semántica. Estos componentes son:

- Expresar significado.

La Web Semántica debe brindar una estructura y añadir semántica al contenido de las páginas web, creando un entorno donde agentes software puedan viajar de una página a otra llevando a cabo sofisticadas tareas para los usuarios.

-Acceso a representaciones del conocimiento.

La Web Semántica debe encargarse de resolver las limitaciones de los sistemas de representación de conocimiento tradicionales creando lenguajes de reglas suficientemente expresivos como para permitir a la Web razonar tan ampliamente como se desee.

-Ontologías.

Para conseguir que los computadores sean mucho más útiles, la Web Semántica extiende la Web actual con conocimiento formalizado y datos que son procesados por ordenadores. Para ser capaz de buscar y procesar información relativa a alguna materia de interés, los programas necesitan información que haya sido modelada de una forma coherente. Una ontología modela todas las entidades y relaciones en un dominio. La ontología es necesaria para la representación del conocimiento.
La clave de las ontologías es que pueden ser compartidas y por lo tanto incrementan en eficiencia e interoperabilidad. Sin embargo, se puede dar el caso en el que dos organizaciones distintas usen dos nombres diferentes para identificar el mismo concepto, es decir, las ontologías sean distintas. En tales casos, la habilidad para asociar los términos de una y otra (*mapping o mapeado*) es crucial para mantener las ventajas de la Web Semántica.

-Agentes

La potencia real de la Web Semántica se conoce cuando agentes capaces de manejar contenido semántico son usados para recoger y procesar información Web e intercambiar los resultados con otros agentes. Herramientas como el intercambio de pruebas o la firma digital asegurarán que los resultados intercambiados entre agentes sean válidos y se pueda confiar en ellos.

4.5.1 Estructura de Capas de la Web Semántica

Tim Berners-Lee [Ber98] ideó una infraestructura de lenguajes y mecanismos para poder llevar a cabo la idea de la Web Semántica. Esta infraestructura se puede esquematizar en diferentes capas o niveles. Esta estructura o esquema de capas que se

ha definido para la Web semántica, fue presentada por Tim Berners-Lee durante *XML Conference* de 2000 (ver figura 4.12):

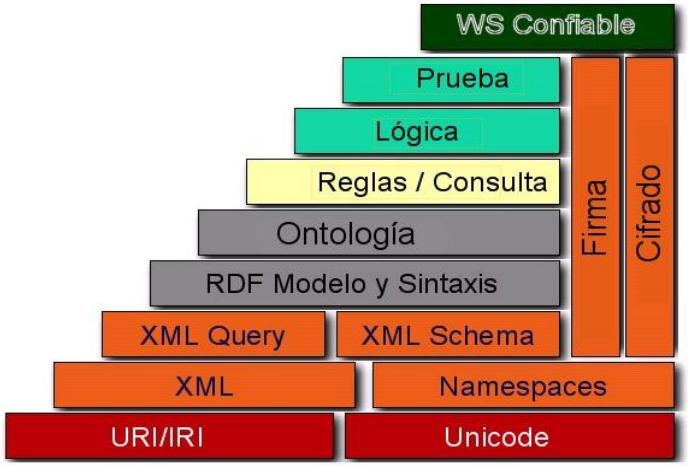


Figura 4.12 Esquema de la Web Semántica [Ber04]

La descripción de la figura 4.12 la haremos atendiendo a las diferentes capas que la conforman:

Las capas Unicode y URI (*Uniform Resource Identifier*) aseguran que se usen conjuntos de caracteres internacionales y aporten significado para identificar los objetos en la Web Semántica.

La capa XML junto al *Namespaces* (Espacio de Nombres) y XML Esquema aseguran que se pueda integrar la definición de Web Semántica con los demás estándares basados en XML.

Con RDF (*Resource Description Framework*) y RDFSchema es posible hacer declaraciones sobre objetos con URIs y definir los vocabularios que pueden ser referenciados por una URI. Ésta es la capa donde podemos definir tipos a los recursos y a los enlaces.

La capa de Ontología da soporte a la evolución de vocabularios compartidos y permite definir relaciones entre conceptos.

Como en todo conocimiento, se deben de poder especificar reglas que sirvan a los razonadores como métodos para inferir sobre el conocimiento, y de esta forma obtener nueva información. La capa Lógica está basada en la posibilidad de definir reglas de tipo “antecedentes a consecuentes” y hechos.

Las últimas capas, Pruebas y Confianza están todavía en fase de investigación. La capa anterior tiene como objetivo la definición de las reglas, para que la capa de pruebas junto a la capa de confianza las evalúen y permitan establecer si son o no confiables. Por lo tanto, los últimos niveles del esquema consisten en establecimiento de una capa

de seguridad que permita asignar fiabilidad a determinados recursos, de manera que ésta sea comprobable por agentes Web.

Blaze fue uno de los primeros en introducir el término ***administración de confianza*** (*trust managemen*t). Aunque priorizaba soluciones de seguridad para aplicaciones de red tenia una noción implícita de administración de confianza basado en certificados PGP (*Pretty Good Privacy*) o X.509 de llave pública. Blaze define *administración de confianza* como una aproximación unificada para especificar e interpretar políticas de seguridad, credenciales, relaciones, las cuales permiten autorización directa de acciones criticas de seguridad [Bla96]

Como indicaron Berners-Lee, Hendler y Lassila [Ber01], una de las características fundamentales de la Web Semántica serían las firmas digitales. El grupo de trabajo XML *Signature* [SIG04], ha desarrollado una especificación, con sintaxis XML, para la representación de firmas digitales en recursos web [Eas02], que alcanzó el grado de Recomendación W3C en febrero de 2002.

Sin embargo, aunque las firmas digitales resultan de utilidad para describir la autenticidad de las relaciones de autoría entre individuos y recursos, no describen la confianza de los usuarios y agentes inteligentes sobre dichos autores y, por lo tanto, sobre el contenido de sus recursos [Hen02]. A su vez la infraestructura utilizada por la firma digital tiene un carácter centralizado, mientras que la Web Semántica pretende tener un carácter completamente descentralizado. La solución a este problema, podría ser la denominada "*Web of Trust*", que basa su funcionamiento en el concepto de "confianza" o "fiabilidad". Este concepto es definido por Reagle en [Rea02] como "el grado en que un agente considera un aserto como verdadero para un contexto dado".

Los individuos, identificados por medio de URIs, expresarían su grado de confianza (o desconfianza) sobre otros individuos, que a su vez harían lo mismo sobre otros, lo que daría como resultado extensas e interoperables redes de confianza procesables por agentes inteligentes.

En la actualidad existen investigaciones iniciales acerca de la Web de Confianza que se han realizado en Estados Unidos y Europa. Dentro de la bibliografía consultada destaca "*A Survey of Trust and Reputation Systems for Online Service Provision*" [Jos04] donde se hace una descripción de los conceptos de confianza y reputación y los avances que han tenido lugar en los últimos años en este tema. Otro trabajo importante es "*A survey of Trust in Internet Applications*" [Tyr00] en el que se presenta un resumen de conceptos de confianza, así como su relación con tecnologías como PGP, plataformas para selección de contenidos (PICS) y REFEREE, entre otras.

4.5.2 Lenguajes de marcado ontológico

Los siguientes lenguajes de especificación de ontologías se diferencian de los anteriores debido a que fueron desarrollados orientados para ser utilizados en la Web. Es decir estos lenguajes anotan su código en el de las páginas Web para que agentes Web o programas que interactúan con ellas puedan extraer la información semántica que esa Web tiene. Destacamos a los principales:

XML, DTD y XML Schema

Desde la aparición de XML en 1998, se han definido multitud de estándares para modelizar información en dominios específicos como las finanzas (XBRL, RIXML, FpXML, ebXML, etc.), el periodismo (p.e. NewsML, XMLNews, PRISM), la enseñanza (SCORM, IEEE LOM y otros), o la medicina (NLMMedline, SCIPHOX, CDA, etc.), entre otros muchos campos [Cas03].

Actualmente es el lenguaje en el que *e-busines*s se está apoyando para mejorar sus servicios, y en el que en un futuro cercano será utilizado por todos en la Web. Fue desarrollado por el grupo de trabajo *XML* del W3C, el cual estableció la versión 1.1 como recomendación el 4 de febrero de 2004 [XML04].

La característica principal de *XML* es su *flexibilidad*: todos podemos escribir un documento *DTD* (*Document Type Definition*) o un *XML Schema* (2ª edición, recomendación de 28 de octubre de 2004) que defina la estructura de un documento *XML* que representa la información en la forma deseada. Aunque se incluye aquí como un lenguaje de especificación de ontologías, en realidad no lo es, no está orientado a cumplir ese objetivo sino el intercambio de datos. Este lenguaje permite únicamente extraer datos, pero no su contenido semántico. Desde este punto de vista, XML presenta una serie de problemas por basar su entendimiento en la estructura de los documentos, ya que cuando un esquema XML cambia, por ejemplo, introduce nuevos elementos intermedios que pueden invalidar alguna consulta anterior que haya sido basada en la estructura.

Además, también falla en un objetivo bastante importante, no mencionado anteriormente, *la escalabilidad*:

- El orden en el cual los elementos aparecen en el documento XML es significativo y muchas veces necesario. Esto es altamente antinatural en el mundo de los metadatos. Da igual el orden con el cual se presenten los distintos atributos de una incidencia de tráfico.
- Además, mantener el orden correcto de los elementos de datos es costoso y difícil.

Sin embargo XML es el pilar en el que se sustentan el resto de lenguajes o tecnologías de la Web. XML supone un formato universal: “Todo debe estar escrito en XML” [Llo05].

Otra de las funciones actuales que tiene es encapsular información o partes de una ontología para que sea intercambiada a través de los diferentes protocolos de intercambios de datos *XML* a través de la Web (por ejemplo SOAP).

La familia de tecnologías asociadas a XML es muy extensa destacando XPath, XPointer, XLink, XSL y CSS principalmente [Mar05].

XOL (*XML-Based Ontology Exchange Language*) [Kar99] como indica su nombre es un lenguaje para establecer un formato de intercambio de ontologías, pero no para su especificación. Este lenguaje se emplea para el intercambio de ontologías entre bases de datos, aplicaciones de ontologías etc. *XOL* define una sintaxis *XML* de *OKBC*-

Lite, permitiendo así el intercambio de datos vía Web. Sin embargo, al estar basado en *OKBC* (*frames)*, solo se pueden definir clases y atributos.

SHOE (*Simple HTML Ontology Extensión*) [SHOE99] desarrollado para ampliar *HTML* incorporando semántica en los documentos en este formato para que las máquinas los procesasen. Consiste en definir una ontología que describa una clasificación válida de objetos y las relaciones existentes entre ellos, para de esta forma añadir información descriptiva. Las ontologías en *SHOE* son una jerarquía de clases junto un conjunto de relaciones entre ellas y otro conjunto de reglas de inferencia en forma de cláusulas de Horn, por lo que nos permite realizar afirmaciones sobre la información que las páginas Web poseen. Se trata de un lenguaje de especificación de ontologías que pretende potenciar y mejorar los motores de búsqueda de los buscadores de Internet.

RDF(S) (*Resource Description Framework* (*Schema*)) [RDF04a] [RDF04b] desarrollado por el World Wide Consortium (W3C), cuya recomendación fue establecida el 10 de febrero de 2004, establece la sintaxis y estructura que permite la descripción de metadatos y permite que el significado sea asociado con los datos a través de RDF Schema [RDFS04], el cual facilita la definición de ontologías específicas de dominio.

Se trata de una infraestructura que permite la codificación, intercambio y reutilización de metadatos estructurados. Esta infraestructura permite la interoperabilidad de metadatos mediante el diseño de mecanismos que dan soporte a las convenciones comunes de semántica, sintaxis, y estructura. RDF hace uso de XML como sintaxis común para el intercambio y proceso de metadatos, proveyendo independencia, extensibilidad, validación, legibilidad humana, y la habilidad para representar estructuras complejas. RDF explota estas características imponiendo a su vez, estructuras que permiten expresividad semántica inequívoca. Provee además un medio para publicar vocabularios[11] tanto legibles por los humanos como capaces de ser procesados por las máquinas, y éstos pueden ser reutilizados, extendidos o refinados para adaptarlos a los diferentes dominios específicos según sus requerimientos.

Las fuentes en las que se basaron para desarrollar *RDF* provienen de la Comunidad de Normalización de la Web en forma de metadatos *HTML* y *PICS*, la Comunidad Bibliotecaria, la Comunidad de documentos estructurados como *SGML* y sobre todo *XML*, así como también la comunidad de representación de conocimiento.

RDF basa su modelo en tres partes: Recursos (sujeto) que son todo aquello de lo que se puede decir algo y son referenciados por un identificador único de recursos (URI); Propiedades (predicados) que definen atributos, aspectos, características o representan una relación que describe a un recurso; y Declaraciones (objeto) los cuales nos sirven para dar valores a las propiedades de un recurso específico. El objeto de un estamento puede ser un recurso o un literal, por ejemplo, un recurso especificado por un URI, o bien un *string* u otras primitivas de datos definidas por XML.

RDF Schema permite la definición de jerarquías de clases (relación de subclase y subpropiedad) de objetos y propiedades (relaciones binarias) y permite la creación de restricciones (rango y dominio). Este esquema se aproxima más al concepto de

[11] Los vocabularios son el conjunto de propiedades o metadatos , definidos por las distintas comunidades de descripción de recursos.

ontología ya que se dispone de relaciones diseñadas para especificar la jerarquía de la taxonomía de conceptos que define un conocimiento.[12]
RDF(S) funciona como un modelo semántico de datos capaz de permitir preguntas referentes a su contenido y no a la estructura del documento.

OIL (*Ontology Interchange Language*) [OIL02] fue una propuesta de un lenguaje que permitiese la especificación de ontologías y que además sirviese como lenguaje de intercambio de éstas. Fue diseñado por un grupo de investigación dentro del proyecto OnToKnowledge [OTK02] basándose en los sistemas de *frames* y en los de lógica descriptiva, mezclando las propuestas de *OKBC*, *XOL* y *RDF* para obtener lo mejor de cada una de ellas. Las ontologías en *OIL* está organizada en tres niveles: un nivel objeto (donde residen las instancias); un primer nivel meta donde residen las definiciones y descripciones de la ontología y por último un segundo nivel meta donde reside un contenedor sobre la ontología, es decir, donde reside la información sobre las características de la ontología. *OIL* también soporta axiomas por lo que el razonamiento sobre ontologías de este tipo es posible.

DAML+OIL [Har01].

Los usuarios de *RDF* y *RDFS* conforme utilizaban estos lenguajes para expresar los metadatos de sus recursos, observaban que ambos lenguajes describían un conjunto escaso de facilidades para expresar estos metadatos. Por ejemplo, no podían hacer uso de datatypes del *XMLSchema* [XMLS01], no podían hacer enumeraciones etc. La comunidad de usuarios de la Lógica también vieron en *RDF* una herramienta para expresar sus recursos, pero también lamentaban que no existiesen facilidades para permitir la inferencia sobre las descripciones *RDF*. Tenía que surgir una evolución de este lenguaje que permitiese el razonamiento sobre las descripciones.

Por estos motivos, en el año 1999 se inició el programa DAML (*DARPA Agent Markup Language*) de DARPA (*Defense Advance Research Projects Agency)* con el objetivo de proveer los fundamentos de la siguiente generación de Web o Web Semántica [Hen00].

Desde este programa surgió el lenguaje *DAML* patrocinado por el gobierno estadounidense, y a partir de este apareció el lenguaje *DAML-ONT* como un lenguaje más sofisticado para realizar descripciones de recursos. *DAML* no es un lenguaje que surgiese del Consorcio W3C pero si muchos de los desarrolladores de este lenguaje pertenecen a él. En poco tiempo unieron sus esfuerzos los desarrolladores de *DAML* con los de *OIL* para favorecerse de los sistemas de clasificación basados en *frames* de este último.

El resultado final fue el lenguaje *DAML+OIL*, un lenguaje que a pesar de tener muchas coincidencias con *OIL*, también tiene sus diferencias, la principal es que se trata de un lenguaje que se aleja de los ideales de *frames* de *OIL*, para acercarse más a una base de lenguajes de lógica descriptiva.

Por lo tanto *DAML+OIL* es un lenguaje que nos ofrece más expresiones sofisticadas para las descripciones de clasificaciones y propiedades de los recursos que las que ofrecía *RDF* y *RDFS*. *DAML+OIL* es la evolución de *RDFS*, en el que se redefinen muchas de sus descripciones y se añaden muchas otras para mejorar el

[12]Mientras que la relación entre XML y XML Schema es una relación de control sintáctico, la relación entre RDF y RDFS es de control semántico. [Ber98]

lenguaje y aportar propiedades y mecanismos para que el lenguaje defina ontologías que después pueden ser empleadas por sistemas de razonamiento para poder inferir sobre la información.

OWL (Ontology Web Language) [OWL04] surge del W3C como la búsqueda de un lenguaje de especificación de ontologías que sirva como estándar para todos los investigadores de la Web Semántica. Se le pretende dar tantas funcionalidades como las que posee *DAML + OIL*, aunque diferenciándose en algunas.

OWL también es una extensión del modelo *RDFS*, y en ella se redefinen recursos y propiedades, al igual que se le añaden nuevas. Con OWL se pueden definir clases mediante restricciones a otras clases, o con operaciones boleanas sobre otras clases, hay nuevas relaciones entre clases como la inclusión, disyunción y la equivalencia, se pueden definir restricciones de cardinalidad en propiedades o dar propiedades sobre las relaciones (transitiva, simetría) así como permitir clases enumeradas.

OWL se decidió separar en tres niveles:

- *OWL Lite*: la versión más simple para los programadores principiantes. Permite la jerarquía de clasificación y las restricciones simples.
- *OWL DL*: esta versión ya tiene todo el vocabulario *OWL* completo. Las limitaciones son que las clases no son instancias ni tipos y los tipos no son ni instancias ni clases. No permite restricciones de cardinalidad en propiedades transitivas. Posee gran expresividad sin perder las propiedades de completitud y decidibilidad.
- *OWL Full*: esta versión también incluye todo el vocabulario de *OWL* pero en este caso no hay limitaciones para explotar todo su potencial. Sin garantías computacionales.

OWL Full se considera la más completa de todas y se supone una extensión de DL que a su vez es una extensión de Lite, por lo que toda ontología correcta en OWL Lite es una ontología correcta en OWL DL, y toda conclusión correcta en OWL Lite es una conclusión correcta en OWL DL (pero no a la inversa). De la misma manera esto también ocurre con OWL DL y OWL Full respectivamente.

Este lenguaje también posee funcionalidades de razonamiento para las ontologías.

Son varias las diferencias existentes tanto en el modelo semántico como en la sintaxis entre *DAML + OIL y OWL,* los dos lenguajes de especificación de ontologías más completos:

DAML + OIL se aproxima más a la versión OWL DL, aunque en este caso la correspondencia con los lenguajes DL difiere. Si al primero le corresponde SHIQ DL (H soporte para reglas de jerarquía (Hierarchy), I reglas inversas (Inverse) y Q restricciones de cardinalidad cualificadas (Qualified)) con la adición de clases definidas existencialmente y dominios concretos o *datatypes*, al segundo SHOIN(D). A su vez a OWL-Lite le corresponde SHIF(D). [Hor03] [OWL04b].

A su vez y a nivel sintáctico destacaremos un aspecto de gran importancia:

- Posibilidad en *DAML + OIL* de realizar restricciones numéricas cualificadas en propiedades de las clases (aporta propiedades y predicados que lo permiten), es decir, se pueden hacer expresiones del tipo "Todos los hijos de X son de tipo persona". Aunque las diferencias sintácticas son pocas, si que lo son semánticamente:
    1. Mediante restricciones cualificadas y haciendo uso del cuantificador existencial podemos especificar por ejemplo, el tipo de "ciclomotores" que posee exactamente dos "ruedas", pero puede tener otros elementos de otro tipo.
    2. Sin restricciones cualificadas y de nuevo haciendo uso del cuantificador existencial, solamente podremos especificar el tipo de "ciclomotores" que tienen dos elementos de los cuales al menos uno es de tipo "rueda".

  La sencillez y la claridad han motivado que al final la recomendación de OWL dada por W3C no recoja este tipo de expresividad, aunque numerosos autores y un servidor pensemos que las restricciones cualificadas son importantes a la hora de expresar conocimiento.

OWL es desde febrero de 2004 una recomendación de W3C, por lo que la mayoría de desarrolladores e investigadores de la Web Semántica van a centrar ya sus esfuerzos en desarrollar herramientas y sistemas orientados a este lenguaje. Un ejemplo puede ser que el *DAML* Consortium de DARPA desde junio de 2003 está migrando sus investigaciones a este lenguaje.

La tabla 4.3 muestra una comparativa entre los lenguajes de ontologías basados en Web más actuales y empleados.

	XML DTD	XML SCHEMA	RDF(S)	DAML+OIL	RDF(S) 2002	OWL
Listas Limitadas				**X**	**X**	**X**
Restricciones cardinalidad	**X**	**X**		**X**		**X**
Expresiones de clases				**X**		**X**
Datatypes		**X**		**X**	**X**	**X**
Clases definidas				**X**		**X**
Enumeraciones	**X**	**X**		**X**		**X**
Equivalencias				**X**		**X**
Extensibilidad			**X**	**X**	**X**	**X**
Semántica formal				**X**	**X**	**X**
Herencia			**X**	**X**	**X**	**X**
Inferencia				**X**		**X**
Restricciones locales				**X**		**X**
Restricciones cualificadas				**X**		
Reificación			**X**	**X**	**X**	**X**

Tabla 4.3: Comparativa entre lenguajes de especificación de ontologías [Daml02]

Otras tablas comparativas pueden ser encontradas en [Onto02b] y [Gom04].

Por tanto, *DAML+OIL* y *OWL* son los lenguajes de especificación de ontologías más completos, principalmente porque son lenguajes basados fundamentalmente en lógica descriptiva (*descriptions logics).*

4.5.2.1 ¿Por qué utilizar DAML + OIL u OWL en lugar de RDF(S)?

Para tomar esta decisión, hay que tener en cuenta varios estudios realizados hasta la fecha sobre los distintos lenguajes de marcas de tipo semántico [Cor00], [Gil02], [Rib00]. A modo de resumen, podemos afirmar que RDF(S) posee algunas deficiencias como:

- Posee semántica solo para algunos términos.
- Solo es posible aplicar restricciones de dominio/rango sobre las propiedades.
- Ninguna propiedad de propiedad.
- No existe equivalencia ni disyunción.
- No es posible establecer condiciones necesarias y suficientes.
- No es posible restringir el rango de las propiedades mediante restricciones de cardinalidad.
- Necesidad de mayor poder de inferencia.

Para entender la discriminación de este lenguaje se harán uso de “vistas” parciales sobre la ontología de tráfico construida en esta tesis.

Por ejemplo, establecer que la propiedad “tipo_Incidencia” (dominio Sucesos) solamente puede contener un valor como máximo, o dicho de otra forma “una incidencia solo podrá ser de un tipo determinado y no de varios”, será algo imposible de expresar mediante RDF(S), pero no así con OWL:

```
<owl:ObjectProperty rdf:ID="tipo_Incidencia">
<rdf:type rdf:resource="http://www.w3.org/2002/owl#FunctionalProperty"/>
        <rdfs:domain rdf:resource=#Incidente"/>
        <rdfs:range rdf:resource=#Tipo_Incidente"/>
</owl:ObjectProperty>
```

O pensemos por ejemplo en relación existente entre un suceso y una causa, donde esta última entidad sirve para definir en el modelo la causa de un suceso, es decir cual ha sido el motivo de su aparición. La relación tal y como se aprecia en el modelo ER (figura 4.13), es una relación M:M denominada “causa”.

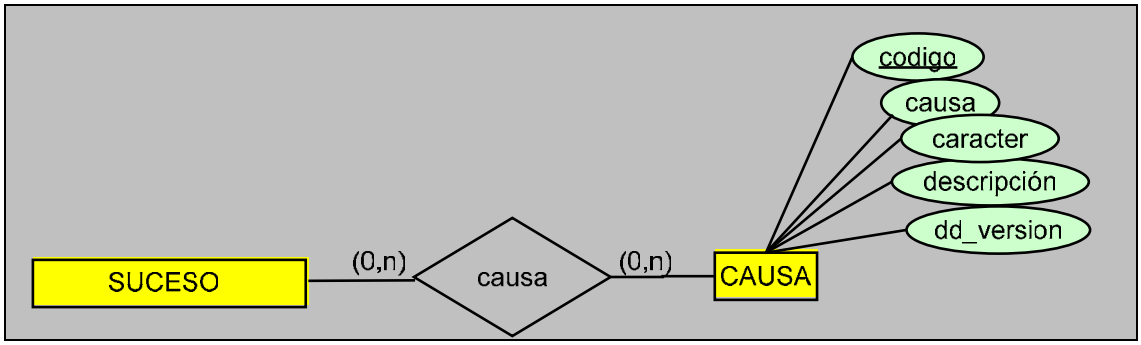


Figura 4.13 Relación de causalidad en el esquema conceptual

- Un SUCESO puede ser causado_por varias CAUSAs o ninguna

  (DL) Suceso $\subseteq (\geq 0\ causa.Causa)$

Podríamos distinguir entre aquellas "causas" que son causa directa de "sucesos" y aquéllas que sin embargo no han incidido directamente. Para describirlo se introduce una nueva propiedad o relación que denominaremos "causa_directa", y mediante la cual se podrá especificar si una causa afecta directamente a un suceso.

Suceso $\subseteq (\geq 0\ causa\_directa.Causa)$

Se puede observar en las características de estas dos relaciones, que aunque coinciden en rango, dominio, grados de cardinalidad etc., sin embargo, hay algo que hace que su comportamiento en el modelo sea diferente y es el rasgo de transitividad. La nueva relación no será transitiva, mientras que la original si.

Esto reflejado mediante Lógica Descriptiva quedará de la siguiente forma:

$causa^{+} \subseteq causa$

Sin embargo, de nuevo esto es algo imposible de representar mediante RDF(S), mientras que su especificación con los otros dos lenguajes si que es posible:

```
<owl:ObjectProperty rdf:ID="causa">
<rdf:type rdf:resource="http://www.w3.org/2002/owl#TransitiveProperty"/>
        <rdfs:domain rdf:resource=#Suceso"/>
        <rdfs:range rdf:resource=#Causa"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="causa_directa">
        <rdfs:domain rdf:resource=#Suceso"/>
        <rdfs:range rdf:resource=#Causa"/>
</owl:ObjectProperty>
```

Podemos representar la instancia de la incidencia en OWL de la siguiente forma:

```
<?xml version='1.0'?>
<rdf:RDF
        xmlns:rdf ='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
        xmlns:xsd ='http://www.w3.org/2000/10/XMLSchema#'
        xmlns:owl='http://www.w3.org/2002/07/owl#'
        xmlns:sucesos ='http://robotica.uv.es/Traffic/Sucesos#'
        xmlns     ='http://robotica.uv.es/Traffic/Incidentes#' >

<sucesos:Incidente rdf:ID='INCI_105.0_0.0_13-07-2004_07_24'>
        < sucesos:tipo_Incidencia rdf:resource='http://robotica.uv.es
/Traffic/sucesos#METEOROLOGICA'/>
        <sucesos:causa rdf:resource='http://robotica.uv.es
/Traffic/Relaciones_Sucesos.daml#VIENTO'/>
        <sucesos:provincia rdf:resource='http://robotica.uv.es /Traffic/Geografia#CADIZ'/>
        <sucesos:poblacion rdf:resource='http://robotica.uv.es /Traffic/Geografia#TARIFA'/>
        <sucesos:nivel_servicio rdf:resource='http://robotica.uv.es /Traffic/sucesos#VERDE'/>
        <sucesos:carretera rdf:resource='http://robotica.uv.es /Traffic/Vias#N-340'/>
        <sucesos:pKini><xsd:decimal><rdf:value>105</rdf:value></xsd:decimal></sucesos:pKini>
        <sucesos:pKfin><xsd:decimal><rdf:value>0</rdf:value></xsd:decimal></sucesos:pKfin>
        <sucesos:latitude>39.5106</sucesos:latitude >
        <sucesos:longitude>-0.41024</sucesos:longitude >

</sucesos:Incidente>
</rdf:RDF>
```

Y con ello, mantener su semántica, lo que nos permitirá formular preguntas como las que a continuación se detallan.

a) Cuestiones directas sobre la incidencia específica:

- ¿La incidencia identificada por 'INCI_105.0_0.0_13-07-2004_07_24' es de tipo meteorológico?
- ¿En qué carretera y qué PKs específicos ha ocurrido la incidencia 'INCI_105.0_0.0_13-07-2004_07_24'?

b) Cuestiones que relacionan los elementos que caracterizan la incidencia:

- ¿La población de Tarifa pertenece a la provincia de Cádiz?
- ¿Qué población corresponde al paso de la carretera N-340 entre los PKs 105 y 0?

c) Cuestiones sobre detalles que explícitamente no aparecían en la especificación:

- ¿Qué significado tiene para las condiciones de circulación en esa localización, que el nivel de servicio sea verde?
- ¿De qué tipo es y cuáles son las características que posee la carretera N-340?

Por tanto, al ser DAML+OIL y OWL lenguajes basados en una representación XML, les hace fácilmente legibles y permiten compartir ontologías y reutilizar conocimiento orientado a la Web para definir nuevas ontologías. Además son lenguajes basados en marcos lo cual hace que sea fácil representar una ontología en términos de Orientación a Objetos. Están basados en DL y tienen una semántica bien definida.

De este modo, para las ontologías propuestas en esta tesis se ha escogido un lenguaje semántico de este tipo por los siguientes motivos[13]:

- ***Flexibilidad y Escalabilidad:*** Estas dos características son fundamentales. Mi aporte se fundamenta en el uso de ontologías para especificar parámetros en los perfiles de búsqueda de servicios. Ante la aparición de nuevos servicios pertenecientes a distintas categorías, puede motivar la necesidad de añadir, modificar o eliminar vocabularios sin tener que realizar grandes modificaciones.

- ***Eficiencia:*** Podemos preguntar al modelo de forma más eficiente que XML. En XML, habríamos de *parsear* la información siguiendo la estructura del documento conforme a un esquema.

4.5.3 Editores de ontologías

Para el desarrollo de la construcción de una ontología la principal herramienta son los editores. Los editores suelen ser desarrollados para un tipo de lenguaje específico, pero muchos de ellos incorporaron módulos para soportar otros lenguajes de especificación diferentes. Los principales editores han sido:

Ontolingua Server [Ontol04], desarrollado en 1990 por el laboratorio de sistemas de conocimiento de la Universidad de Stanford, está orientado al lenguaje *Ontolingua,* aunque posteriormente incluyeron otros lenguajes.

WebOnto [WebOnto04] fue desarrollado por el *Knowledge Media Institute* (KMI) en 1997 como editor de ontologías *OCML*, cuya característica principal es que permite la participación de varios usuarios en el desarrollo de la ontología.

Protégé2000 [Protege04] fue desarrollado por *Stanford Medical Informatic* (SMI) en la Universidad de Stanford. Actualmente es uno de los editores de ontologías más usado por investigadores para desarrollar sus ontologías, ya que es una herramienta que se actualiza con bastante regularidad y a la que se le pueden añadir módulos y *plugins* con nuevas funcionalidades. Permite que la ontología desarrollada se exporte a los diferentes lenguajes de especificación más empleados actualmente (*RDF, DAML, OWL*, etc.). La última versión lanzada ha sido la v2.1.2 en agosto de 2004 y actualmente está en fase Beta la v. 3.0.

[13] Debido a ciertas limitaciones derivadas de la escasez en herramientas (almacenamiento y razonamiento) para el tratamiento de OWL, se hará uso de DAML+OIL en el sistema prototipo sin pérdida de expresividad, por lo que las referencias a la especificación formal podrán ser indistintamente en cualquiera de los dos lenguajes.

WebODE [WebODE04] es una herramienta que basa la construcción de ontologías en el método *Methontology*, permite exportar el conocimiento a diferentes lenguajes de especificación (*RDFs, OWL, OIL, DAML + OIL*). La última versión es *WebODE 2.0.9* de noviembre de 2003.

OntoEdit [OntoEdit04] es un editor que soporta *F-Logic RDF y OIL* aunque después almacena el conocimiento en *XML*. La última versión es *OntoEdit v0.6.*

OilEd [OilEd04] es la herramienta bautizada como el "notepad" de los editores de ontologías. Basado inicialmente para el desarrollo de ontologías *OIL* y *DAML + OIL* se han ido realizando numerosas actualizaciones para que acepte la mayoría de los lenguajes de especificación actuales. Es un editor bastante utilizado por los investigadores porque aporta la posibilidad de interactuar con un razonador como *FACT* o *RACER* que permiten comprobar la consistencia de una ontología. Una de las desventajas que presenta este editor es la carencia de recursos para soportar ontologías grandes, migración e integración de otras ontologías y diferenciación de versiones.

Para la elección de las herramientas a utilizar se procedió a una revisión bibliográfica que mostrara las diferentes comparativas entre éstas, como las aparecidas en [Den02], [EON02], [Ont02] y [Gom04], teniendo en cuenta como punto de partida la necesidad de hacer uso de DAML+OIL u OWL .Tras estas revisiones y diferentes pruebas realizadas, se pudo apreciar que en su mayoría las herramientas de edición no son totalmente satisfactorias en el sentido de que no son capaces de explotar y manejar todas las características de los lenguajes, por ejemplo, el uso de tipos definidos por el usuario[14].

El editor empleado para describir la ontología de conceptos fue OilEd [Rob02], ya que permite construir el conocimiento y luego ser exportado tanto a RDFS como DAML + OIL u OWL. Permite establecer jerarquías, chequear las consistencias de las ontologías y añadir relaciones de subsumción de manera implícita. Además actualmente mediante OilEd es posible acceder a ontologías OWL almacenadas en repositorios *Sesame* utilizado en nuestro sistema, lo que permitirá migrar sin la necesidad de abordar nuevas herramientas.

El hecho de que Protégé-2000 en principio y cuando se abordó este trabajo, sólo trabajara directamente con RDF hizo desestimar este editor. Más tarde y tras la ampliación de Protégé al uso de DAML+OIL mediante plugins, se pudo comparar ambos editores, haciéndonos decantar por el primero.

4.5.4 Sistemas de almacenamiento

Otra de las herramientas más desarrolladas por los investigadores de la Web Semántica han sido los sistemas de almacenamiento de ontologías. Mediante estos sistemas es posible mantener las ontologías en bases de datos e ir añadiendo nueva información, y

[14] Gracias a la alta expresividad de los lenguajes semánticos, se pueden establecer vías alternativas a ciertas limitaciones impuestas por las herramientas de uso. Esto ha permitido establecer guías de desarrollo para este tipo de expresividad semántica, durante el desarrollo de esta tesis (Referirse a: "**Problemas: limitaciones en las herramientas de edición**" en la URL http://robotica.uv.es/tesis/javi/problemas.html y en el CD-ROM que acompaña esta memoria).

con la ayuda de razonadores probar la consistencia de la ontología. La mayoría de los sistemas de almacenamiento están orientados a descripciones de conceptos escritas en RDF, aunque algunas han ido actualizándose a los últimos lenguajes de especificación. Las principales herramientas en esta área son:

ICS-FORTH RDFSuite [Ale01]: desarrollado por el proyecto *EU C-Web* y *MesMuses*, es un conjunto de herramientas para el manejo de recursos RDF en aplicaciones Web. El objetivo de estas herramientas es parsear, validar, almacenar y consultar los recursos RDF. Para el almacenamiento, la aplicación empleada es RSSDB, una herramienta que lee descripciones RDF para almacenarlas en bases de datos objeto-relacionales como PostgresSQL, manteniendo el conocimiento y relaciones existentes en la ontología. El lenguaje de consulta que emplean es RQL.

Sesame [Bro02] es un repositorio para RDF-Schema desarrollado por *Aidministrator Nederland bv.* Tiene funciones para añadir y eliminar información escrita en RDF en los repositorios, para ser almacenada en cualquier tipo de base de datos (MySQL, Oracle etc.). Soporta los lenguajes de consulta RQL, RDQL y SeRQL, para acceder al conocimiento.

RdfDB [Dum00] implementado por R. V. Guha, es la base de datos más simple que hay para recursos RDF. Es escalable, y posee un lenguaje de consulta parecido a SQL.

Jena 2 [JENA05] [McB01] colección de herramientas desarrollado por Hewlett-Packard para la Web semántica. En esta colección, hay un parser para RDF, un API, el lenguaje de consulta RDQL, soporte para ontologías RDFS, DAML + OIL y OWL y un sistema de almacenamiento basado en bases de datos BerkeleyDB.

KAON Tool [KAON04] desarrollado por la infraestructura KAON (Karlsruhe Ontology) Semantic Web, implementa un interfaz independiente del sistema en el que se almacenarán las ontologías, ya sean cualquier base de datos o un fichero texto. Implementa un API para leer las descripciones de los recursos, emplea RQL para realizar consultas y soporta tanto ontologías DAML + OIL como RDF.

Se presentan varios estudios que detallan comparativas entre este tipo de herramientas como el encontrado en [Mag02].

4.5.5 Razonadores

Una de las herramientas más utilizadas con las ontologías son los razonadores, que sirven para realizar inferencia, a través de los conceptos y en algunos casos las instancias, para obtener nuevo conocimiento. Generalmente difieren en el lenguaje formal en el que se especifica el conocimiento, así como los lenguajes de consulta que puedan utilizar:

En la actualidad, son varios los razonadores o sistemas de inferencia basados en *lógica descriptiva* que permiten el razonamiento y la inferencia en las ontologías. Los principales son:

FaCT (*Fast Classification of Terminologies*) [FACT04] desarrollado por Ian Horrocks, que puede ser usado para chequear el grado de satisfacción de las descripciones. Permite reglas transitivas, inversas, restricciones cualificadas, jerarquías etc. Es lo suficientemente expresivo para soportar y razonar sobre cualquier base de conocimiento. Escrito en *Common Lisp*, fácilmente ejecutable por cualquier programa *Lisp* de forma local, tiene escrita un versión servidor FaCT para ser usada vía interfaz CORBA sobre cualquier sistema con acceso a la red. Actualmente es el razonador empleado por defecto por el editor de ontologías OilEd para clasificar los conceptos en una jerarquía según las descripciones que tengan.

RACER (*Renamed ABox and Concept Expression Reasoner*) [RACER04] desarrollado por Ralf Möller y Volker Haarslev en 1999, pero que ha sido renovado periódicamente hasta la fecha. Es un razonador diseñado para la Web Semántica. Permite la inferencia tanto en conceptos como en instancias, soporta ontologías escritas en RDF/RDFS/Daml/OWL (apenas tiene restricciones con estos lenguajes) y posee un lenguaje de consulta sencillo para la inferencia de instancias. Puede ser utilizado tanto por OilEd como por Protégé tanto para comprobar la consistencia de la ontología, como para hacer consultas sobre el conocimiento.

BOR [BOR02] es un razonador desarrollado por el Laboratorio Sirma del proyecto On-To-Knowledge. Tiene soporte para ambos tipos de inferencia, tanto sobre instancias como sobre conceptos: chequeo de la consistencia y del modelo de la ontología, construcción de la jerarquía de conceptos, clasificación de conceptos definidos. El razonador puede ser usado con ontologías escritas en DAML + OIL, con algunas restricciones, y con ontologías escritas en la especificación OWL Lite. Además se puede incorporar a la aplicación Sesame, para dar soporte a ontologías DAML + OIL en este tipo de repositorios y poder inferir conocimiento o simplemente recuperarlo.

DamlJessKB [JessKB04] es el razonador para DAML + OIL descrito por *National Science Foundation* (NSF), *Knowledge and Distributed Intelligence in the Information Age* (KDI) *Initiative*. Implementado usando Jess (Java Expert System Shell), basa su mecanismo en parsear la información expresada en DAML + OIL en reglas para el sistema experto como es Jess. Tiene funcionalidades para soportar la clasificación de clases definidas a partir de las reglas lógicas como las disyunciones, restricciones de valores, etc. También tiene soporte para los Data types de XML Schema y para las propiedades transitivas. Actualmente se está migrando a OWL.

Cerebra [Cerebra04] es un motor de inferencia desarrollado por *Network Inference*. Parecido al razonador de *lógica descriptiva* FaCT, solo que en este caso si que aporta soporte para la inferencia sobre instancias y *datatypes*. Soporta tanto ontologías RDF, como DAML + OIL y OWL, y como lenguaje de consulta para razonar emplea XQuery [XQuery04] Tiene versiones para trabajar conjuntamente con los editores OilEd y Protégé.

Pellet [Pellet03] Pellet es un razonador de OWL-DL basado en Java. Puede ser utilizado conjuntamente con bibliotecas del API de Jena o del OWL. Mediante su uso es posible validar, comprobar la consistencia de ontologías, clasificar la taxonomía y contestar a un subconjunto de consultas RDQL (conocido como consultas a ABox en terminología del DL). Se trata de una razonador DL basado en los algoritmos *tableaux*

desarrollados para DL expresiva. Soporta todas las construcciones del OWL DL incluyendo las relacionadas con los nominales, es decir, *owl:oneOf* y *owl:hasValue*.

Por otra parte, la lógica de primer orden (FOL), necesita de un comprobador de teoremas para especificar axiomas. Con FOL se aporta expresividad pero se pierde razonamiento, por lo que el manejo de una gran base de conocimiento y axiomas no es tratable computacionalmente. Debido a esta circunstancia su uso en un entorno como es la Web no es recomendable, dada las dimensiones que ésta posee. El principal razonador de FOL es el siguiente:

JTP [Fik02] (Java Theorem Prover) es una arquitectura para razonar sobre conocimiento descrito en DAML + OIL. Escrito en java, soporta el modelo de axiomas de DAML + OIL, realiza una precomputación cuando se carga la ontología, y ha añadido un clasificador para realizar la jerarquía de conceptos de la ontología. Por el momento, es el único razonador que permite el lenguaje de consulta DQL específico para DAML + OIL.

Los razonadores basados en *lógica descriptiva* deben de poseer la suficiente expresividad para poder permitir que el conjunto de constructores que forman parte de los lenguajes de ontologías sean soportados, y poder permitir ambos tipos de inferencia, tanto de clasificación como de chequeo de instancias.

4.5.6 Lenguajes de reglas

Otro de los objetivos de la Web semántica es la definición de un lenguaje de reglas semánticas. Las reglas en la Web semántica pueden ser de utilidad a los razonadores como métodos para inferir sobre el conocimiento, obtener nueva información, y que además, puedan ser empleadas por los SW para especificar tareas.

Como lenguajes de reglas destacan dos, RuleML [RuleML03] y SWRL [SWRL04]. RuleML ha sido el primer lenguaje empleado para definir reglas en aplicaciones basadas en Web Semántica, aunque recientemente se ha propuesto el uso de SWRL.

RuleML (Rule Markup Language)

RuleML es una iniciativa de diferentes instituciones y grupos investigadores que tenían como objetivo definir un lenguaje de hechos y reglas semánticas que pudiese ser utilizado en los diferentes sistemas de inferencia comerciales de reglas. La primera especificación estable de este lenguaje es de 2001, y en ella se quiso dar una sintaxis para todas las formas de reglas existentes en la lógica, pero al ser demasiado extenso se decidió por proporcionar al lenguaje una sintaxis que permitiese especificar reglas de implicación y hechos.

El lenguaje que se describió debía de poseer ciertas características para poder escribir las reglas, pero la principal fue la capacidad de expresarlas. Para ello, se siguió la notación siguiente:

$$\text{Consecuente} \leftarrow \text{Antecedente}_1 \wedge \text{Antecedente}_2 \wedge \ldots\ldots \wedge \text{Antecedente}_k$$

La sintaxis del lenguaje para definir reglas fue escrita primero en una DTD (http://www.ruleml.org/indtd0.85.html), y a partir de ésta, más tarde se desarrolló la misma sintaxis pero escrita en XML Schema (http://www.ruleml.org/xsd/0.85/), gracias a que este ofrecía más facilidades que las DTD.

SWRL (Semantic Web Rule Language)

SWRL se basa en la combinación de OWL y RuleML [SWRL04]. Las características principales del lenguaje son:

- Basado en OWL DL y OWL Lite y RuleML.
- Incluye sintaxis para expresiones de cláusulas de Horn.
- Se aporta un modelo teórico-semántico para incluir reglas en ontologías OWL
- Aporta sintaxis para emplearse junto OWL-XML y OWL-RDF

La propuesta de SWRL es la de permitir definir reglas de *Horn* en una base de conocimiento OWL. Para ello extiende el conjunto de axiomas de OWL, es decir, añade los axiomas adecuados para que el lenguaje de especificación de ontologías OWL permita definir las reglas.

Las reglas propuestas para este lenguaje tienen forma de implicación, con un antecedente (*body*) y un consecuente (*head*), igual a las definidas en RuleML. El significado que se quiere dar con estas reglas es el siguiente: si las condiciones especificadas en el antecedente son ciertas, entonces las condiciones del consecuente también lo son.

SWRL se ha convertido en el lenguaje de reglas recomendado por la comunidad de Web Semántica. Por ejemplo en los lenguajes de descripción semántica de servicios, es utilizado para expresar las condiciones necesarias en la especificación de diferentes aspectos.

Actualmente se puede encontrar un editor de este lenguaje como plugin del editor de ontologías Protégé [SWRLEd05].

4.5.7 Lenguajes de consultas

RQL (RDF Query Language) [RQL] es un lenguaje de consulta para RDF y RDF Schema basado en OQL (Object Query Language). RQL nos permite navegar por los grafos que hay en el modelo RDF y proporciona un mecanismo para preguntar y seleccionar los nodos del modelo que queramos recuperar.

La característica más destacable de este lenguaje es que posee construcciones propias específicas para las relaciones semánticas dentro del RDF Schema, como pueden ser las relaciones de clase/instancia, clase/propiedad o el dominio y rango de una propiedad, por lo que resulta más fácil recuperar información de los nodos del modelo.

RDQL (RDF Data Query Language) [RDQL04] fue desarrollado por HP para que fuese el lenguaje de consulta para RDF en los modelos de Jena, con la idea de convertirse en un modelo de consulta orientado a datos por ser una aproximación más declarativa. Debido a esto, solo se pueden hacer consultas sobre la información que hay en el modelo, por lo que la inferencia o razonamiento no es posible.

RDQL deriva de SquishQL que es un lenguaje de consulta para RDF, que a la vez deriva de rdfDB, y son una clase de lenguajes para RDF, no para el RDF Schema a no ser que esté explícitamente en el modelo que se maneje.

Como RDF provee una estructura de grafos, donde los nodos son recursos o literales, RDQL permite especificar el patrón de la tripleta <sujeto, predicado, objeto> que queremos buscar en el grafo del modelo para poder recuperar cualquier parte de la tripleta, devolviendo todas las tripletas que cumplan el patrón que le pasamos (muy parecido a como actúa RQL).

Además de la desventaja de no permitir realizar ninguna inferencia, la utilización de filtros para obtener resultados es muy limitada. Las ventajas son la sencillez de manejo, ya que solo hace falta tener claro que tripleta <sujeto, predicado, objeto> se quiere preguntar, y otra de las ventajas son la facilidades de integración con Java, debido a que ha sido implementado por los mismos desarrolladores de Jena.

SeRQL (Sesame RDF Query Language) [SeRQL04] es un lenguaje de consulta desarrollado por los administradores de Sesame, combinando las características de RQL, RDQL, N-Triples y N3, añadiendo alguna más propia. Las principales características son:

- Transformación de grafo
- Soporte de RDF Schema
- Soporte de Datatype de XML Schema
- Sintaxis expresivas para los path (o URI)
- Matching opcional de path (o URI)

Este lenguaje de consultas posee tres componentes importantes: URIs, literales y variables.

Una de las diferencias de este lenguaje respecto al resto es la de presentar dos formas diferentes de hacer la consulta, la devolución de tablas con los posibles valores que pueden tomar las variables en nuestra consulta ("*Select*", común al resto de lenguajes), y devolviendo el resultado en la forma de subgrafo que nosotros le sugerimos, almacenando la información del resultado en un grafo RDF (son consultas conocidas como "*Construct*").

DQL (Daml Query Language) [Fik03] se trata de un lenguaje formal y protocolo para conducir el diálogo entre un agente cliente que realiza consultas y un agente que

contesta los requerimientos, haciendo uso para ello de conocimiento representado en DAML + OIL.

La diferencia principal de este lenguaje con el resto reside en la sintaxis para la descripción de las consultas. En este caso se deben de seguir los patrones de *Query* y *Answer* descritos por una ontología *dql.daml*. Esta ontología describe como deben de estar formadas tanto las consultas como las respuestas.

Es un lenguaje de consulta y no de inferencia, y respecto a éstos tiene la ventaja de que está preparado para manejar conocimiento descrito en DAML + OIL. Sin embargo, las herramientas que implementan este lenguaje, han sido escasas. DQL es soportado por el razonador JTP, que está preparado para admitir consultas mediante mensajes SOAP y retornar los resultados en el correspondiente mensaje de respuesta SOAP.

OWL-QL [Fik04] es una actualización del lenguaje anterior (DQL). En este caso, se trata de un lenguaje formal y protocolo para conducir el diálogo entre un agente cliente que realiza consultas y un agente que contesta los requerimientos, haciendo uso para ello de conocimiento representado en OWL.

Intenta convertirse en un estándar, y para ello tiene en cuenta una serie de factores dentro de la Web Semántica:

- La previsión de que incluirá muchos tipos de servicios petición-respuesta con acceso a muchos tipos de información representados en muchos formatos. Por lo que se debe lograr manejar esta heterogeneidad.
- El hecho de que algunos servidores contengan solo información parcial sobre algún tópico y por tanto serán incapaces de manejar o dar respuesta a ciertos tipos de requerimientos. Se pretende que los propios protocolos de consulta puedan proveer algún medio de transferencia de resultados parciales.
- La idea de que una consulta o requerimiento especificada en un lenguaje de la Web Semántica necesita dar soporte a requerimientos que no incluyan una especificación de la base(s) de conocimiento, de igual manera que los usuarios de un navegador en Internet, no necesitan describir que portales web se han de considerar cuando realizan una búsqueda. Se pretende que los propios servidores sean capaces de encontrar las apropiadas bases de conocimiento.
- El hecho de que el conjunto de notaciones y sintaxis usadas en la Web se ya demasiado extenso, y varias comunidades tengan diferentes preferencias, y por tanto ninguna universal. Se debe lograr que los aspectos esenciales del lenguaje sean independientes del conjunto de su sintaxis.
- Por último, la premisa de que en la Web Semántica, los lenguajes declarativos usados para representar en la Web tenga una semántica definida formal y lógica. Este es el caso de OWL y sus predecesores.

OWL-QL tiene en cuenta todos estos factores e intenta subsanar todas las deficiencias que hasta ahora se habían encontrado en los demás lenguajes de requerimientos.

SPARQL [SPARQL05] es un lenguaje de consultas para grafos RDF propuesto recientemente por W3C. Ofrece a los desarrolladores y usuarios finales un camino para

presentar y utilizar los resultados de búsquedas a través de una gran variedad de información como puede ser datos personales, redes sociales y metadatos sobre recursos digitales como música e imágenes. SPARQL también proporciona un camino de integración sobre recursos diferentes. Permite:

-Extraer información en diversas formas, incluyendo URIs.
-Extraer subgrafos RDF.
-Construir nuevos grafos RDF basados en la información de los grafos consultados.

Una comparativa de los lenguajes de consulta fue realizada por Aimilia Magkanaraki et al. en "*Ontology Storage and Quering*" [Mag02] y más recientemente por Peter Haase en su documento "*A Comparison of RDF Query Languages*" [Haa04].

4.6 SERVICIOS WEB

Los Servicios Web permiten a las compañías publicar componentes y servicios en un directorio en el que otras aplicaciones Web pueden buscar e implementar nuevos servicios a través de una llamada.

Las tecnologías previas al desarrollo de los SW fueron conocidas como objetos distribuidos, dentro de las que se encuentran DCOM de Microsoft, RMI de Sun, y CORBA de OMG (*Object Management Group*). Sin embargo, y a pesar de las alianzas de las compañías para conseguir interoperabilidad entre diferentes aplicaciones, y la no participación de Microsoft en el desarrollo de CORBA, ocasionó que se frenara el desarrollo de esta tecnología para dar paso a los SW, que aunque se fundamentan en nuevas características, siguen buscando la interacción entre aplicaciones.

Muchos esfuerzos se han hecho y se están realizando en la actualidad para definir las diferentes especificaciones y arquitecturas que posibiliten la expansión de este nuevo tipo de aplicaciones denominadas "Servicios Web". Grandes empresas como Microsoft [MS04], IBM [IBM04], Intel [INT04], y consorcios como W3C están profundamente involucrados en esta investigación.

En septiembre de 2000, el W3C estableció *XML Protocol Activity*. La meta del grupo es la de establecer un estándar formal para SOAP. El 23 de junio de 2003 se alcanzó la fase de recomendación oficial de W3C para SOAP 1.2. El 25 de enero de 2002, W3C también anunció la formación de *Web Service Activity*. Esta actividad incluye el actual trabajo de SOAP así como otros grupos. El grupo de *Web Services Description*, trabaja sobre WSDL. *Web Services Architecture Working Group*, intenta crear un cohesivo marco de trabajo para los protocolos de servicios de Web. Y por último *Web Services Choreography Working Group* se ocupa del desarrollo de mecanismos para coordinar la interacción entre servicios y usuarios basándose en la interoperabilidad.

Otra de las organizaciones implicadas en la evolución de este tipo de tecnología es OASIS (*Organization for the Advancement of Structured Information Standards*). Fue fundado en 1993, y tiene más de 4.000 participantes representados en 600

organizaciones y miembros individuales en 100 países. Es un consorcio global sin ánimo de lucro que controla el desarrollo, la convergencia y la adopción de estándares para la seguridad, e-business, y los esfuerzos de estandarización en el sector público y los mercados de aplicaciones específicas. Fruto de sus estandarizaciones podemos citar como ejemplo a UDDI v2 en abril de 2003. El consorcio mantiene dos de los portales de información más ampliamente considerados por las comunidades de SW y uso de XML, como son: *Cover Pages* y XML.org.

W3C aporta la siguiente definición en su grupo de investigación de SW: "*Un servicio de Web es un sistema de software diseñado para soportar la interacción entre dos máquinas a través de una red. Posee un interfaz descrito en un formato que puede ser procesado por una máquina (específicamente WSDL). Otros sistemas pueden interactuar con el Servicio Web en la manera prescrita por su descripción utilizando mensajes SOAP, típicamente transportados utilizando HTTP y serializados con XML, en conjunción con otros estándares relacionados con la Web*" [W3C04].

Otras definiciones sobre "Servicio Web" fueron dadas en [IBM04], [Gar04], [MS04].

En definitiva, tal y como afirma Vasudevan [Vas01] "un *Servicio Web* es un nuevo tipo de aplicación Web". Son aplicaciones modulares autodescriptivas que se pueden publicar, ubicar, e invocar desde cualquier punto de la Web o desde el interior de una red local basada en estándares abiertos de Internet. Los *SW* realizan funciones, las cuales van desde simples peticiones hasta procesos de negocios complicados. Una vez que un servicio Web es publicado, otra aplicación u otro Servicio Web puede localizarlo e invocarlo. Un *Servicio Web* también puede ser visto como una aplicación que existe en un ambiente distribuido, como Internet, que acepta una petición, realiza su función de acuerdo a la petición y devuelve una respuesta. Este acceso puede ser ya sea utilizando algún protocolo de paso de mensajes, algún estándar de programación distribuida, o algún formato establecido de invocación de programas en el Web como lo es la especificación de CGI. No es necesario que el proveedor y el usuario del Servicio Web tengan en cuenta los detalles de la implementación, y acceso al servicio, ya que se basan en estándares como XML, HTTP y SMTP y siguen una estructura común para su descripción e invocación.

Un aspecto importante en los SW es su capacidad de agregación o composición con otros SW para proveer un conjunto de mayores características. Por ejemplo, la compra de un libro por Internet se puede componer de varios servicios: su búsqueda y posteriormente dada su disponibilidad, su compra y algún servicio adicional que muestre otros libros relacionados con el buscado y que pudieran interesar al comprador.

4.6.1. Arquitectura de los Servicios Web.

Los SW son componentes software independientes de la plataforma hardware o software sobre la que está implementado, y pueden ser:

- Descritos mediante un lenguaje de descripción de servicio, como el lenguaje WSDL (*Web Service Description Language*)

- Publicados al someter las descripciones y políticas de uso en algún registro bien conocido, utilizando el método de registro UDDI (*Universal Description, Discovery and Integration*) u otro método.
- Encontrados al enviar peticiones al registro y recibir detalles de enlace (*binding*) del servicio que se ajusta a los parámetros de la búsqueda.
- Asociados al utilizar la información contenida en la descripción del servicio para crear una instancia de servicio disponible o proxy.
- Invocados sobre la red de forma local o remota al utilizar la información contenida en la descripción del servicio o detalles de *service binding* (vinculación).
- Compuestos con otros servicios para integrar servicios y aplicaciones nuevas.

El grupo de trabajo *W3C Web Services Architecture Working Group* en [WSA04] identifica los componentes funcionales y define las relaciones entre ellos para lograr las propiedades deseadas de una arquitectura global. Establece un meta-modelo de la arquitectura compuesto por cuatro modelos que se caracterizan por un aspecto particular entorno al cual se constituyen: Modelo Orientado a Servicio, Modelo Orientado a Mensajes, Modelo Orientado a Recurso y Modelo de Políticas. Por ejemplo, el Modelo Orientado a Servicio tiene su enfoque en los aspectos del servicio y acciones posibles, y se supone que es el más complejo de todos los modelos en la arquitectura. En este modelo se describen conceptos como agente proveedor, agente consumidor, servicio, semántica del servicio, tarea del servicio etc., estableciendo las relaciones entre todos los conceptos a través de predicados.

La arquitectura de servicios de Web se puede estudiar examinando sus componentes y examinando la arquitectura de capas de los protocolos [Cer02], [UWO04]. Examinando los componentes de un Servicio Web podemos encontrar los siguientes elementos:

- Servicio. La aplicación es ofrecida para ser utilizada por solicitantes que completan los requisitos especificados por el proveedor de servicios. La implementación se realiza sobre una plataforma accesible en la red. El servicio se describe a través de un lenguaje de descripción de servicio. Tanto la descripción como las políticas de uso han sido publicadas de antemano en un registro.
- Proveedor de Servicio. Desde el punto de vista comercial, es quien presta el servicio. Desde el punto de vista de arquitectura, es la plataforma que provee el servicio.
- Registro de Servicios. Es un depósito de descripciones de servicios que puede ser consultado, donde los proveedores de servicios publican sus servicios y los solicitantes encuentran los servicios y detalles para utilizar dichos servicios.
- Solicitante de servicios. Desde el punto de vista comercial, la empresa que requiere cierto servicio. Desde el punto de vista de la arquitectura, la aplicación o cliente que busca e invoca un servicio.

La figura 4.14 muestra las relaciones operacionales entre los componentes:

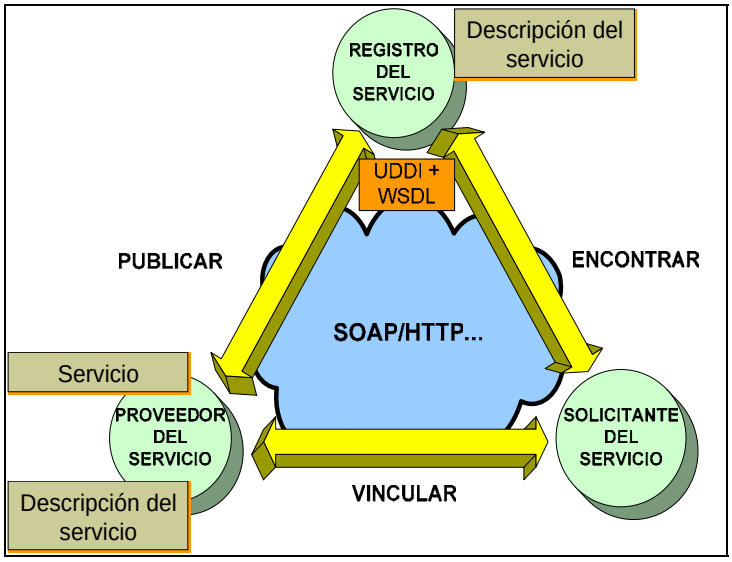


Figura 4.14 Arquitectura Orientada a Servicios (SOA) [WSA04]

En la arquitectura también puede aparecer otro componente denominado en algunos medios "broker", el cual provee al solicitante de servicios de un mecanismo de descubrimiento para encontrar el servicio requerido y al mismo tiempo, también provee a los proveedores de servicios de un medio de publicitar y gestionar las descripciones sobre los servicios ofertados.

Otra forma de ver la arquitectura de un servicio Web es examinando la pila de protocolos. Esta pila está todavía evolucionando, pero actualmente posee cinco capas principales. El diagrama muestra esta pila de tecnologías y el actual estándar elegido, correspondiente para cada una de estas capas (figura 4.15):

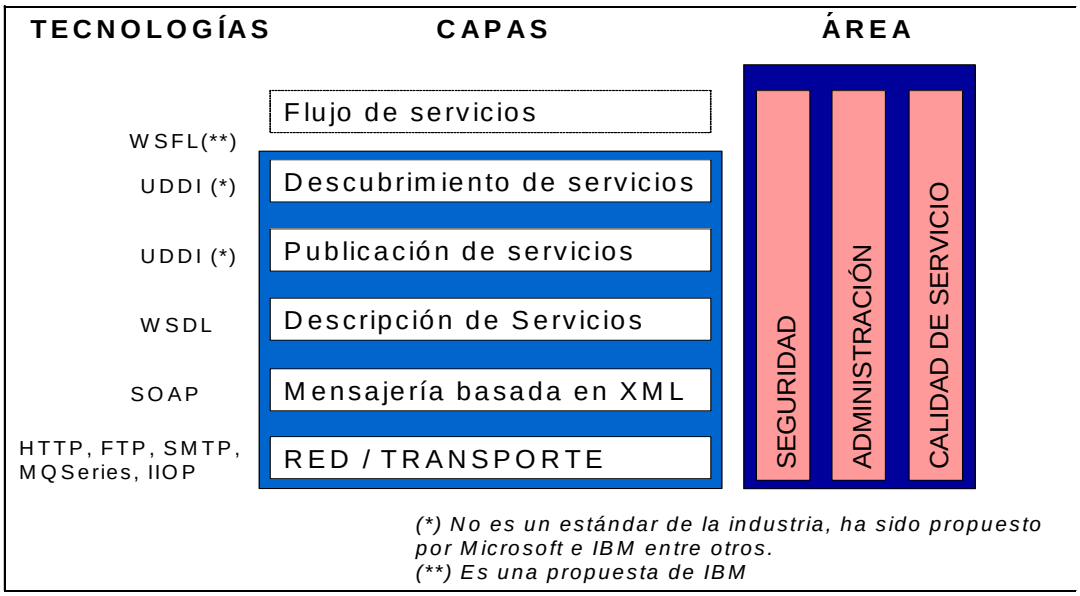


Figura 4.15 Arquitectura de capas en Servicios Web [Mye02]

4.6.2. Tecnologías y Estándares de Servicios Web

Los SW se registran y anuncian utilizando los siguientes lenguajes y protocolos. Mucho de estos estándares y otros están siendo desarrollados en el proyecto UDDI [UDDI04], un consorcio de industrias que coordina los esfuerzos de diseño y creación. Estas tecnologías encapsulan un conjunto de estándares que permite a los desarrolladores implementar aplicaciones distribuidas.

XML (*eXtensible Markup Language*).

Nació en febrero de 1998 y ha revolucionado la forma en que estructuramos, describimos e intercambiamos información. El diseño de XML se deriva de dos fuentes principales: SGML (*Standard Generalized Markup Language*) y de HTML (*HyperText Markup Language*). Independientemente de las múltiples formas en que se utiliza hoy en día el XML, todas las tecnologías de SW se basan en él. A continuación se citan las tecnologías de SW más importantes.

UDDI (*Universal Description, Discovery and Integration*).

Fruto de las estandarizaciones de OASIS podemos citar a UDDI v2 en abril de 2003. Es un protocolo para describir los componentes disponibles de SW. Este estándar permite a las empresas registrarse en un tipo de directorio de páginas amarillas de Internet que les ayuda anunciar sus servicios, de tal forma que las compañías se puedan encontrar unas a otras y realizar transacciones en la Web. El proceso de registro y consultas se realiza utilizando mecanismos basados en XML y HTTP(S). En el proyecto UDDI se trabaja para proveer un método de acceso común a los metadatos necesarios para determinar si un elemento de código previamente elaborado es suficiente, y si lo es, cómo acceder a él.

Un registro UDDI se puede comparar con un buscador en Internet, puesto que el buscador contiene información indexada y categorizada sobre las páginas Web. A diferencia del buscador, que únicamente devuelve un URL de una página Web, un registro UDDI necesita ofrecer no solo la localización de los servicios, sino también información sobre el servicio, cómo funciona, qué parámetros utilizar, qué valores devuelve, etc. Más específicamente, UDDI incluye tres tipos de información registrada: Páginas Blancas (empresas organizadas por nombre), Páginas Amarillas (empresas categorizadas por sector), y Páginas Verdes (empresas organizadas por servicio).

SOAP (*Simple Object Access Protocol*)

En mayo de 2000, el W3C reconoció la propuesta de SOAP presentada por diversas empresas de forma conjunta. El 23 de junio de 2003 se alcanzó la fase de recomendación oficial de W3C para SOAP 1.2 [SOAP03]. El SOAP simplifica el acceso a los objetos, permitiendo a las aplicaciones invocar métodos objeto o funciones, que residen en sistemas remotos. Se basa en XML y describe un formato de mensajería para la comunicación entre equipos. Contiene también varias secciones opcionales que describen las llamadas a métodos (RPC) y proporcionan detalles sobre el envío de mensajes SOAP a través de HTTP.

Los objetivos primordiales de SOAP, son:

- Establecer un protocolo estándar de invocación de servicios remotos, basado en protocolos estándares de Internet: HTTP para la transmisión y XML para la codificación de datos.

- Independencia de plataforma, lenguaje de desarrollo e implementación (modelo de objetos).

Un mensaje SOAP se compone de tres partes principales: un sobre, un encabezado y un cuerpo. El sobre indica que el código XML es un mensaje SOAP estándar. El encabezado contiene información de carácter opcional sobre el mensaje (datos para la autenticación, por ejemplo,). Por último, el cuerpo contiene información relativa al componente al que la aplicación está efectuando la llamada (es decir, los parámetros de entrada y salida esperados y los tipos de datos de dichos parámetros).

WSDL (*Web Service Description Language*).

Es el estándar propuesto para la descripción de los SW, el cual consiste en un lenguaje de definición de interfaz (IDL - Interface Definition Language) de servicio basado en XML, que define la interfaz de servicio y sus características de implementación. El WSDL es apuntado en los registros UDDI y describe los mensajes SOAP que definen un servicio Web en particular. WSDL fue propuesto por Ariba, IBM, y Microsoft para describir SW en el ámbito de protocolos de los grupos de trabajo del WWW Consortium W3C-XML y es ya una recomendación oficial. La versión 1.0 de WSDL data de septiembre de 2000. Actualmente el grupo de trabajo de W3C está desarrollando la versión 2.0 [WSDL04].

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.

Es importante observar que WSDL no introduce un nuevo lenguaje de definición de tipos. WSDL reconoce la necesidad de disponer de diferentes sistemas de tipos para describir los formatos de mensaje y admite como sistema de tipos canónico la especificación de los esquemas XML. Sin embargo, puesto que no es razonable esperar una única gramática del sistema de tipos que se utilice para describir todos los formatos de mensajes presentes y futuros, WSDL permite el uso de otros lenguajes de definición de tipos mediante la extensibilidad.

WSCI (*Web Service Coreography*[15] *Interface*)

WSCI [WSCI04] es una iniciativa (no estándar) lanzada por BEA, INTALIO, SAP, y SUN para complementar a WSDL.

WSCI complementa a WSDL indicando el comportamiento requerido o la lógica de un servicio más completo. Permite expresar las distintas operaciones que deben ejecutarse para realizar un servicio complejo, y las relaciones/condiciones entre ellas.

- No extiende ni cambia a WSDL.
- No compite con WSDL, sino que requiere de él y lo complementa.

[15] **Coreography** es un modelo de la secuencia de operaciones, estados, y condiciones que controlan las interacciones involucradas en los servicios participantes [WSA04].

- WSCI hace que WSDL sea más utilizable, ya que WDSL únicamente permite describir servicios más simples y sin estados.
- WSCI permite formar servicios completos, mediante descripciones de servicios WSDL.

Haciendo uso de WSDL tendríamos la visión WSDL (estática) del servicio y sus operaciones, pero no tenemos idea del orden de las operaciones, ni de las relaciones de dependencia entre ellas. Esto es lo que permite completar WSCI.
El cliente puede así observar el orden lógico de las operaciones que tiene que realizar para por ejemplo, planificar un viaje completo, como una secuencia de operaciones WSDL. WSCI también permite tomar decisiones en función del resultado de una operación (condiciones IF), realizar bucles, etc.

WSFL (*Web Services Flow Language*)

Se trata de un lenguaje creado por IBM y diseñado exclusivamente para representar flujos de SW y de este modo facilitar la creación de orquestaciones[16] de estos servicios. Se trata de un lenguaje muy sencillo y basado en XML.

Dentro de WSFL sin duda los conceptos más importantes son los de actividad y proveedor de servicios. Las actividades son las tareas a realizar mientras que los proveedores de servicios son las personas/roles encargadas da realizar dichas actividades. La versión 1.0 se puede encontrar en [WSFL04].

4.7 CONCEPTO DE SERVICIO WEB SEMÁNTICO

La Web no solamente proporciona acceso a contenidos sino que también ofrece interacción y servicios (comprar un libro, reservar una plaza en un vuelo, hacer una transferencia bancaria, simular una hipoteca). Los SWS son una línea importante de la Web Semántica, que propone describir no solo información sino definir ontologías de funcionalidad y procedimientos para describir SW: sus entradas y salidas, las condiciones necesarias para que se puedan ejecutar, los efectos que producen, o los pasos a seguir cuando se trata de un servicio compuesto. Estas descripciones procesables por máquinas permitirían automatizar el descubrimiento, la composición, y la ejecución de servicios, así como la comunicación entre unos y otros.

Los SWS proponen extender las tecnologías de SW tradicionales, en vías de consolidación, con ontologías y semántica que permitan la selección, integración e invocación dinámica de servicios, dotándoles así mismo de la capacidad de reconfigurarse dinámicamente para adaptarse a los cambios (p.e. interrupción de servicios o aparición de otros más adecuados) sin intervención humana. En la figura 4.16 se puede observar la evolución de la Web y de los SW.

[16] Una **orquestación** define la secuencia y condiciones en las que un Servicio Web invoca otros Servicios Web, con el objetivo de realizar alguna función de utilidad. [WSA04]

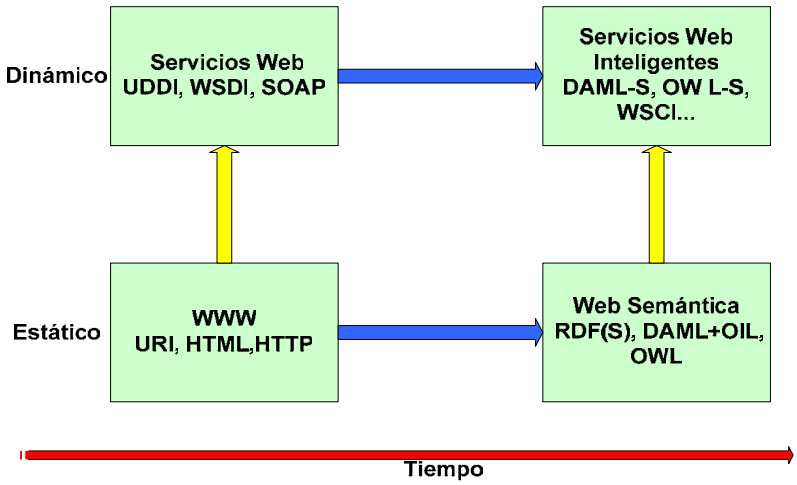


Figura 4.16 Evolución de WWW y Servicios Web. [Bou05]

Los SWS son un nuevo paradigma de la computación, definido generalmente como el surgimiento de las descripciones de SW con anotaciones de Web Semántica, para facilitar la automatización del descubrimiento, composición, invocación, y monitorización de los servicios en un ambiente no regulado y caótico como es la Web [Pay04].

Las interrelaciones entre los componentes involucrados en el desarrollo de SWS pueden ser vistas en la figura 4.17:

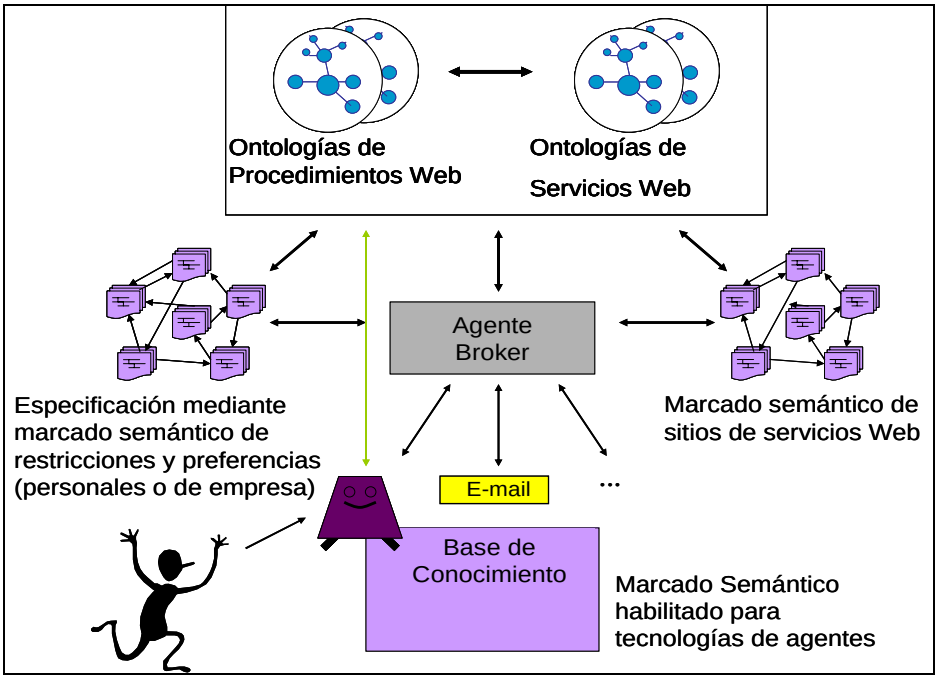


Figura 4.17 Un marco de trabajo para Servicios Web Semánticos. [McI01]

4.7.1 Tareas asociadas a una ontología de descripción de servicio

Las tareas que debe permitir el marcado semántico de SW ([McI01] [DAMLS03] [OWLS03]) son:

- ***Descubrimiento automático del servicio***:

Está relacionado con la localización automática de los SW que provee un servicio particular y que se adhiere a las restricciones dadas en la petición.

Por ejemplo, supongamos que se requiere "encontrar un servicio de suscripción de alertas sobre información de incidencias en una determinada carretera". Esta tarea hoy en día es desarrollada por un humano quien deberá realizar los siguientes pasos:

a) Posibilidad de usar un motor de búsqueda para encontrar dicho servicio.
b) Lectura del sitio web encontrado.
c) Ejecución del servicio manualmente (introducción de franja horaria, nomenclatura de la carretera etc.)

Con el marcado semántico de servicios, es posible especificar la información necesaria para el descubrimiento del servicio como marcado semántico interpretable por computadores, de manera que un servicio de registro o motor de búsqueda (mejorado con ontologías) pueda localizar automáticamente los servicios apropiados.

- ***Ejecución automática del servicio Web***:

Involucra la ejecución automática de un servicio Web identificado por un programa de ordenador o agente. La ontología de descripción del servicio debería proveer APIs declarativos para SW, necesarios par una correcta ejecución automática de éstos. Siguiendo con el ejemplo anteriormente expuesto, supongamos que quisiéramos finalmente suscribirnos al servicio de alertas para recibir información actualizada en la franja horaria y carretera establecida. Posiblemente en este caso los pasos sean los siguientes:

a) Acceder al sitio web, rellenar el formulario de suscripción y presionar sobre la opción de aceptar para ejecutar el servicio.
b) Alternativamente, enviar una petición "HTTP" directamente al servicio mediante la dirección URL con los parámetros apropiados.

- ***Composición e interoperación del servicio Web***:

Equivale a la selección automática, composición e interoperación de SW para realizar alguna tarea, dada una descripción detallada de un objetivo. Se denomina composición del SW a la técnica de componer funcionalidades de servicios relativamente más simples para producir una aplicación compleja y significante de forma arbitraria.

La ontología de descripción de servicio debe proveer especificaciones declarativas de los prerrequisitos y consecuencias del uso individual de un servicio, necesarios para la composición e interoperación automática. Imaginemos que el usuario

quisiera planificar un viaje por carretera, en una fecha concreta. Las tareas a desempeñar podrían ser las siguientes:

a) Especificar la composición manualmente.
b) Asegurar que cualquier software para la interoperación está hecho a la medida.
c) Proveer la entrada en los puntos de elección (Por ejemplo seleccionar todas aquellas carreteras por las que se prevé pasar).

- Ejecución y monitorización [17]

Esta tarea involucra la habilidad de los usuarios / agentes para conocer el estado de su petición durante el periodo de ejecución o cualquier violación de las restricciones motivada por las acciones llevadas a cabo por los agentes software.

En este caso la ontología de descripción de servicio debe proveer descriptores para la ejecución de servicios. En el ejemplo, podríamos querer saber el estado de nuestra suscripción o cualquier detalle adicional.

Con el marcado semántico de SW, la información necesaria para seleccionar, componer, y responder a servicios es codificada en los sitios del servicio. De esta forma es posible crear software para manipular este marcado, junto con los objetivos de la tarea, para conseguir la automatización de este proceso.

4.7.2 Evolución tecnológica hacia Servicios Web Semánticos

A partir de los esfuerzos de estandarización del W3C y a través de su grupo de interés *Semantic Web Services Interest Group* [SWSIG05] se está fomentando la integración de la tecnología de la Web Semántica y el trabajo de otros grupos sobre SW desarrollado en W3C. Una de las iniciativas más importantes surgidas a partir de este trabajo a tomado como base el trabajo desarrollado por la coalición DAML-S (DAML for Services), surgida en el año 2001, la cual propuso una ontología para la descripción semántica de servicios basada en DAML+OIL (DAML-S), cuya última versión (0.9) fue lanzada en mayo de 2003. A partir de noviembre de 2003 se desarrolló OWL-S, basado en el lenguaje de marcado ontológico OWL. Actualmente esta iniciativa es el resultado de la unión de otras dos: *Semantic Web Services Initiative* (SWSI) [SWSI05] y *Semantic Web Services Language (SWSL) Committee* [SWSL05]. OWL-S fue enviado a W3C para su reconocimiento en julio de 2004 y en el momento de escribir estas líneas, todavía no supone una recomendación.

Existen diversas tecnologías involucradas en la evolución de SW a SWS. En la figura 4.18 se puede ver esta evolución, así como las principales fechas que recogen algún momento importante, como el lanzamiento de una determinada tecnología, el envío a W3C o finalmente la recomendación o estandarización por parte de este consorcio. Sin embargo, hay que aclarar que las fechas que aparecen en la figura no siguen un orden

[17] Esta tarea no aparece en [McI01b] y en la versión OWL-S 1.1 Release [OWLS04] fue eliminada de la documentación pertinente, debido principalmente a que el foco de la investigación hasta la fecha habían sido las otras tres tareas.

cronológico en esta evolución, ya que a veces, las recomendaciones son posteriores al uso de estas tecnologías o simplemente las fechas reflejan el lanzamiento de versiones posteriores. Es importante, destacar el continuo y dinámico trabajo llevado a cabo en cada uno de las áreas, lo que demuestra el gran esfuerzo que se lleva realizando así como el gran interés suscitado.

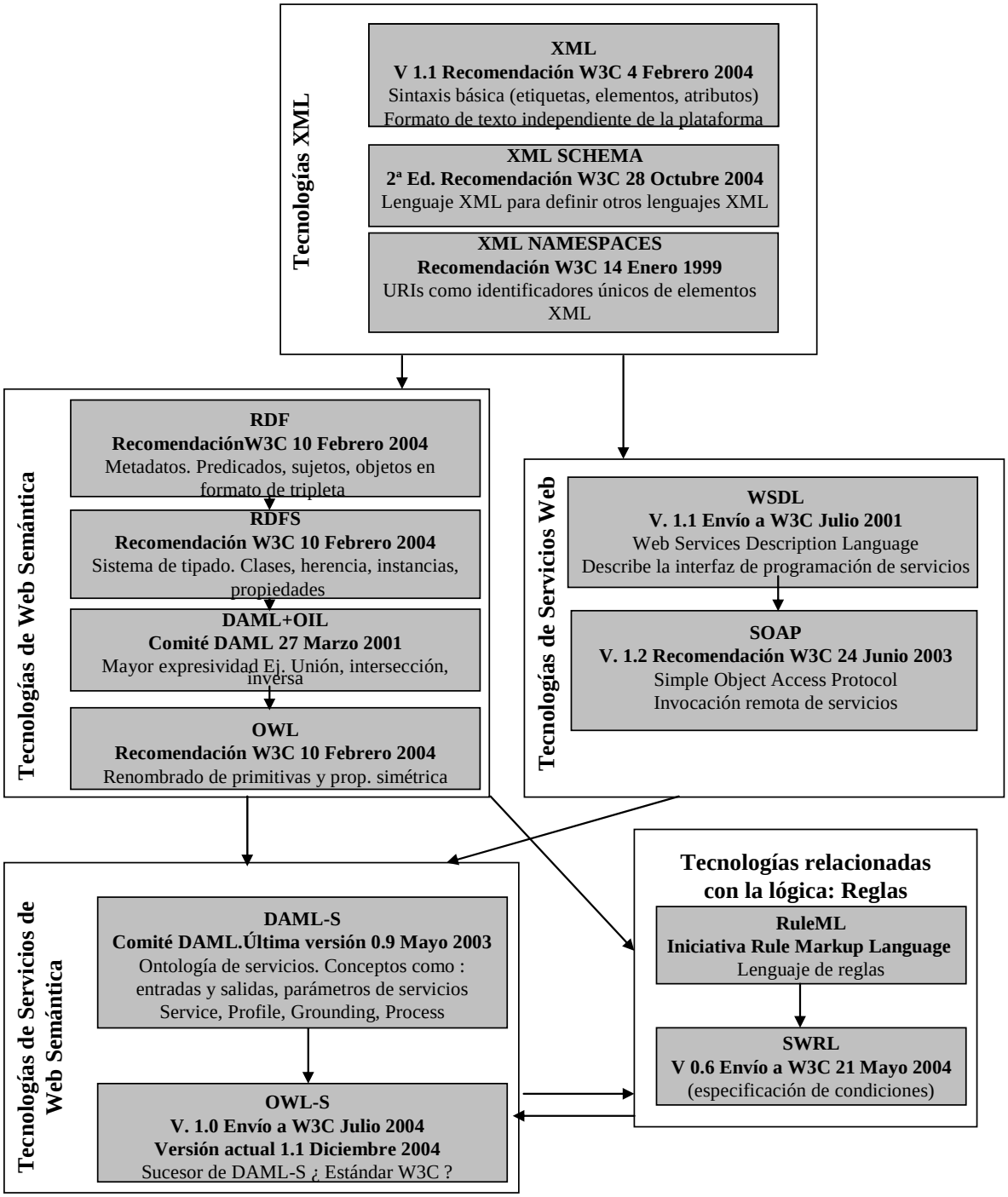


Figura 4.18 Tecnologías involucradas en Servicios Web Semánticos.

Mientras que lenguajes como WSDL son los encargados de aspectos de comunicación de bajo nivel para los servicios de la Web, DAML-S fue desarrollado para proporcionar una capa de aplicación de alto nivel por encima de éstos. Como estos lenguajes de comunicación de bajo nivel y protocolos definen *cómo* tener acceso a un servicio de Web, tanto DAML-S como su predecesor OWL-S tienen como objetivo el proporcionar una respuesta en cuanto a *porqué* se utiliza un cierto servicio, y lo que este servicio hace realmente.

Castillo, Trastour et al. [Tras01] y [Gon01], presentan una serie de requerimientos en la descripción de un servicio mediante el uso de RDF(S) fácilmente extrapolables al uso de otros lenguajes semánticos como DAML + OIL u OWL:

- Alto grado de flexibilidad y expresividad.
- Soporte de tipos y subsumción.
- Soporte de Datatypes.
- Facilidad para poder expresar restricciones.
- Necesidad de compartir la misma semántica.
- Sintaxis apropiada para la Web.

En [Kle03], además del requerimiento de expresividad semántica, añaden también el hecho de que las descripciones de servicios permitan realizar comparaciones de manera automática, y por tanto que los servicios tengan una cierta estructura para lograr este objetivo, además de los requerimientos de flexibilidad y edición. La búsqueda de servicios mediante descripciones dadas por estos lenguajes, a veces no se puede llevar a cabo mediante plantillas fijas o comunes y por tanto el lenguaje debe ser lo suficientemente flexible como para permitir tal expresividad. La conveniencia de que las descripciones de servicio deban ser creadas, leídas y editadas por humanos establece el último de los requerimientos. En [Pil03] se establecen los requerimientos para el descubrimiento de un servicio Web, para cada una de las partes que forman parte del proceso: descripción del servicio, publicación, descripción de la petición, y por último el servicio de emparejamiento.

Otro aspecto importante argumentado anteriormente ya por Trastour et al. en [Tras01] al describir diferentes casos de uso, es la distinción entre peticiones de servicio volátiles y persistentes. Las primeras serían aquéllas en las que el *matchmaker* devuelve en el instante del requerimiento aquellos anuncios compatibles con la petición, mientras que las segundas se corresponden con peticiones que pueden ser respondidas en un tiempo posterior y dentro de un periodo de validez definido previamente, tras la incorporación de nuevos anuncios o modificación de los ya existentes. Por último afirman la necesidad de dar soporte a la composición de servicios, debido a que posiblemente determinados servicios no puedan satisfacer una determinada petición por si solos, pero sin embargo una combinación de éstos sí.

4.7.3 Arquitecturas para diseño conceptual de SWS

4.7.3.1. Caracterización conceptual del servicio

En [Gom04b] se han propuesto los tipos de características que permiten a los agentes o programas externos descubrir, invocar y componer SWS, así:

- Acceso (o comunicación): Está relacionada con el protocolo de comunicación requerido para invocar la ejecución del servicio (ejemplo SOAP o HTTP).

- Descriptivas: Detalla propiedades de los SWS tales como ubicación geográfica, clasificación comercial (ejemplo UNSPSC) o proveedor. Esas características definen el dominio (tal como minerales en UNSPSC) en el cual es desarrollada la operación del servicio. También pueden guiar el descubrimiento del servicio rechazando los servicios que operan en un dominio diferente.

- Funcionales: Especifican las capacidades de los SWS (sus datos de entrada y salida), efectos, pre y postcondiciones de ejecución. Una vez el dominio del servicio ha sido establecido, esas características permiten definir a un agente externo cuando la ejecución del servicio puede obtener el resultado solicitado.

- Estructurales: Describen la estructura interna del servicio, es decir, los componentes estructurales y cómo se combinan para ejecutar el servicio. Típicamente, los agentes usan esas características para composición del servicio debido a que ellas determinan si existen interacciones entre subservicios y otros servicios usados para componer un nuevo servicio.

4.7.3.2 Propuestas existentes

En el desarrollo de SWS se requieren arquitecturas conceptuales a nivel de conocimiento y en forma independiente del lenguaje. Estas arquitecturas deben permitir especificar un conjunto de capas que cubran las características del servicio.

Las estructuras internas de los SW y de los SWS han sido modeladas tradicionalmente como procesos de negocio, teniendo en cuenta las actividades que se deben realizar para ejecutar el proceso. Este enfoque rompe el proceso en actividades cuyas interacciones pueden ser modeladas como patrones de flujo [Gom04b].

Otra propuesta para el desarrollo de arquitecturas conceptuales existentes es ODE SWS [ODESWS05] que utiliza modelado basado en métodos de solución de problemas (PSM: *Problem Solving Methods*) para la descripción de la estructura de los servicios.

En [Nar03] se plantea la traducción de la semántica detrás de OWL-S en lógica de primer orden, obteniendo una serie de axiomas para describir las características de los servicios. Por otra parte, utilizan Redes de Petri para expresar la semántica operacional

distribuida para las ontologías DAML-S y OWL-S, permitiendo razonar a cerca de la interacción en el proceso de composición de la estructura de un servicio.
Posiblemente un aspecto que dificulta notablemente la existencia de propuestas concretas de arquitecturas se basa en la complejidad de integración tecnológica (agentes inteligentes, SW, sistemas de emparejamiento) y la diversidad de propuestas existentes que componen los SWS.

4.7.4 Ontologías para la descripción de servicios Web: DAML-S (OWL-S)

Para poder hacer uso de un servicio Web, un agente software necesita una descripción del servicio que sea interpretable por un computador, y el medio por el cual es accedido. El uso de DAML+OIL o posteriormente OWL proporciona el marco de trabajo apropiado para poder desarrollar esto. A continuación se describirá este tipo de ontologías basándonos en la especificación en el primero de los lenguajes [Ank02], [DAMLS05].

Un servicio descrito con DAML-S permite a un agente software localizar, invocar, interoperar y monitorizar dicho servicio. DAML-S proporciona respuesta a las principales preguntas que cabe plantearse a la hora de describir un servicio Web:

- ¿Qué es lo que el servicio requiere de los usuarios, u otros agentes y que les proporciona?
- ¿Cómo funciona el servicio?
- ¿Cómo se puede utilizar el servicio?

Atendiendo a estas cuestiones anteriormente planteadas se hace necesario el desarrollo de una ontología de orden superior de servicios en la que aparecen determinados conceptos que intentan responder a estas preguntas.

La ontología de servicios fue estructurada considerando las funcionalidades arriba descritas. Esto dio lugar a tres subontologías reunidas en una cuarta que sirve para identificar el servicio como un concepto *Service* y que permite relacionar los tres tipos de conocimientos del servicio descritos por las otras tres.

La clase *Service* es el punto de referencia para poder anunciar un servicio Web. Esta clase *Service* tienen tres propiedades *presents*, *describedby*, *supports* donde los rangos de estas propiedades son las tres clases asociadas para representar cada parte de un servicio: *ServiceProfile*, *ServiceModel*, *ServiceGrounding* respectivamente. Cada vez que queramos anunciar un servicio, deberemos de instanciar cada una de estas clases.

- La clase *Service* es presentada (*presents*) por *Service Profile*, que nos sirve para definir qué es lo que el servicio necesita del usuario o agente.
- La clase *Service* es descrita (*describedby*) por *Service Model*, en el que explica cómo trabaja el servicio (cómo debe usar los inputs, el flujo de datos en un servicio compuesto etc.).

- La clase *Service* es soportada (*supports*) por *Service Grounding*, nos indica cómo debe ser invocado el servicio.

Service Profile responde a la primera de las preguntas, es decir a "qué es lo que el servicio requiere de los usuarios, u otros agentes y qué cosas proporciona el servicio a ellos". Service Profile provee una forma de describir los servicios ofrecidos por los proveedores y los servicios necesitados por los solicitantes. Esencialmente, es una pequeña descripción de lo que hace el servicio, describiendo el servicio como una función de tres tipos básicos de información:

- especificación comprensible para un humano del servicio, que contiene información sobre la organización que provee el servicio y participantes,
- especificación de las **funcionalidades** que proporciona el servicio en términos de parámetros (entradas, salidas, precondiciones y efectos).
- conjunto de **atributos no funcionales** que proveen información adicional sobre las necesidades del servicio (*QoS*, región en la que se aplica el servicio etc.).

Service Model responde a la segunda de las cuestiones anteriores, es decir, a "cómo funciona el servicio". Describe que sucede cuando un servicio se lleva a cabo. El SW que queramos modelar será un ejecutable o un CGI o cualquier aplicación que sea accesible por web. La perspectiva del SW se vio desde DAML-S como un proceso, así pués el Service Model se especializa en un Modelo de Proceso integrado por dos componentes:

- Process Model: Describe el servicio en términos de sus componentes o subprocesos. El servicio es descrito en términos de sus entradas, salidas, precondiciones y efectos.
- Control Model: permite al agente monitorizar la ejecución del servicio seleccionado.

El ***Service Grounding*** de un servicio sirve para especificar como acceder al servicio (formato de mensajes, protocolo de interacción, direccionamiento y transporte). Se puede ver como una correspondencia o mapeo entre la especificación abstracta y la concreta, de aquellos elementos de descripción del servicio que son requeridos para la interacción con el servicio (entradas y salidas de los procesos atómicos[18]).

La figura 4.19 representa la ontología de servicios asociada:

[18] Proceso atómico es áquel que es directamente invocable (previo paso de mensajes de forma apropiada), no tienen subprocesos y se ejecutan en un solo paso, desde la perspectiva del cliente. Es decir, toman un mensaje de entrada, se ejecutan y devuelven un mensaje de salida.

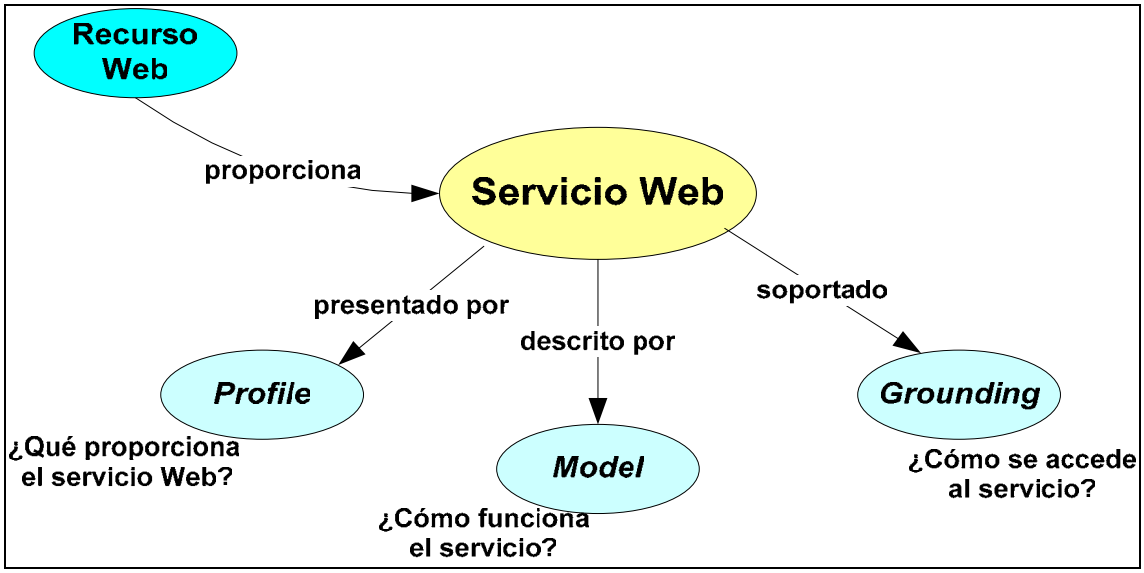


Figura 4.19 Ontología de servicios Web DAML-S [Mar02b]

Esta ontología DAML-S presenta dos restricciones de cardinalidad en sus propiedades:

- Un SW puede ser descrito como mucho por un *ServiceModel*
- Un *ServiceModel* debe estar asociado por al menos un *ServiceGrounding*

En principio para describir un servicio Web se necesitarán las tres propiedades para hacerlo correctamente, pero no se indica ninguna restricción respecto una cardinalidad mínima para las propiedades *presents*, *described by*, ni máxima cardinalidad para las propiedades *presents*, *supports* (un servicio podrá ser representado de diferentes formas, así como ser usado o invocado de diferentes formas si se quiere).

Una exposición detallada de las ontologías de descripción de servicios, puede ser encontrada en el documento denominado "**Ontologías necesarias para la descripción de servicios Web"** en la URL http://robotica.uv.es/tesis/javi/document.html y en el CD-ROM que acompaña esta memoria.

4.7.4.1 Diferencias entre DAML-S 0.9 y OWL-S 1.0

La versión 1.0 ofrece un número de refinamientos en el Service Profile y en el Process Model. Las mejoras sobre DAML-S 0.9 incluyen: clarificación y simplificación de los parámetros de descripción de la capacidad (es decir, entradas, salidas, condiciones previas y efectos), una integración más ajustada con el modelo de proceso (Process Model), y de una mejor organización / modularidad de los constructores del perfil (Profile). Los constructores en el Process son ahora los mismos que en el Profile. [Mar04].

Existe una diferencia en la definición de un proceso según se emplee la ontología de DAML-S versión 0.9 u OWL-S 1.0. En la versión 0.9, los procesos de un servicio Web son entendidos como subclases de la clase *atomic* o *composite*, mientras que en

OWL-S 1.0, se describe a los procesos como instancias de dichas clases. Este cambio de mentalidad, tratar ahora los procesos como instancias, surgió de diferentes reuniones del comité de DAML-S y discutidas por los propios desarrolladores y miembros de DAML-S como David Martin en el foro de W3C sobre SW [Mar03b]. En la comunicación se muestran las ventajas que tiene representar los procesos como instancias o como clases. La ventaja de los procesos como instancias se resume en que permite ser más concreto en la descripción, facilidad para leer y entender un proceso. Parece más intuitivo entender los procesos como instancias, además de que al definir procesos como clases se deben usar constructores difíciles de ser soportados; mientras que las ventajas de tener los procesos como clases son la elegancia con la que quedan representadas, una sensación de mejor aprovechamiento del razonamiento con DL y la posibilidad de tener instancias como trazas de ejecuciones del servicio.

A su vez en OWL-S 1.0 se plantean la necesidad de representar las condiciones[19] mediante dos lenguajes de reglas: SWRL, bajo desarrollo en W3C y DRS [McD04], descrito por Drew McDermott en un apéndice de OWL-S 1.0 Release. Sin embargo dejan en manos del modelador la tarea de decidir sobre la adopción de uno u otro.

Se pueden observar las diferentes especificaciones para los SWS de tráfico creados en la URL http://robotica.uv.es/tesis/javi/sws.html y en el CD-ROM que acompaña esta memoria.

4.7.5 Limitaciones de los estándares

Los estándares tales como SOAP y WSDL han sido diseñados para proveer descripciones de mecanismos de transporte mediante mensajes y para describir la interfaz usada por cada servicio. Sin embargo, ni SOAP ni WSDL sirven de ayuda para la localización automática de los SW en base a sus capacidades. Con WSDL, las descripciones de los servicios no son tan expresivas como los perfiles de servicios expresados mediante DAML-S u OWL-S profiles (precondiciones, postcondiciones y efectos no pueden ser expresados con WDSL). No soporta las descripciones semánticas de los servicios. Al carecer de una semántica explícita, dos descripciones idénticas pueden convertirse en diferentes dependiendo del contexto de su uso. Esto supone claramente una gran desventaja en cuanto a la búsqueda y descubrimiento de servicios, puesto que el cliente no conocerá que servicios son provistos en un determinado instante, y además tanto proveedores como clientes tendrán diferentes perspectivas y diferente conocimiento sobre el mismo servicio [Pao02a]. Como queda patente, WSDL y DAML-S / OWL-S describen los servicios de forma diferente. Mientras que WSDL describe los aspectos físicos (operaciones que el servicio soporta, estructura de los mensajes intercambiados para cada operación, codificación y destino del mensaje), DAML-S / OWL-S describe los aspectos semánticos (clasifica operaciones, mensajes y tipos), aunque los dos son complementarios.

Por otra parte, UDDI provee un registro de servicios y describe los negocios por sus atributos físicos como el nombre, dirección, y los servicios que proveen. Además, las descripciones de UDDI son complementadas por un conjunto de atributos, llamados *Tmodels*, los cuales describen características adicionales tales como la clasificación

[19] La condiciones son usadas tanto para describir salidas y efectos que resultan de la ejecución de los procesos como para la especificación de constructores tales como sentencias-if y bucles.

taxonómica de servicios tal como NAICS. Pero UDDI no representa las capacidades de servicios, y por tanto no ayuda a la búsqueda de servicios en base a lo que éstos proveen sino al uso de *keywords* o palabras clave. UDDI no da soporte a las descripciones semánticas de servicios ni provee o especifica lenguajes de contenido para anunciar servicios (hasta la fecha). Por consiguiente, la búsqueda de un servicio Web con una funcionalidad dada se convierte en una tarea muy difícil.
Hay que añadir además que hoy en día UDDI carece de moderación en los registros existentes (muchas de las entradas no son válidas), y además poseen una inadecuada garantía de calidad de servicio (*QoS*) en los SW (confiabilidad del servicio encontrado) [Hor02].

4.7.6 Concepto de emparejamiento semántico

El sistema de localización o de descubrimiento de servicios, debe estar basado en un emparejamiento semántico de las capacidades y descripciones de los SW. El concepto de ***emparejamiento semántico*** consiste en la búsqueda de dos descripciones de servicios con propiedades similares o exactamente las mismas. Este tipo de emparejamiento tiene a su favor respecto al emparejamiento implementado por sistemas basado en XML la independencia de contexto.

En la tarea de buscar un servicio individualmente o buscar un servicio que forma parte de un proceso de la Web es importante que las entradas y las salidas del servicio del candidato emparejen a las exigidas por el consumidor. La aproximación más común es utilizar ontologías para describir el servicio requerido. De acuerdo con esta aproximación, la mayoría de investigación se ha hecho en esta área.

Este proceso de *emparejar* consiste en podar el espacio de posibles emparejamientos entre posibles servicios ofertados y peticiones. Un servicio de match requiere tanto *metadatos* ricos y flexibles como algoritmos de emparejamiento [Pay01].

El proceso general que se sigue en el emparejamiento de servicios ofertados y requeridos se basa en que cuando un agente envía una petición a un agente *matchmaker o emparejador*, éste le devolverá posteriormente como resultado, información respecto al servicio que mejor se adapta a la descripción dada en el requerimiento. Esta información incluirá las IOPEs (*Inputs, Outputs, Preconditions, Effects*) las cuales están reflejadas en el perfil del servicio así como información adicional de parámetros de servicio, información del proveedor etc., que el proveedor del servicio habrá comunicado con anterioridad al *matchmaker*.[20] De esta forma si como resultado hay varias correspondencias, entonces el cliente que solicita el servicio (o el agente en su lugar) puede usar esta información para refinar su elección de servicio a usar. Haciendo uso de esta información suplementaria permitirá a los clientes restringir su búsqueda, especificando la categoría de servicio (B2B, B2C etc.), recursos que pudiera recurrir,

[20] Si las peticiones son expresadas en un formato diferente de las realizadas en el anuncios surgirá la necesidad de transformarlas en algún punto del proceso: en el registro de anuncios haciendo que éste acepte y sea capaz de reconciliar las diferencias entre múltiples formatos, en la parte del cliente compilando la petición en algún formato y posteriormente convirtiéndola en el lenguaje de descripción utilizado, o por último en la etapa de transporte, por la acción de intermediarios entre el cliente y el registro.

grado de calidad del servicio etc. Aparece por tanto un conjunto extensible de parámetros de servicio, que restringen la búsqueda, alcanzando el objetivo de que los servicios devueltos satisfagan las necesidades.

En la práctica el cliente debería ya conocer las IOPEs y lo que no sabrá es el tipo de proceso, si será atómico o por el contrario se tratará de una composición de servicios en cuyo caso habrá que realizar pasos intermedios que consultarán el modelo de proceso para identificar el flujo de trabajo desde la entrada a la salida (con la posibilidad de que la salida de un proceso intermediario sea usada como entrada en el siguiente proceso atómico en la secuencia dada) hasta que todos los procesos involucrados sean ejecutados. Por tanto los agentes que realizan la petición de servicio, deberán ser capaces de interpretar el contenido de la ontología de proceso y el *grounding* para poder comunicarse con un agente proveedor de tal servicio [Pay02] [Mar02].

La idea principal que subyace en la aproximación de emparejamiento viene dada por la búsqueda de un servicio basada en el tópico o categoría y después en el uso de los parámetros de salida etc. El problema tal y como apuntaba Massimo Paolucci en [Mar02] es que el uso de categorías de servicio puede no tener significado si las categorías permitidas en la ontología son demasiado amplias. Por otra parte, para dotar de significado se necesitará especificar muchas clases diferentes, una por cada tipo de servicio. El uso de los parámetros de entrada y salida permitirán solventar este problema, permitiendo una definición implícita de los servicios. Esto requerirá el uso de ontologías menos precisas aunque, por el contrario, serán necesarias descripciones de servicios más esmeradas y un proceso de *emparejamiento* más complejo. La idea por tanto es soportar ambos tipos de emparejamiento, debido al resurgimiento de grandes ontologías y clasificaciones de servicios. Como afirma David Martin, en la jerarquía de clases que refleja los diferentes tipos de servicios, existe la posibilidad de introducir propiedades adicionales, específicas para cada tipo de servicio, de manera que estas propiedades podrían ser usadas para establecer la correspondencia.

Lei y Horrocks en [Lei03], describen algunos problemas derivados del diseño de la especificación de un perfil de servicio. Ellos afirman que mediante éste, se puede dar demasiada información acerca de un servicio de tal forma que esto puede dificultar el uso de un razonamiento automático que procese los emparejamientos entre descripciones semánticas. David Martin en [Mar03] como respuesta a este problema, afirma que para solucionarlo se requiere que se den algunas convenciones:

1) Que el proveedor exprese una clase de servicios usando una expresión lo más general posible.
2) Que el cliente exprese una petición para encontrar el servicio que desea usando una expresión lo más específica que pueda.
3) Cuando un cliente no tenga en cuenta el modo de restringir una propiedad, su petición deberá expresar esta situación.

Limitando la búsqueda semántica de servicios a estas convenciones se restringe de manera notable su funcionalidad. El uso de anuncios lo más generales posibles que engloben la mayoría de requerimientos dados para un determinado tipo de servicio, dejará fuera bajo estas restricciones a algunos servicios en los que el cliente también pueda haber estado interesado, como por ejemplo servicios mucho más restrictivos o

servicios con características similares pero con distintos valores para las propiedades de las restricciones.
Thomi Piliora et al. [Pil03], establecen los requerimientos que se han de tener en cuenta en el proceso de emparejar. Los autores afirman que deberían de ser tratados tanto aspectos sintácticos como semánticos, pero que primero debería examinarse la compatibilidad semántica antes que la sintáctica debido a la necesidad de que la petición y el anuncio de servicio deben de "hablar" del mismo tema.

4.7.6.1 Grados de similitud o match

El emparejamiento entre anuncios y peticiones se considera idóneo cuando ambos son lo suficientemente similares. Un *match* o emparejamiento vendrá determinado por los diferentes grados de similitud. El grado de similitud depende de la relación entre los conceptos (tomada de las ontologías) que se están comparando, y generalmente se reduce a la mínima distancia entre ellos en el árbol taxonómico.

La denominación de los grados varía según la literatura. [Pao02a],[Abe01],[Lei03],[W3C02]:

- *Exact*: cuando los conceptos tanto en la petición como en el anuncio son equivalentes.
- *Subclass of*: determinado por la relación "ser subclase de". Cuando los conceptos en la petición son subclase (relación directa) de los del anuncio. (En [Pao02a] es considerado como e*xact matching* también.)
- *Subsumption,* la cual puede ser de dos tipos:
    - Plug-in o Contained: cuando los conceptos en el anuncio A incluyen los de la petición P. Formalmente, ($P \subseteq A$). En este caso, la petición puede ser satisfecha debido a que los conceptos del anuncio son más generales que los de la petición, y por tanto, existe la posibilidad de que el cliente pueda cumplir sus objetivos. Sin embargo, no se considera que exista una relación de subclase directa (grado anterior)
    - Subsume o Container: cuando los conceptos en la petición incluyen los del anuncio; formalmente, ($A \subseteq P$). Este tipo de emparejamiento no satisface completamente la petición pero puede ser considerado como una solución parcial válida ya que puede permitir al cliente que realizó la petición ir alcanzando parcialmente sus objetivos o metas.
- *Fail ,nul o Disjoint* cuando no hay relación de inclusión entre los conceptos; formalmente, $(A \cap P) \subseteq \perp$

En [Gon01] se introdujeron nuevos tipos de emparejamiento que más tarde fueron adoptados por Li y Horrocks [Lei03] como extensión a los anteriormente expuestos:

- *Intersection* u *Overlap*. Si la intersección de un anuncio A y una petición P se satisface, es decir son compatibles; formalmente, $\neg (A \cap P \subseteq \perp)$. La idea de compatibilidad entre conceptos ya fue expuesta por David Trastour et al. en [Tras02].

Para entender el proceso de emparejamiento mencionado, hay que considerar la definición de "*Open World descriptions*" aportada por Tommaso Di Noia et al. en [Noi03]: *"La ausencia de una característica en la descripción de un anuncio o de una*

petición no se debe interpretar como una restricción de ésta. En su lugar, debe ser considerada como una característica que se podría refinar más tarde, o dejarla abierta si se considera irrelevante para el usuario". Esta definición clarifica la idea de que incluso cuando los servicios anunciados y los requeridos no tienen un emparejamiento exacto, puede ser posible o necesario usarlos en instancias específicas. Por tanto, los emparejamientos parciales también son importantes [W3C02]. Lo anterior es conocido como "*emparejamiento flexible*" para distinguirlo del exacto, considerado el más restrictivo de todos.

En [Col03] se establecen tres tipos de emparejamientos dados por las diferentes relaciones entre perfiles de petición y ofertados. Para expresarlo formalmente hacen uso de Lógica Descriptiva. Siendo T una ontología común establecida para la descripción de servicios:

- Implicación: $T \models (P \subseteq A)$ y $T \models (A \subseteq P)$[21]. Cada restricción impuesta por P es completada por A y viceversa. Da lugar al emparejamiento exacto.

- Consistencia: $(A \cap P)$ es satisfactoria en T, donde las restricciones no se excluyen mutuamente. En este tipo de emparejamientos es necesario establecer un límite en la distancia entre ambas descripciones, que se medirá teniendo en cuenta dicha ontología. Es decir, ¿cuántos destalles en P tengo que preguntar a la otra parte A? Emparejamiento potencial.

- Inconsistencia: $A \cap P$ es insatisfactoria en T, lo que implica que algunas restricciones de una descripción están en conflicto con las de la otra. En este tipo de emparejamientos cabe preguntarse por el grado de inconsistencia de A en P, es decir ¿Cuántos detalles en P tengo que eliminar para poder aceptar A?. Emparejamiento parcial.

4.7.6.2 Diferentes sistemas e investigaciones

La base de toda la investigación relacionada con el emparejamiento semántico de servicios, recae principalmente en los estudios de Ingeniería de Software llevados a cabo por Zaremski y Wing en [Zar95] y [Zar97], en cuanto a la reutilización de software y recuperación de librerías, mediante la especificación y posteriormente emparejamiento de componentes software. En [Zar97] extienden el concepto de *signature matching* [Zar95], para tener en cuenta las restricciones en las entradas y en las salidas en funciones y módulos. Este concepto es llamado *specification matching* y provee no solo información de tipo estático sino también descripciones de comportamiento dinámico. Los autores distinguen varios tipos de emparejamientos en la especificación de emparejamientos de software. Establecen diferentes clases de emparejamiento basados en precondiciones y postcondiciones, siendo los diferentes tipos de emparejamientos aquí tratados la base de trabajos posteriores en el área de emparejamiento semántico.

En [Syc99], [Syc02] Sycara et al. describen LARKS, en el cual los servicios son vistos como *frames* y sus slots *input*, *output*, *inConstraints* y *outConstraints* pueden ser usados para describir los atributos esenciales de un servicio. Los conceptos usados por

[21] Extensión del concepto de equivalencia dado por la visión simplista de tratar únicamente la equivalencia sintáctica.

las descripciones son definidas mediante un lenguaje de descripción de conceptos denominado ITL (*Information Terminological Language*). El proceso de emparejamiento en LARKS realiza tanto emparejamiento sintáctico como semántico, y se compone de un conjunto de filtros (independientes entre sí) que progresivamente restringen el número de anuncios candidatos a ser emparejados. En este trabajo se utilizan técnicas basadas en "*subsumption matching*" entre conceptos y en computación de distancias entre ellos, además hacen uso de técnicas de recuperación de información clásicas como TF-IDF (*Term Frecuency-Inverse Document Frecuency*).

En [Kle01] se describe una aproximación basada en ontologías que emplea las características de una taxonomía de proceso de recuperación sin sacrificar la precisión y la complejidad de cómputo del proceso de recuperación del servicio. Las preguntas también se expresan usando los mismos modelos de proceso. El algoritmo que empareja utiliza la relación semántica codificada en la ontología de proceso al emparejar el modelo del proceso del servicio con requerimientos.

La aproximación usada por [Gon01] para emparejamiento de servicios es un algoritmo basado en el árbol del subsumción dado por un Razonador de Lógicas Descriptivas. Las descripciones del servicio son especificadas en DAML+OIL y puesto que DAML+OIL se basa en Lógica Descriptiva, el razonador de DL es el corazón del algoritmo de emparejamiento. Los diferentes tipos de emparejamiento para un servicio S, se definen como (1) conceptos equivalentes a S, (2) subconceptos de S, (3) superconceptos de S que son incluidos por el concepto en la ontología de descripción del servicio, (4) los subconceptos de cualquier superconcepto directo de S cuya intersección con S sea satisfactoria.

Otra aproximación utilizada es cuando los servicios pueden también ser descritos como un grafo de RDF [Tras01] y el emparejamiento de anuncios se reduce a emparejar grafos de RDF. Este algoritmo se basa en "*visitor pattern*". Los anuncios emparejan cuando su nodo raíz lo hace y sus respectivos nodos hijos también emparejan. Esto significa que uno de los nodos es subtipo del otro y una regla que empareja es definida positiva o se cumple una regla por defecto.

El equipo *Intelligent Software Agents Group* de la *Carnegie Mellon University* (CMU) diseño una arquitectura basada en ATLAS (*Agent Transaction Language for Advertising Services*), lenguaje basado en DAML, para emparejamiento de servicios [Pay01]. Éste utiliza dos filtros: emparejamiento de atributos funcionales para determinar la aplicabilidad de los anuncios y emparejamiento de funcionalidades de servicio, para determinar si el servicio anunciado empareja el servicio requerido. El emparejamiento de atributos funcionales es alcanzado mediante la realización conjuntiva de comparaciones de pares guiados para las propiedades (tipo de servicio, calidad del servicio etc.). Diferentes tipos de inferencia son usados para testear cada par. El emparejamiento de la funcionalidad del servicio es llevado a cabo por inferencia de tipo subsumción en el conjunto de entradas y salidas de la petición y el anuncio.

De nuevo en CMU [CMU04], el *Daml-s Matchmaker* emplea técnicas de recuperación de datos, de IA, y de Ingeniería de Software para procesar la semejanza sintáctica y semántica entre descripciones de capacidad de servicios. El motor que empareja su sistema contiene cinco filtros para la comparación del *namespace*, la comparación de la frecuencia de la palabra, la semejanza de la ontología, subsumción en la ontología, y

filtro sobre las restricciones (*Constraints Filter*). El usuario configura estos filtros para alcanzar el grado deseado entre el funcionamiento y la calidad. Especial relevancia tiene el establecimiento de filtros para las restricciones o reglas sobre las entradas y salidas, ya que pocos son los trabajos que abordan este problema debido a la falta de madurez de un lenguaje de reglas. Para la especificación de éstas, hacen uso de RuleML [RuleML03].
Estas investigaciones son las que más se han tomado como base o referencia para otros trabajos [Pao02a]. Su modelo y algoritmo de emparejamiento semántico que plantean es el que ha sido utilizado por la mayoría de investigadores como base para sus sistemas. Plantean un sistema de emparejamiento basado en la semántica descrita en el Profile de los servicios y en la utilización de los registros UDDI para mantener las descripciones de los servicios, aunque el referente para el resto de investigadores ha sido el algoritmo de emparejamiento utilizado. Hacen uso de la ontología perfil del servicio y de DAML-S como lenguaje de especificación para las descripciones del servicio. Se asume que los servicios de la red anuncian sus interfaces (inputs/outputs) usando la misma ontología. En este trabajo se aborda la importancia para la clasificación del emparejamiento de las salidas. Un emparejamiento entre un anuncio y una petición de servicio consiste en emparejar todas las salidas de la petición con las del anuncio; y todas las entradas del anuncio con las de la petición. El de las entradas se usa más que nada para resolver las igualdades que se produzcan (ver tabla 4.4).

Tipo de emparejamiento	**Entradas**	**Salidas**
Exacto	out(A) = out(P)	in(A) = in(P)
Plug in	out(A) $\supset$ out(P)	in(A) $\subset$ in(P)
Subsumed	out(A) $\subset$ out(P)	in(A) $\supset$ in(P)
Fallo (ninguno)		

Tabla 4.4: Tipos de emparejamiento mediante entradas y salidas. [Pao03b].

El grado de coincidencia entre el proveedor y la petición de servicio del cliente dependerá del grado de similitud entre los parámetros de entrada y salida (IO's). Una vez comprobado el grado de similitud para cada uno de los IO's, el siguiente paso es el de ordenar todos los servicios almacenados. Para esta ordenación se diseñó un algoritmo que ordena los servicios resultantes del emparejamiento según el grado de similitud que tienen, y en el que los emparejamientos de los outputs tienen más peso que los de los inputs. El motivo por el que la prioridad de los outputs es mayor, es debido a que es más importante el emparejamiento de lo que el cliente espera obtener como resultado, es decir, la salida del servicio en cuestión. Dados dos servicios, primero se ordenan por el nivel de similitud de los outputs, y solo si se obtiene un empate, se ordenará por el nivel de similitud de los inputs.

En este algoritmo, una vez ordenados los servicios, se devolverá al cliente aquel servicio resultante del algoritmo de ordenación que es el que tenga un mayor grado de similitud o coincidencia.

Para este algoritmo, Paolucci et al. decidieron que en las peticiones del cliente, éste pudiera incluir el nivel de profundidad en los subarboles para los grados de similitud de plugin y subsume.

Una de las ideas más importantes que aportan Paolucci et al. en su algoritmo para que tenga un mejor comportamiento, es la de que tanto clientes como proveedores empleen las mismas ontologías de conceptos para dar valor semántico a cada uno de los IO's de sus Profiles. De esta manera, se facilita la tarea al motor de inferencia o razonador, porque solo necesita que se carguen las ontologías de conceptos que utilizaron para construir los Profiles para poder medir el grado de similitud que tengan los IO's que definiesen. Evitamos el caso en el que tanto proveedores como clientes empleen ontologías de conceptos diferentes para definir IO's que tengan el mismo valor semántico, porque en ese caso el razonador tendría que cargar un "*mapping*" de los conceptos definidos en diferentes ontologías que tienen el mismo valor semántico.

Este algoritmo es la parte principal del sistema emparejamiento que diseñaron. La arquitectura que emplearon para introducirlo fue utilizar UDDI como registro de SW [Pao02b], [Pao03]. Es decir, intentan relacionar ambas tecnologías, añadiendo la capacidad de uso de la semántica en UDDI para poder elaborar mecanismos de emparejamiento mucho más eficientes, que permitan la utilización de registros de servicios en UDDI ya implementados.

Se basan en la compilación de anuncios DAML-S en registros UDDI que permitan el emparejamiento entre ellos. Si UDDI es un registro de servicios que almacena descripciones WSDL de los servicios, sin ningún valor semántico, ellos trataron de modificar la estructura para permitir que aceptase descripciones de servicio como instancias de Profile para poder aportar el valor semántico a las capacidades de un servicio. Por lo tanto, tienen un sistema de emparejamiento que puede utilizar la búsqueda por *keywords* de UDDI o utilizar el algoritmo anterior para hacer búsqueda semántica por los servicios anunciados en UDDI.

Después de este algoritmo presentado por Paolucci et al., han sido varios los investigadores que han basado su sistema de emparejamiento en este algoritmo, añadiéndole algunas funcionalidades:

Charlie Abela [Abe01] desarrolló un sistema de emparejamiento basado en el algoritmo que desarrollaron Paolucci et al. Hace uso de un algoritmo que usa dos operaciones de filtrado para realizar el emparejamiento. El primer filtro limita el espacio de búsqueda a las restricciones definidas por el usuario en la petición. (búsqueda por *ServiceProvider*, *ServiceCategory* o *ServiceName*). El segundo filtro es un emparejamiento de las salidas que el usuario define en la petición, permitiéndose el uso de un solo filtro o ambos. En el resultado de un emparejamiento pueden darse varios servicios y por tanto se deberá crear un rango para especificar cual de ellos es el que más se ajusta a las necesidades del cliente. A veces será necesaria la intervención de éste para poder elegir aquél que considere más apropiado.

La aproximación desarrollada en LSDIS lab, UGA [Car02] es similar, pero puede albergar múltiples ontologías. La función de similitud está basada en la taxonomía de la ontología y responde de las relaciones padre-hijo entre conceptos. Intenta manejar totalmente conceptos no relacionados basándose en sus propiedades, El emparejamiento calculado tiene tres dimensiones, sintáctico, semántico y QoS (Calidad del servicio). Este último recibe el nombre de similitud operacional. En este trabajo de investigación

es importante destacar el uso de funciones matemáticas de similitud aplicadas en cada uno de los casos.

Otra aproximación fue desarrollada por *Information Management Group* en la *University of Manchester* [Lei03]. El prototipo descrito de su algoritmo matchmaking también se basa en las descripciones de DAML-S. Utiliza el razonador RACER para realizar emparejamientos semánticos entre los servicios, y utiliza JADE como plataforma de agentes. Como en el diseño de DALM-S Matchmaker, las peticiones del servicio en DAML-S se emparejan con los anuncios del servicio. Sin embargo, este algoritmo de emparejamiento no divide el procedimiento en varias partes; en su lugar intenta encontrar un emparejamiento semántico directamente de los perfiles de servicios especificados. En esta aproximación los perfiles son tratados como entidades enteras, y sus componentes no son tratados por separado. Una vez más al resultado del emparejamiento se le asocia un cierto grado de similitud.

Una alternativa al descubrimiento de SW mediante el *Profile* fue la desarrollada por Sharad Bansal y José M. Vidal [Ban03] de la universidad de *South Carolina*. Plantean un sistema de emparejamiento en el que el usuario construye una petición de servicio con los IO's que espera que tenga el servicio, pero en este caso, en vez de comparar con los IO's descritos en el Profile del proveedor, comparan con los IO's que se hayan descrito en los distintos procesos que forman el Process Model definido por el proveedor. Para ellos, la comparación con los IO's definidos en el Process Model es mejor porque hay IO's definidos aquí que tal vez no son definidos en el Profile. Sin embargo, en mi opinión la idea no es muy acertada, debido a que la comunidad de la Web Semántica, y en el caso particular de la comunidad de descripción semántica de SW, se ha desarrollado la subontología Profile con este fin, mientras que la subontología de Process Model está orientada a una descripción más "física" del servicio Web y de como es la secuencia de ejecución que ha de seguir éste. Otra de las desventajas que tiene este sistema de emparejamiento es el mayor coste de ejecución del algoritmo debido a que éste es mucho más complejo, lo que da lugar a un tiempo de ejecución del algoritmo de tipo exponencial.

Wolf-Tilo Balke de la universidad de California y Matthias Wagner del Laboratorio *Future Networking de Munich* [Bal03] idearon otro algoritmo de emparejamiento diferente. Primero realizan una búsqueda basada en *keywords*, y posteriormente, una vez obtenida una lista de resultados, se le pide al usuario que introduzca aquellos parámetros (IO's) que deban tener los servicios y los valores a introducir para éstos. Posteriormente un razonador elimina aquellos servicios que no tengan estos parámetros definidos y los resultantes serán ejecutados con los valores que introdujo el cliente. Los resultados de las ejecuciones se ordenan siguiendo algunas restricciones que beneficien al cliente, como la calidad de servicio. Si al final, o durante la ejecución de los filtros el sistema se queda sin ningún servicio Web, se le pide al cliente que reconsidere sus exigencias, sobre todo cuando defina los parámetros.

Algunas investigaciones tratan el emparejamiento semántico basado en las precondiciones y efectos [Zar97], [Moh02]:

$$(\text{precondiciones}_P \Rightarrow \text{precondiciones}_A) \wedge (\text{efectos}_A \Rightarrow \text{efectos}_P)$$

Es decir, el conjunto de precondiciones de la petición implica de manera lógica las precondiciones del anuncio, y el conjunto de efectos de la petición es implicado por los efectos del anuncio. La implicación lógica supone una condición demasiado fuerte y por tanto se hará uso de la subsumción lógica, al igual que se hizo en el tratamiento de las entradas y salidas (ver tabla 4.5).

Tipo de emparejamiento	**Efectos**	**Precondiciones**
Exacto	Eff(A) = eff(P)	pre(A) = pre(P)
Plug in	Eff(A) $\supset$ eff(P)	pre(A) $\subset$ pre(P)
Subsumed	Eff(A) $\subset$ eff(P)	pre(A) $\supset$ pre(P)
Fallo		

Tabla 4.5: Tipos de emparejamiento mediante efectos y precondiciones. [Moh02].

Los brasileños Ricardo Ferraz, Sofiane Labidi y Bernardo Wanghon en [Fer03] exponen una extensión del algoritmo de emparejamiento de Paolucci et al. En este caso el cliente define las precondiciones que piensa que debe tener el servicio para que emparejen en el mayor grado posible con las que los proveedores introducen en los Profiles. Para ello utilizan el mismo algoritmo de emparejamiento que el de los IO's (Inputs, Outputs). Incluyeron las precondiciones porque piensan que son unos parámetros importantes a tener en cuenta en las negociaciones, ya que puede darse el caso de Servicios Web que incluyan precondiciones que negocian el tipo de pago o incluso el tipo de tarjetas de crédito que admiten.

Sin embargo, tal y como se observará en el capítulo 7 este tipo de emparejamiento no se ha tenido en cuenta por la peculiaridad de los servicios de información de tráfico[22].

4.8 CONCLUSIONES

En este capítulo ha sido descrito el concepto de representación de conocimiento como una parte de la Inteligencia Artificial. Su cometido es el de diseñar, e implementar sistemas y lenguajes que representen conocimiento sobre el mundo, mediante la representación de objetos reales, así como eventos y relaciones.

Cabe destacar la gran importancia de los sistemas terminológicos o lógica descriptiva (DL) como medio de representación del conocimiento, así como la arquitectura de este tipo de sistemas formada por una base de conocimiento, un interfaz y por último un mecanismo de inferencia que permitirá razonar sobre la base de conocimiento para obtener información no explícita, permitiendo razonamiento de tipo terminológico o instancial. DL ha permitido crear una familia de lenguajes con diversos equilibrios entre expresividad (constructores) y complejidad del razonamiento que hacen posible la creación de ontologías como base de la Web Semántica.

[22] En el caso de servicios que ofrezcan información de tráfico gratuito a un usuario, después de la ejecución del servicio, el usuario dispondrá del conocimiento requerido, pero este cambio de estado no es un cambio tangible y por la tanto no queda muy claro la posibilidad de su uso en el proceso de descubrimiento de un servicio o en la composición, donde los efectos de unos servicios se convierten más tarde en precondiciones de otros.

Se ha podido constatar a través de los diferentes estudios previos por parte de otros autores, la gran cantidad de lenguajes, herramientas y metodologías que han ido surgiendo y evolucionando entorno al concepto de ontología como medio de representación de conocimiento. En general, el lenguaje usado determinará el tipo de razonamiento que puede llevarse a cabo, lo que hará que sea tenido en cuenta este factor a la hora de elegir uno de ellos.

Por otra parte, es interesante observar la gran aportación de las ontologías puesto que suponen no solo un aporte estructural a la representación de conocimiento, sino que permiten establecer mecanismos de inferencia y de este modo, obtener conocimiento no explícito a priori mediante el uso de axiomas y reglas. Cabe destacar de igual modo las ventajas de conjuntar las aportaciones concernientes a las bases de datos relacionales con este tipo de formalismo.

Resulta importante destacar que la importancia de los agentes en el proceso de compartir conocimiento está basada en su interacción con ontologías comunes permitiendo interpretar las comunicaciones, llegando a una comprensión mutua (y en última instancia) a un comportamiento predecible. En relación con el estado actual de la tecnología de agentes se puede observar una proliferación de metodologías de ingeniería del software para su desarrollo así como de herramientas y marcos de implementación, y ante la falta de consenso se podría aumentar el riesgo de obtener soluciones poco estables y escalables en la aplicación de los sistemas multiagente en el descubrimiento automático de servicios en la Web.

En el desarrollo de la Web Semántica podemos apreciar que la capa de ontologías y de reglas se encuentra en una etapa bastante estable, y que a partir de las recomendaciones del W3C se propone la creación de ontologías a partir de lenguajes de marcado de desarrollo, siendo el más reciente OWL. Por su parte para la definición de reglas el lenguaje SWRL se propone como la alternativa más apropiada. Finalmente, el desarrollo de las capas superiores se encuentra en su fase inicial, pero puede observarse que la firma digital forma parte de su núcleo de desarrollo, así como su integración con otras tecnologías que se están desarrollando para permitir el manejo de seguridad en el intercambio de información basado en XML.

En relación con los SW por medio de los cuales se busca el intercambio efectivo de aplicaciones en la Web, se puede advertir que los esfuerzos de estandarizacion tanto de W3C como de OASIS están confluyendo, y dicha evolución se ve caracterizada por la evolución de sistemas convencionales basados en servicios de directorios como UDDI hacia la búsqueda de técnicas que permitan descubrimiento automático de servicios.

Los SW son la nueva generación de la aplicación Web, autocontenidos, autodescriptivos, aplicaciones modulares que pueden ser publicadas, localizadas, e invocadas a través de la Web. La aplicación de técnicas de Web Semántica en ambientes distribuidos como los SW son de especial utilidad en el enriquecimiento de las descripciones de los recursos con los que interactúan dichos servicios. La composición dinámica de servicios requiere que entendamos sus capacidades así como su compatibilidad. Una automatización completa de la composición de servicios es aún un área de investigación en desarrollo.

Los SWS mejoran la capacidad de los agentes software para buscar servicios particulares y son un importante paso en la dirección de la implementación de la Web Semántica. Uno de los principales inconvenientes existentes en la actualidad se basa en que en el desarrollo de servicios por diferentes comunidades se tienden a definir diferentes ontologías para describir tales servicios, por lo que uno de los principales problemas que surge es la dificultad de conseguir interoperabilidad y la necesidad de realizar procesos de traducción de ontologías.

Resulta importante destacar el gran interés suscitado por instituciones académicas e industria como una tecnología capaz de llevar a cabo la construcción de aplicaciones Web distribuidas. Pero los estándares que en un principio surgieron y que se siguen utilizando en la actualidad carecen de ciertas características que no permiten localizar los servicios distribuidos por la Web de forma eficiente. Ni SOAP ni WSDL sirven de ayuda para la localización automática de los SW en base a sus capacidades. Por otra parte UDDI no representa tampoco las capacidades, y por tanto no ayuda a la búsqueda de servicios en base a lo que éstos proveen. Sin embargo las iniciativas llevados a cabo por DARPA y más tarde W3C (DAML-S y OWL-S) han dado como fruto una ontología o semántica para describir las propiedades y capacidades de los SW. DAML-S se asienta en el nivel de aplicación sobre WSDL y describe que está siendo enviado y no solamente como está siendo enviado (funcionalidad provista por WSDL). Usando DAML-S/OWL-S, el agente software será capaz de descubrir la capacidad requerida por el cliente observando en las declaraciones de las capacidades anunciadas por los servicios.

En relación con el emparejamiento de servicios semánticos, la ventaja principal que tienen los sistemas de emparejamiento semántico respecto a los sistemas de emparejamiento basados en los estándares de XML (UDDI, E-Speak etc.) es la utilización de significado o descripciones semánticas. Dos descripciones XML iguales de un servicio Web tienen un significado u otro dentro del contexto donde se encuentren, mientras que las descripciones semánticas si cubren esta carencia, porque saben distinguir semánticamente las descripciones. Por lo tanto, el emparejamiento semántico tiene a su favor respecto al emparejamiento implementado por sistemas basado en XML la independencia de contexto. Aunque han aparecido muchas aproximaciones en este campo, sin embargo no han supuesto una solución final, en la búsqueda de algoritmos de emparejamiento eficaces.

A partir del estudio de arquitecturas para desarrollo de SWS, se ha podido observar que las propuestas existentes se caracterizan por tomar como origen las ontologías descriptivas de servicios como OWL-S y transformar éstas en formalismos, para razonar sobre ellas asegurando de esta forma la consistencia del modelo. Sin embargo no plantean el proceso inverso, es decir, modelizar conceptualmente estos servicios y después obtener la especificación correspondiente en cualquier lenguaje de descripción. Algo que si ocurre en otras aproximaciones de carácter ontológico como por ejemplo, aquéllas que modelizan inicialmente en UML. De lo anterior podemos deducir la necesidad de investigar en búsqueda de métodos, metodologías, propuestas y arquitecturas software para el desarrollo de este tipo de servicios tal y como han sido desarrolladas en la construcción de ontologías.

Finalmente, partiendo de una revisión bibliográfica así como experimental, relacionada con herramientas existentes para la edición de ontologías y de SWS que permitieran la generación de código en diferentes lenguajes de marcado semántico, se comparó su funcionalidad. La primera necesidad que surgió fruto de este análisis fue la falta de una herramienta que combinase la visualización de ontologías con la creación de perfiles de búsqueda, que fuera independiente de cualquier editor básico y a su vez, que permitiera su utilización de forma aislada o su integración en una plataforma de agentes.

SECCIÓN III: PROPUESTAS Y CONTRIBUCIONES

CAPÍTULO 5

PROCESO SEGUIDO EN LA BÚSQUEDA DE LA SOLUCIÓN A LA PROBLEMÁTICA

La participación en distintas actividades relacionadas con la información de tráfico, a través de varios proyectos europeos, y mayoritariamente dentro del dominio[23] de aplicación denominado "Servicios de información al viajero" (TIS), me ha permitido conocer en profundidad la problemática existente en los sistemas ITS relacionados con este campo y la forma en que las administraciones europeas han asumido este reto, sus esfuerzos y sus deficiencias.
El desarrollo de actividades enmarcadas en el subdominio "Servicios basados en Internet y Telecomunicaciones" me hizo tomar contacto con tecnologías relacionadas con el lenguaje de marcado, y su utilidad para ser aplicadas en el mundo de los ITS. Sin embargo, en el proceso creativo de estas actividades quedaron patentes algunas deficiencias en cuanto a la ausencia de vocabularios comunes para ser usados por aquellas entidades involucradas en la gestión, intercambio y distribución de este tipo de información. Desde el punto de vista del usuario se identificaron inicialmente una serie de problemas iniciales como el gran volumen de información distribuido, y la disposición de éstos a través de fuentes heterogéneas y a su vez en múltiples formatos.

El planteamiento inicial de este trabajo fue de alguna forma distinguir qué problemas existían en cuanto a la manera de distribuir información por parte de proveedores, y la posibilidad de que el uso de un vocabulario común de tráfico vial basado en lenguajes de marcado, pudiera ayudar a simplificar el proceso de búsqueda independientemente de la fuente donde esta información se encontrase.

La determinación de este objetivo marcó el inicio de mi investigación, y fueron algunos trabajos que en ese momento se estaban realizando en otros campos los que alentaron mi decisión inicial.

Por todo lo expuesto anteriormente surgieron dos cuestiones iniciales como base de una investigación exhaustiva:

- **¿Qué vocabularios comunes se podían definir para representar información de tráfico en lo relacionado con accidentes, medidas, localizaciones, tipos de vehículos etc.?**

[23] Los proyectos que forman parte del MIP han sido organizados siguiendo siete dominios de aplicación más dos dominios adicionales (Project Management y Horizontal Issues). Ver sección 3.6.2

- **¿Qué mecanismos eran necesarios para permitir que cualquier usuario tuviera acceso de forma transparente o a través de interacciones sencillas a la información deseada por él, expresada según sus requerimientos?**

Tomando como punto de partida estos trabajos y la identificación de las distintas fuentes que me permitieran la creación de este lenguaje de marcado, la idea de ir un poco más allá surgió de la lectura de artículos así como de interesantes y fructíferos comentarios de investigadores del Instituto de Robótica como Gregorio Martín, sobre la visión de la Web Semántica y el uso de ontologías para alcanzar plenamente mis objetivos iniciales. El lector podrá encontrar todos los detalles sobre el proceso de modelado de los elementos de información de tráfico vial en el capítulo 6.

Paralelamente a esta conclusión surgió en vista de la revisión de las investigaciones de otros autores así como la lectura de importantes y decisivos artículos como "*The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*" de Tim Berners Lee et al., la necesidad de incorporar un nuevo campo de trabajo en mi investigación: los agentes. Su uso quedó plenamente justificado debido a que las tecnologías de agentes y en especial los sistemas multiagentes (donde los agentes interaccionan unos con otros) ofrecen mayores promesas de flexibilidad, fiabilidad y modularidad que las aplicaciones distribuidas tradicionales. Por tanto, y, teniendo en cuenta su habilidad para interaccionar con sistemas remotos de todo tipo y, realizar tareas de forma autónoma sin la constante comunicación con el usuario, pude descubrir que los agentes cumplen la condición fundamental de cualquier sistema con capacidad para procesar el masivo volumen de información disponible en el World Wide Web dedicado a tráfico.
A su vez, ciertas características de los agentes como la repartición de tareas, mayor tolerancia a fallos que sistemas centralizados, eficiencia debido a la simultaneidad en la ejecución de las tareas, optimización conseguida gracias a que los agentes pueden crearse o eliminarse según las necesidades y recursos disponibles etc. reforzaron la decisión de su uso.
No obstante, los agentes únicamente han constituido un medio o herramienta para alcanzar mis objetivos y no han supuesto ninguna aportación. La extensión de metodologías sobre agentes, desarrollo de nuevos entornos de desarrollo etc. ha quedado fuera del alcance de esta tesis.

Por tanto, mi cometido inicial de construir un vocabulario común se tradujo por una parte en la **construcción de ontologías de tráfico,** pensadas como un medio de publicar y por tanto distribuir información, que a su vez permitiera captar el significado de cada uno de los elementos definidos en ella, y la de integrar estas ontologías en una plataforma de agentes que permitiera gestionar la información de forma adecuada, gracias a su autonomía y eficiencia.

La construcción de la ontología me llevó a una revisión exhaustiva de la literatura y a una profundización en las diferentes tecnologías relacionadas. Era necesario conocer con exactitud las distintas metodologías que existían, lenguajes, herramientas así como el estudio de las fuentes del conocimiento que me permitieran su creación.

Llegados a este punto, se concibieron dos situaciones que a continuación se exponen:

a) La identificación como fuente principal y por tanto, la base de las ontologías creadas, del modelo conceptual semántico y diccionario de datos, correspondiente a la reingeniería inversa del actual Concentrador de Información de Tráfico (CIT) de la DGT que en ese momento estaba realizándose en el seno del grupo de investigación al que pertenezco.

b) Por otra parte, la revisión de las metodologías que en ese momento solo planteaban la construcción de ontologías desde cero y no como resultado de la traducción desde modelos de datos.

Las dos situaciones planteadas, determinaron la necesidad de **extender los métodos o metodologías, que hasta entonces existían, para la construcción de ontologías**, tomando como base la experiencia propia. Esta extensión metodológica tuvo su principal argumento en la decisión de obtener modelos formales basados en lógica descriptiva[24], a partir de los modelos de datos provenientes de las fuentes citadas.

El desarrollo de las ontologías ha sido un trabajo arduo y difícil, que involucró diversas decisiones importantes a menudo en falso por problemas que iban surgiendo con posterioridad. Los dos elementos clave en este proceso fueron primeramente disponer de un mecanismo de descripción de recursos (lenguaje de desarrollo) que permitiera expresar conocimiento (y por tanto, superar la barrera impuesta desde la creación de Internet: *la asunción de ser un medio para humanos y estar orientado a la presentación, no al significado*) y dependiendo de esta elección una herramienta de edición apropiada. Se pudo resumir esta fase en el intento de dar respuesta a esta otra pregunta de investigación:

- **¿Qué principios, tecnologías y herramientas de representación del conocimiento y de definición de ontologías son las más apropiadas para abordar la representación del dominio de tráfico escogido?**

En ese momento, la representación de conocimiento mediante ontologías era posible mediante los lenguajes de desarrollo RDFS y DAML +OIL. La elección del segundo ha sido expuesta en el estado del arte, capítulo 4. La aparición de OWL fue posterior al inicio de la fase de implementación del prototipo (necesario para validar las propuestas) y como todavía no era una recomendación de W3C y puesto que las herramientas no lo soportaban, se optó por continuar con el primero debido a la necesidad de probar experimentalmente el trabajo. Por otra parte, OWL es una evolución de DAML+OIL y lo expuesto para el primero es aplicable al segundo.

Tras la revisión de las diferentes organizaciones para la elaboración de estándares en el área de los agentes, se pudo apreciar que FIPA era la más consolidada y documentada, estando plenamente aceptada por la comunidad de investigadores. Una de las funciones de FIPA es la de proporcionar una serie de normas para implementar y diseñar sistemas

[24] Lógica Descriptiva es un lenguaje de representación de conocimiento adaptado para expresar conocimiento sobre conceptos y jerarquías entre éstos, mediante la creación de ontologías formales. (Ver sección 4.3.2.1)

multiagente de forma correcta. Debido a la gran diversidad de sistemas, FIPA proporciona un estándar que constituye un marco de referencia. Su modelo de administración de agentes me permitió establecer los elementos básicos que deberían formar parte del sistema multiagente. Una vez, identificado el modelo de referencia adecuado, era necesario elegir el marco de trabajo apropiado para el desarrollo de software, orientado al despliegue de sistemas multiagentes y que a su vez cumpliera con los estándares de FIPA para agentes inteligentes. JADE es un marco de referencia obligada para el resto de sistemas desarrollados y se ha convertido en una de las plataformas más populares de sistemas multiagentes tanto por su difusión como por su potencia. La elección de JADE fue motivada por una serie factores que éste cumple: gratuidad, ser código abierto, estar plenamente documentado, seguir las especificaciones FIPA, y por último el ofrecer mejoras y actualizaciones continuas.

En ese momento, fue importante tener conocimiento de algunas afirmaciones aseveradas por el consorcio W3C como la que a continuación cito: "*Hoy en día, el principal uso de el World Wide Web se basa en el acceso interactivo a documentos y aplicaciones. En la mayoría de los casos, tal acceso es realizado por humanos, a través de navegadores de Web, reproductores de audio, u otros sistemas finales interactivos. La Web puede aumentar de forma significativa su poder y alcance si ésta logra extenderse para dar soporte a la comunicación entre aplicaciones, desde un programa a otro*." La profundización en el estudio de este planteamiento, así como el análisis de bibliografía relacionada me llevaron a la conclusión de que la utilización de SW (de tráfico) suponían un paso importante para alcanzar resultados que ayudaran a conseguir una aproximación fundamentada a la solución de la problemática definida inicialmente.

Sin embargo, mientras se estaban desarrollando las ontologías de tráfico, y se estaba revisando el estado del arte concerniente a los SW, el descubrimiento de las labores desempeñadas por la coalición de OWL-S (inicialmente DAML-S)[25], me hicieron comprender que el tratamiento de la información contenida en los portales Web actuales podía ser utilizada a través de SWS haciendo posible la consecución de mis objetivos e incluso el beneficio de aprovechar la labor de construcción de las ontologías creadas debido principalmente a dos razones:

a) La propia naturaleza de los SWS para describir los servicios que necesita de conceptos pertenecientes a ontologías de dominio para la especificación de sus parámetros y,

b) Como más tarde descubrí, el uso de ontologías soporte (de tráfico vial) como medio para poder llevar a cabo la búsqueda de información, traducida en este caso en el emparejamiento de peticiones y anuncios de servicio, capaces de proveer dicha información.

Establecida la necesidad de crear descripciones de SWS y de nuevo tras una revisión de las distintas herramientas de construcción relacionadas, se observó fruto de este análisis la falta de un **entorno de desarrollo** que combinase la visualización de ontologías con

[25] DAML-S y OWL-S son ontologías para la descripción semántica de servicios (ver sección 4.7.4).

la creación de perfiles de búsqueda, que fuera independiente de cualquier editor básico[26] y a su vez, que permitiera su utilización de forma aislada o su integración en una plataforma de agentes. Por este motivo, se procedió a la creación de ***OntoServices***, y para su desarrollo se llevó a cabo una profunda revisión de trabajos entorno a la personalización y el uso de seguridad aplicadas a la gestión de perfiles de usuario. Lo anterior dio como resultado la creación de un **servidor de perfiles con labores de caché**, mediante el cual se logró optimizar considerablemente el proceso de búsqueda. El análisis, diseño, implementación y ejecución de este entorno ha sido ampliamente detallado en el capítulo 9.

La decisión del uso de SWS, Ontologías y SMA determinó el estudio de la Arquitectura Orientada a Servicios (SOA) y por tanto la identificación de sus diferentes componentes. En esta arquitectura destacan los agentes solicitantes o clientes, agentes proveedores y agentes intermedios. La misión de los últimos es la de mediar para que pueda tener lugar una correcta comunicación entre los primeros y los segundos. En la literatura encontrada definían un *agente intermedio* como aquél que ayuda a otros a localizar y conectar con agentes proveedores de servicios. La anterior definición y el objetivo marcado previamente de encontrar aquel servicio que más se ajustara a los requerimientos del usuario, me hicieron declinar por los denominados modelos de intermediación *matchmaking* donde el resultado devuelto por el agente intermedio (denominado en este modelo *matchmaker)* es una lista o referencia de proveedores que pueden proporcionar el servicio, y en estos casos, es el propio solicitante el encargado de contactar y negociar con el proveedor del servicio. En este tipo de modelo, se necesitaba todavía algo fundamental como era un **algoritmo de emparejamiento** capaz de abordar de forma eficaz el cometido de encontrar el servicio adecuado haciendo uso de las descripciones semánticas que los definen.
Posteriormente a un análisis exhaustivo de las distintas aproximaciones ya finalizadas, y aquéllas que se estaban llevando a cabo en ese momento, así como de las ontologías aportadas por la coalición OWL-S para la descripción de los servicios, se pudo apreciar que algunos aspectos funcionales podían ser objeto de estudio y de esta forma establecer mejoras en los algoritmos iniciales.
Al igual que la mayoría de aproximaciones se tomó como punto de partida el trabajo realizado por CMU, y el algoritmo presentado por ellos en el año 2002 supuso la base de esta investigación. Analizando en profundidad esta aproximación y también los trabajos que se estaban realizando tanto por la coalición como por otros entornos de investigación académicos, llegué a la conclusión de que eran posibles ciertas optimizaciones tanto en el algoritmo en sí, como en las ontologías de descripción de SW relacionadas. Algunas de estas ideas tomaron como foco de estudio las propias ontologías de descripción (subontología *profile*) y otras se centraron en la creación de una taxonomía de servicios de tráfico basada en los trabajos de normalización realizada por ISO TC 204, en el cual participo en labores de revisión. Las propuestas a las que hago referencia son:

a) Propuesta de un nuevo **parámetro no funcional**, denominado **valor añadido**, no incluido por la coalición OWL-S en sus ontologías de descripción de

[26] Se entiende por Editor Básico, áquel mediante el cual se es capaz de construir ontologías, pero no Servicios Web.

servicios. La adición de este nuevo parámetro supone aumentar las capacidades de descripción de un servicio y por tanto la optimización del proceso de descubrimiento de éstos, en cuanto a que el cliente no solamente obtiene aquel servicio que está buscando, sino otra serie de servicios que complementan al primero, y los cuales pudieran ser necesitados en búsquedas posteriores. De igual manera, desde el punto de vista de proveedor se facilita el anuncio de servicios que complementan o pudieran interesar al servicio anunciado. La adición de este nuevo parámetro aporta una mayor expresividad en cuanto a la caracterización y relación entre servicios.

b) Propuesta de una **ontología de categorización de servicios de tráfico**. (*Jerarquía de perfiles*) distinta a las hasta ahora utilizadas como NAICS y UNSPSC. Esto ha mejorado considerablemente tanto la creación de un determinado perfil como su posterior búsqueda mediante filtros por categorías, debido principalmente al uso de la taxonomía o jerarquía propuesta, específica con el tipo de servicios que se manejan. La creación de esta ontología de clasificación de servicios ha cubierto este tipo de infraestructura ontológica a la vez que puede suponer una base sólida en el uso de servicios en los ITS. El fundamento principal de la conclusión anterior reside en el hecho de haber tomado como base el trabajo de estandarización llevado a cabo por el WG TC 204 de ISO.

c) Propuesta para **medir la calidad de servicio en servicios de tráfico**. En la especificación de OWL-S se incluyen constructores para expresar parámetros relativos a la calidad de servicio (QoS), sin embargo, no proveen de un conjunto detallado de clases y propiedades que permitan representar esta métrica. Tomando como punto de partida anteriores trabajos y viendo la dificultad de aplicar dichos modelos a los SW de tráfico, se creó un modelo diferente que tuviera en cuenta las características propias de los portales de información de tráfico objeto de estudio. Debido a la no disponibilidad de ninguna escala de calidad inicial para este tipo de servicios, se ha propuesto una **métrica basada en determinados factores de calidad** que ha permitido su creación. De esta manera, cualquier servicio provisto podrá especificar su grado correspondiente, el cual será considerado en la búsqueda si el cliente así lo requiere.

Los detalles del estudio entorno a las aproximaciones de algoritmos, la propuesta y las pruebas de funcionalidad del denominado emparejador semántico que han permitido establecer las comparaciones entre el algoritmo de CMU y su extensión propuesta en esta tesis pueden encontrarse en el capítulo 7.

Tras una revisión en el dominio de tráfico, se pudo constatar la carencia actual de SW de información de tráfico por lo que de nuevo surgió una cuestión a resolver:

- **¿Era posible obtener un marco de referencia que asistiera en la descripción y posterior uso de información distribuida a través de páginas Web tradicionales, para convertir éstas en SWS?**

Para solucionar esta cuestión se definió un **marco de trabajo para la conversión de portales Web convencionales de información de tráfico en SWS** de manera que la información aportada pudiera ser almacenada con la adición de significado consiguiendo a su vez, potenciar nuevas capacidades que en un principio no existían. La descripción de las propuestas referentes a los SWS y que han sido detalladas anteriormente, el marco de trabajo definido, así como su uso para la descripción completa de un servicio de tráfico concreto se pueden consultar en el capítulo 8.

Por último, una vez concluido todo el proceso de análisis, diseño e implementación se procedió a la integración de todos los elementos en una arquitectura diseñada en capas como una concretización de arquitecturas generales existentes.
En el capítulo 10 se presenta y valida una **arquitectura de integración de SWS** de información sobre tráfico vial mediante la construcción de un prototipo software capaz de hacer uso del algoritmo de emparejamiento propuesto, así como la implementación de este último como un componente dentro de un sistema multiagente que permita la publicación, descubrimiento e invocación de los servicios.

En la figura 5.1, se puede observar de forma esquemática el proceso metodológico seguido desde la identificación de los problemas iniciales, hasta el resultado final de construcción de un prototipo que integra todos los elementos surgidos de esta investigación, y las pruebas de verificación de funcionalidad realizadas.

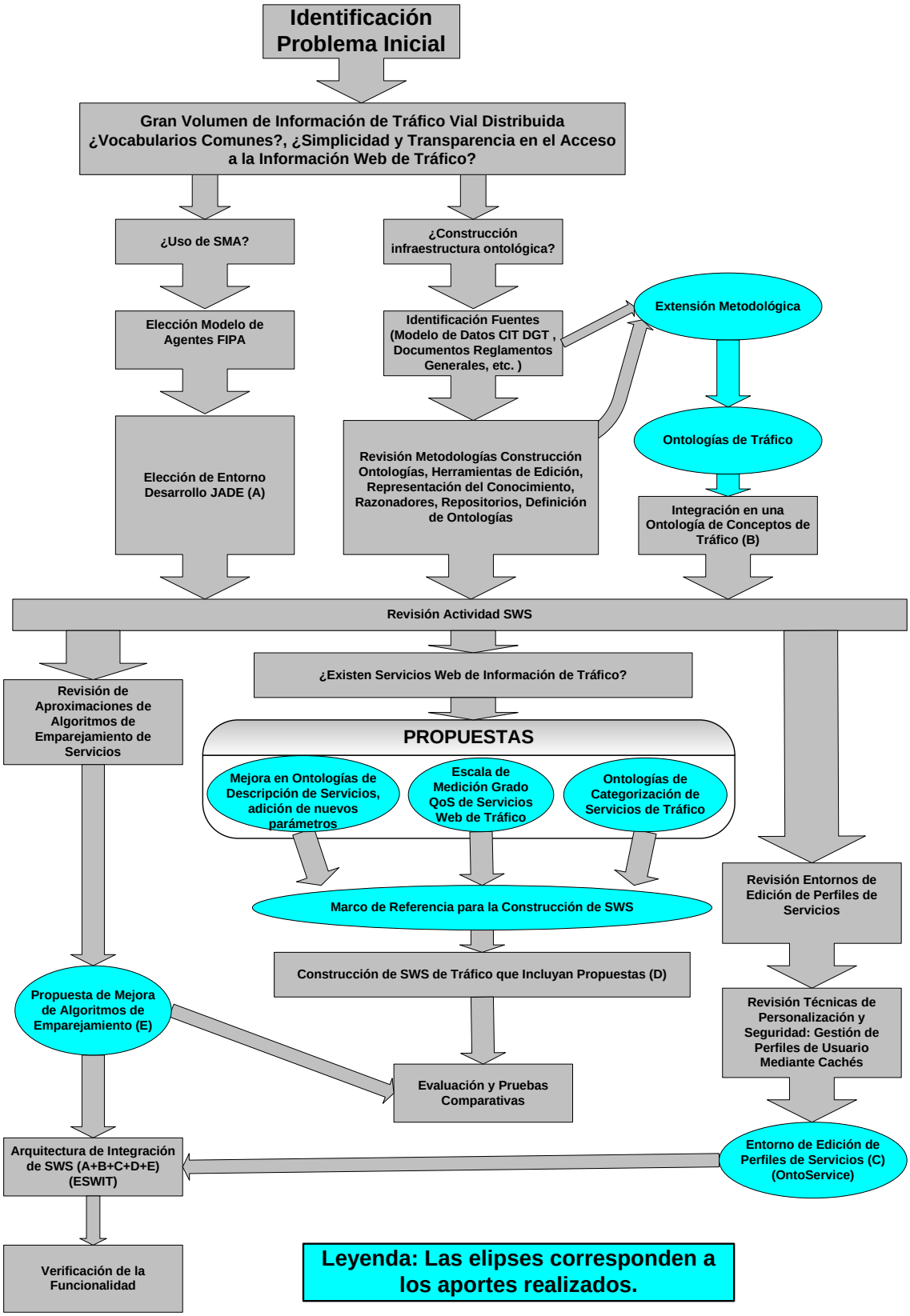


Figura 5.1 Visión global de las etapas desde la descripción del problema. Aportaciones concretas.

CAPÍTULO 6

MODELADO ONTOLÓGICO DE ELEMENTOS DE INFORMACIÓN DE TRÁFICO VIAL

6.1 RESUMEN

En este capítulo se muestra en que ha consistido el modelado de elementos de información de tráfico vial. Primeramente se introduce la problemática existente derivada de la carencia de descripciones semánticas, y se plantea su uso, así como la necesidad de construir ontologías. Posteriormente se detalla en que consiste el desarrollo de una infraestructura conceptual que permita organizar y conectar los elementos de información que constituyen cualquier elemento característico del tráfico vial, así como el modelamiento de la estructura y semántica de los diferentes elementos y el establecimiento de sus relaciones, permitiendo la relocalización de elementos que formaban parte del mismo modelo conceptual en diferentes dominios conceptuales o subdominios. Tomando como punto de partida el análisis de las diferentes metodologías o métodos expuestos en el capítulo 4 de la presente memoria, y los beneficios del uso de un modelo semántico formal, se expone una propuesta de un método para obtener un modelo semántico formal a partir de un modelo de ER.
Se describen los detalles en el desarrollo de la ontología, destacando las diferentes fuentes de información utilizadas, así como los aspectos a tener en cuenta en la especificación. Por último, tomando como base la arquitectura general de un sistema de representación de conocimiento basado en DL, se muestran las dos partes integrantes de éste (TBox y ABox), con ejemplos de su construcción mediante diferentes vistas al modelo ontológico creado y describiendo alguno de los subdominios más representativos de éste.

6.2. INTRODUCCIÓN: PLANTEAMIENTO DE LA PROBLEMÁTICA

Actualmente, existen vocabularios o lenguajes que describen conceptos y estructuras de datos relacionados con tráfico, pero son solo descripciones sintácticas, carentes de semántica. Dado que al menos hasta donde llegó la búsqueda y revisión bibliográfica de esta tesis, no se encontraron ontologías o vocabularios con valor semántico que aporten significado a los conceptos y sus relaciones, que pudiéramos tomar como base para nuestra investigación, este vacío fue cubierto mediante su construcción.

Por ello, el objetivo a alcanzar en esta parte de la investigación ha sido desarrollar un esquema de representación de un dominio particular, ***información de tráfico vial***

ofrecida al viajero a través de servicios basados en Internet, con una semántica bien definida. El uso de esta semántica nos permitirá obtener un conocimiento formalizado que hará posible el desarrollo de una arquitectura de integración de SWS de información de tráfico. Las ontologías aquí creadas servirán para describir o dar el valor semántico a parámetros pertenecientes a los servicios de tráfico que se definan, así como para la construcción de la base de conocimiento utilizada para dichos servicios.

El análisis de nuevas tecnologías para la difusión de información de tráfico se hace indispensable, debido a la necesidad de mantener a los conductores debidamente informados.

Varios aspectos describen la problemática actual relativos al acceso y distribución de la información:

- Un gran volumen de información de tráfico se distribuye entre varios sitios web. El principal problema para un usuario que necesita información de este tipo es encontrar estos sitios web y además tratar con los diferentes accesos a ésta, así como sus diferentes formas de presentación.
- Por otra parte, un usuario puede necesitar información de diferente naturaleza o tipo, y por lo tanto el almacenaje de toda esta información en un solo sitio web no es viable a nivel de costes de almacenamiento ni incluso a nivel de mantenimiento.
- El tratamiento de la información no permite realizar inferencias sobre ella de manera que pueda ser obtenido como resultado, información que a priori no estuviera explícitamente detallada.

La difusión de información de tráfico abarca un gran abanico de posibilidades que hacen que el tratamiento de la información se deba de abordar de manera eficaz. El usuario se ve obligado a tratar con un gran volumen de información que aparece a su vez distribuida por los diferentes *portales web* de administraciones o entidades privadas que la ofrecen. A su vez, la aparición de múltiples dispositivos de usuario final ha dado lugar a que desde el punto de vista del usuario haya una gran dificultad para encontrar estos lugares y tratar con las diferentes representaciones y accesos a la información.

Se plantean varios requisitos entorno a esta problemática:

a) El usuario debe obtener la información adecuada (concisa y clara) a sus requisitos.
b) La información debe alcanzar un grado de confianza que permite tener la certeza de la veracidad de ésta.
c) Información actualizada, ya que de lo contrario podría suponer la toma decisiones incorrectas, las cuales no se ajusten a la realidad.
d) Posibilidad de acceso desde diferentes medios que de igual manera permitan su obtención de forma contextualizada y personalizada.

Con todo, los usuarios u operadores tiene que saber en todo momento como poder tratar con esta información y tomar decisiones a veces en muy poco espacio de tiempo. La

contextualización de la información así como la adopción de un carácter conciso de ésta, permitirán en cierta medida simplificar este proceso.

En este sentido, hay que indicar que, entre otras formas posibles, es habitual encontrar información más o menos dispersa, e incluso "disfrazada" y por tanto, el usuario debe hacer un análisis visual (o auditivo) para sintetizar y concretizar la información relevante para él.

Como consecuencia de lo anterior, la aplicación manual de las búsquedas de información, que fundamentalmente se realiza en la actualidad, presenta claros inconvenientes. Por una parte, la enorme cantidad de tiempo que los usuarios tienen que dedicar a familiarizarse con las diferentes fuentes de información para obtener un alto rendimiento, y por otra parte, el riesgo, siempre presente, de no considerar cierta información, que pueda ser importante y que por alguna razón pase desapercibida.

Actualmente son numerosos los desarrollos que permiten primero distribuir y más tarde obtener información de tráfico. Hasta hace muy poco tiempo la idea de encontrar información sobre tráfico estaba concebida para o limitada por la obtención de la información de tipo Pre-Trip (antes de efectuar el viaje); el receptor de ésta sólo era capaz de recibir información previa a su posible desplazamiento por la red de carreteras. Posteriormente, el avance en telemática permitió el desarrollo de diversas tecnologías como el RDS-TMC, que originaron un gran cambio en cuanto a la concepción de la difusión de información. El usuario y receptor, puede hoy en día recibir información On-Trip (en ruta). El concepto de Ubicuidad por ejemplo, se ha convertido en algo esencial para posibilitar la interoperabilidad entre los dispositivos móviles con acceso a Internet, bajo condiciones diferentes. Sin embargo, el principal problema, la búsqueda y la obtención de información ajustada a los requerimientos del usuario sigue siendo el principal escollo a superar. Generalmente, el proceso de búsqueda está basado en criterios puramente sintácticos y se limitan a encontrar las ocurrencias de palabras clave en el texto. Esta forma de operar puede conducir a recuperar información irrelevante cuando se usa una palabra clave en un contexto distinto al deseado, o puede ignorar completamente información que sí sea relevante si en cambio se emplean términos diferentes en su contenido.

El actual procesamiento de la información y su posterior distribución a través de las páginas web en Internet o en distintos soportes de almacenamiento magnético hacen posible su tratamiento informático. Sin embargo, la representación obtenida es únicamente interpretable por el hombre, y, en todo caso, el tratamiento informático que puede hacerse a esta forma de representación, si bien útil, es bastante limitado.

Siendo conscientes de la necesidad de asistencia en la búsqueda y tratamiento de la información, se ha abordado el desarrollo de una arquitectura basada en semántica que cubra estas necesidades de una manera inteligente.

¿Por qué semántica?

En la actualidad, las diferentes propuestas de metadatos pueden ser vistas como una solución a uno de los conflictos más importantes de la WWW existente hoy en día: La gran dificultad que existe al intentar automatizar tareas que realizan trabajo útil en la

web, debido principalmente a que en sus orígenes ésta fue únicamente pensada para el consumo humano de información. Para solventar este problema se hacen necesarios mecanismos que permitan precisar descripciones sobre la información que se encuentra disponible en la Web, de tal forma que las máquinas sean capaces de interpretar la información circulante.
Tanto RDF/RDFS como DAML+OIL u OWL son lenguajes basados en una representación XML, lo cual les hace fácilmente legibles y permiten compartir ontologías y reutilizar conocimiento orientado a la Web para definir nuevas ontologías.

A las ontologías aquí creadas las denominaremos en general "ontologías de soporte", porque como veremos son aquellas ontologías en las que el sistema prototipo se apoya para su correcto funcionamiento.

6.3. ONTOLOGÍAS EN TRÁFICO

Como ya se ha comentado en el capítulo 3 (sección 3.4), actualmente, existen vocabularios o lenguajes que describen conceptos y estructuras de datos relacionados con tráfico, pero son solo descripciones sintácticas, carentes de semántica. Por lo tanto, aunque existen numerosos trabajos y desarrollos que hacen uso de lenguajes de marcado XML (difusión de información, intercambio de información, modelado de tráfico), no se conocen especificaciones semánticas como las propuestas en este trabajo.

6.3.1 ¿Por qué se necesitan ontologías de tráfico?

Todo lo expuesto anteriormente evidencia que el tratamiento informático actual de la información de tráfico es muy limitado y susceptible de ser mejorado desde distintos puntos de vista. Son precisamente estas ideas las que se han tomado como punto de partida de esta investigación.

El esquema de representación cumplirá los siguientes aspectos:

- Ser interpretable por el ordenador y fácilmente intercambiable entre aplicaciones.
- Adherirse en sus aspectos sintácticos a los estándares de representación de información existentes.

De este modo, dada una información específica de tráfico, se dispondrá de una herramienta conceptual y metodológica que permita producir una representación en la forma base de conocimiento que podrá ser explotada por distintas aplicaciones informáticas para proporcionar servicios inteligentes y otros usos que requieran del conocimiento contenido en dicha información específica.

Como ejemplos de aplicaciones posibles basadas en el esquema de representación elegido, pueden citarse los siguientes:

- Un sistema de consultas y búsqueda inteligente de información de tráfico.
- Una herramienta para la visualización de la información estructurada.

Si un usuario u operador del sistema necesita conocer información específica sobre los accidentes acontecidos en una determinada carretera o localización, posiblemente quiera saber detalles sobre los vehículos implicados, tipos de vías etc. y por tanto este tipo de consulta involucre no solo una base o fuente de conocimiento sino varias, de ahí la importancia del uso de ontologías, algunas de cuyas características son la distribución y la posibilidad de inferir conocimiento no explícito a priori.

Como caso concreto pensemos en la ocurrencia de un accidente en una autopista de peaje concreta, como pueda ser la "AP-7". Haciendo uso de la especificación terminológica de este tipo de vías y considerando los atributos o propiedades (número de tramos de peaje, origen, destino, vías alternativas, denominación) podríamos establecer una serie de cuestiones que de manera inmediata serán resueltas por nuestro sistema de inferencia:

- ¿Qué número de tramos de peaje hay en la totalidad de la autopista A-7? 4
- ¿Cuál es el origen y destino para cada uno de esos tramos? Tramo 0: La Jonquera-Puzol, Tramo 1 Silla-Alicante etc.
- ¿Qué carreteras se pueden convertir en alternativas? N-340.
- ¿Cuál es el origen y el destino de estas carreteras alternativas? Origen: Cádiz, Destino: Barcelona.
- ¿Qué denominación particular reciben cada una de ellas? AP-7 (A-7) recibe el nombre de "Mediterráneo".

Toda esta información forma parte de la base de conocimiento de dicha ontología, pero podemos ir todavía más lejos, es decir podemos preguntar por ciertos elementos que en principio no han sido especificados en este dominio, y que formarán parte de una especificación en otro lugar que no tiene porque ser físicamente el mismo. Por ejemplo:
¿A qué comarcas, provincias, comunidades o países corresponden cada uno de los orígenes y destinos? Podríamos averiguar que Silla es una población de la provincia de Valencia, perteneciente por tanto a la Comunidad Valenciana y por tanto encuadrada dentro del territorio español y que como tal, cualquier ciudadano de Silla deberá tener nacionalidad española etc. Para lograr este objetivo, se hará uso de asociaciones entre diferentes ontologías estableciendo enlaces entre distintos dominios y la posibilidad de estrechar o expandir las consultas profundizando o no en los diferentes niveles jerárquicos, gracias de nuevo a las características intrínsecas de las ontologías.
Otra característica de las ontologías que se puede explotar es la capacidad de éstas para definir sin ambigüedad los conceptos. Gracias a ésta, un concepto será definido de forma unívoca, de tal forma que aunque por ejemplo, distintas organizaciones determinen diferentes significados para un mismo término, el receptor de la información siempre obtendrá el significado correcto correspondiente. Por ejemplo, el uso de colores para identificar el nivel de servicio se ha convertido en un problema a resolver en este ámbito de información ya que las diferentes administraciones de tráfico no se ponen de acuerdo en los colores que definen cada estado de tráfico. Sin embargo, mediante el uso de la semántica, podemos especificar el significado dado para cada color por cada una de las administraciones, de manera que cualquier usuario obtendrá inequívocamente cuál es el estado real de la red. Si la incidencia a la que hacíamos referencia anteriormente posee un nivel de servicio "amarillo" y esta información es aportada por la administración española, el sistema podrá inferir la siguiente información:

"Velocidad baja condicionada por el trafico. Estado de precaución. Paradas esporádicas", sin la necesidad de conocer este conocimiento y sin que éste se haya dado de forma explícita.
Para obtener todo este conocimiento adicional se debe hacer uso de relaciones entre diferentes sistemas terminológicos o lo que es lo mismo relaciones entre distintos subdominios y aplicar los diferentes métodos de inferencia que permita nuestro sistema.

Por otra parte, el uso de agentes software y reglas puede hacer todavía más eficiente la utilización de ontologías en tráfico:

Pensemos en la declaración de un axioma[27] del tipo "Si se trata de camiones que circulen en el tramo de peaje comprendidos entre la ciudad de Castellón y Oropesa durante el mes de agosto, no tendrán que pagar peaje". Los agentes, utilizando el conocimiento representado en los conceptos, sus relaciones y utilizando el axioma, podrían "aconsejarnos" sobre el uso de este tramo en lugar de hacer uso de carreteras libres de peaje pero más lentas.

Consideremos ahora la siguiente regla: "Si durante un intervalo de tiempo dado, acontece un cierto numero elevado de incidencias en el tramo comprendido entre dos puntos kilométricos de una carretera dada, entonces dicho tramo es un punto negro". De la misma forma que en el ejemplo anterior un agente podrá deducir, siguiendo esta regla, lo siguiente: "Un tramo de una carretera ha superado ese límite dado y por tanto dicho tramo es un punto negro".

6.4 INFRAESTRUCTURA CONCEPTUAL

A continuación se describe como han sido tratados los requisitos conceptuales y tecnológicos que debe cumplir el esquema de representación necesario y las aplicaciones informáticas que lo exploten para proporcionar servicios inteligentes.

La construcción del modelo consiste en tres fases:

1. Adquisición del conocimiento: identificación de las clases o términos básicos y sus propiedades,
2. definición: identificación de las relaciones entre las clases y,
3. especificación de restricciones: identificación de las restricciones que limitarán la manera en la que las descripciones pueden ser formadas.

Este proceso será iterativo, de manera que una vez que las definiciones han sido especificadas, sucesivos refinamientos darán como resultado definiciones mucho más elaboradas. En este proceso, es de vital importancia el uso de razonadores como FaCT o RACER, los cuales asisten en la organización del modelo y comprueban su consistencia.

La información y conocimiento relacionados con el tráfico vial se refieren a determinados conceptos: estado actual de la red de carreteras, disposiciones o normas, restricciones, rutas, meteorología etc.

[27] Axioma es una regla que se aplica sobre un domino específico. Los axiomas se utilizan para crear una serie de reglas sobre una ontología que se han de cumplir para que ésta sea consistente.

Estos conceptos, sus propiedades, las relaciones que existen entre los mismos y los términos específicos que se emplean para designarlos tienen una gran importancia puesto que proporcionan una infraestructura básica que permite organizar y conectar los elementos de información que constituyen cualquier especificación.
A partir de este análisis surge entonces el interrogante: ¿Qué vocabularios comunes pueden ser definidos para representar información de tráfico en lo relacionado con incidencias, medidas, previsiones etc.?

Tras una revisión bibliográfica, no se encontraron vocabularios que incluyeran elementos semánticos para información se tráfico vial. Surgió por tanto la necesidad de crear esta ontología y probarla mediante la construcción de diferentes prototipos.

Para abordar el problema del desarrollo de una ontología de tráfico vial se ha planteado la definición de subdominios que se relacionarán entre sí. En este sentido los subdominios de relevancia que han sido definidos son los siguientes:

- Clasificación de Vías (Autopista Libre, Autopista de Peaje, Conexión, Acceso, Ronda etc.)
- Clasificación de Vehículos (Ciclomotor, Turismo, Camión etc.)
- Localización (Área, Punto, Tramo, Métrica, Sentido etc.)
- Geografía (Poblaciones, Países etc.)
- Sucesos (Accidentes, Incidentes, Medidas etc.)
- Personas (Peatón, Conductor, Titular de Vehículo etc.)
- Rutas (Ruta Urbana, Ruta Interurbana etc.)

A su vez hay que destacar la reutilización de vocabularios ya definidos como FOAF y GEO y la ampliación de otros como Time.

Una vez construidas las diferentes subontologías de tráfico se fusionan en una ontología de conceptos de tráfico llamada (Traffic Ontology) que es la que reúne todos los conceptos principales. Esta ontología solo hace referencias a los principales conceptos definidos en el resto de subontologías y al esqueleto de las taxonomías, y por tanto no añade ningún tipo de conocimiento o valor semántico a éstos (ver figura 6.1).

La especificación completa de cada uno de los subdominios puede ser encontrada en la URL http://robotica.uv.es/tesis/javi/ontologias.html y en el CD-ROM que acompaña esta memoria.

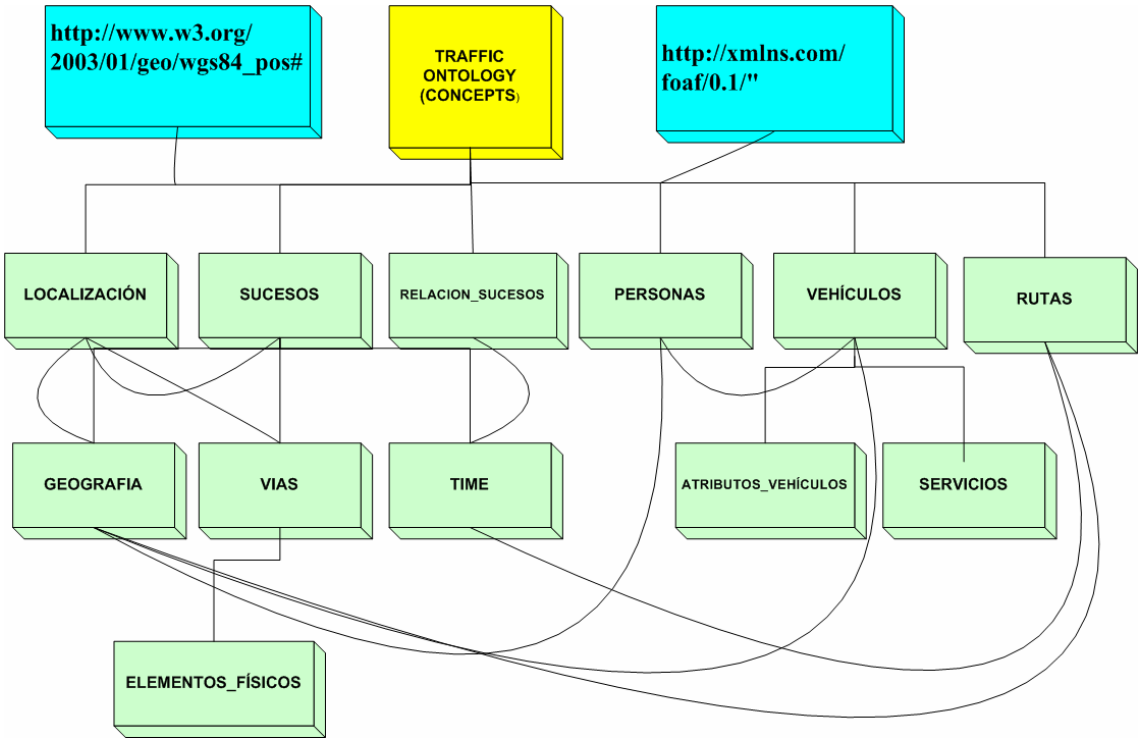


Figura 6.1 Relaciones entre los diferentes subdominios

En la tarea de desarrollar ontologías en el ámbito de información de tráfico, y más concretamente en el marco de los servicios, se especificaron éstas bajo distinto grado de generalidad. De esta forma, podemos encontrar en un nivel más general los conceptos específicos del dominio (mayor aplicación), como en el caso en el que nos ocupa, la información y conocimiento contenidos en cualquier servicio de información de tráfico. Por ejemplo, y tal como se puede apreciar en la figura 6.2, un servicio de información de incidencias, cuya ontología incluirá detalles como tipo de la incidencia, hora y fecha de ocurrencia, nivel de servicio derivado, localización etc. Por el contrario, en un nivel más abstracto, se especificaron ontologías de validez general en el contexto de tráfico vial, y por tanto no ligadas a ningún dominio específico (mayor reutilización). A este nivel corresponden las ontologías relacionadas con el instante de tiempo (día, mes, año, duración), ontologías sobre geografía, clasificación de vías etc. Por ejemplo algunos conceptos especificados en la ontología de clasificación de vías, como el término “vía”, son usados por servicios correspondientes a categorías diferentes, como pueda ser un servicio de incidencias (localización) o por servicios de cálculos de itinerarios (como parte de su ruta).

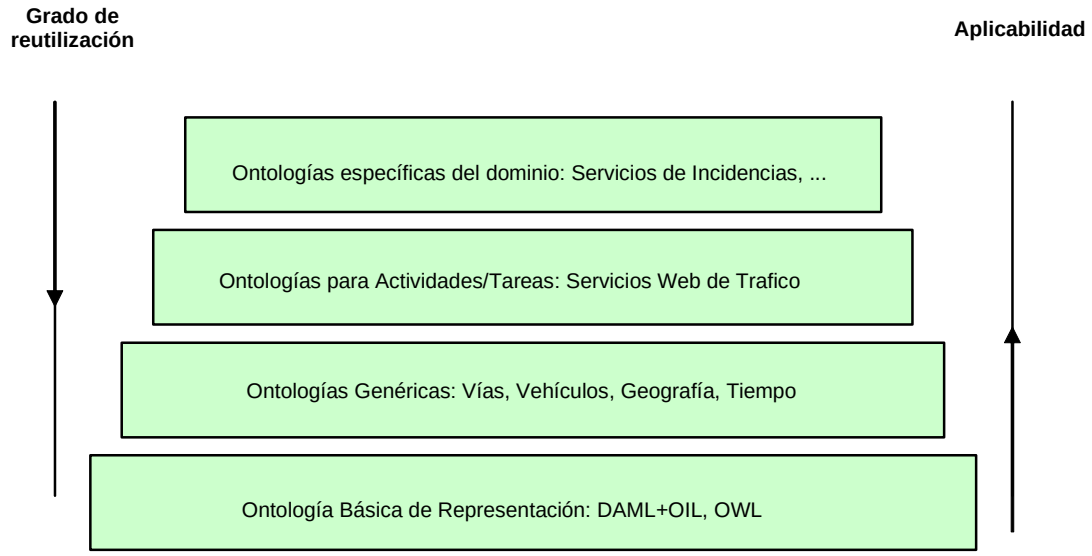


Figura 6.2 Infraestructura conceptual.

El desarrollo de la infraestructura conceptual que necesitamos sería una empresa titánica si tuviéramos que partir desde cero. Sin embargo, existen algunas iniciativas importantes desde hace unos pocos años encaminadas a facilitar estas tareas de desarrollo. En estas iniciativas se han creado bibliotecas de ontologías que están disponibles y que son útiles para desarrollar nuevas ontologías. El conocimiento ya adquirido, conceptualizado y expresado en un lenguaje formal contenido en las ontologías de estas bibliotecas, puede reutilizarse para construir una nueva ontología, eliminando, añadiendo y/o modificando los conceptos que sea preciso (ver aspectos de ontología temporal en sección 6.7.2.3).

6.5. MODELADO DE LA INFORMACIÓN DE TRÁFICO.

6.5.1 Metodologías y métodos existentes

Como ya vimos en el capítulo 4 (sección 4.3.1.5), existe un gran número de metodologías y métodos que detallan la manera de desarrollar una ontología como Uschold & King [Usc95], Kactus [Ber96], Sensus [Swa97], Cyc [Len90], On-to-Knowledge (OTK) [Sur04] y Methontology [Gom96], así como guías de desarrollo de ontologías [Fri01], [Rec02].

Las metodologías existentes y las experiencias de desarrollo prácticas comparten algunos pasos en común, como inicializar la construcción identificando el propósito y alcance de la ontología y la necesidad para un determinado dominio de adquisición de conocimiento. Sin embargo, difieren en el enfoque y el resto de pasos llevados a cabo [Sur04].

Como veremos más adelante, debido a la heterogeneidad de las fuentes de información (documentos de texto en diferentes formatos y aquéllas correspondientes al análisis y

diseño de modelos de datos), se plantean dos situaciones muy diferentes pero complementarias:

- El paso de un modelo semántico de datos a un modelo formalizado, y
- la clasificación de ciertos elementos en el modelo, que extenderán de manera considerable el conocimiento de éste mediante taxonomías (clasificaciones de vehículos y vías etc.).

El modelo final, resultado de ambos, está completamente integrado y ha sido formalizado a través de DL, y sus correspondientes lenguajes de marcado ontológico como DAML + OIL y OWL.

6.5.2 De ER a modelo semántico formal

En la sección 4.3.3 del capítulo 4, se detalló la posibilidad de obtener beneficios de la aplicación de modelos semánticos formales, a modelos de datos. Tomando como base un modelo conceptual, es conveniente su formalización mediante la aplicación de un modelo formal semántico, el cual será implementado en el lenguaje de especificación de ontologías escogido (ver figura 6.3).

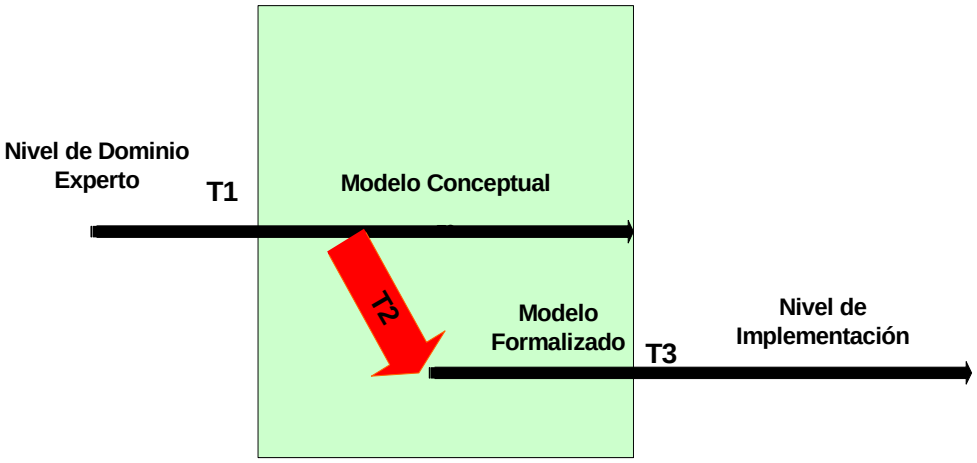


Figura 6.3 Modelo de proceso esencial en el desarrollo de una ontología. [Gom04]

En el modelo formalizado, de especial interés son las DL, las cuales soportan la clasificación automática de los conceptos con respecto a otros, y de ese modo revelan generalizaciones que pueden no haber sido reconocidas por el modelador.

6.5.2.1 Propuesta de un método para obtener un modelo semántico formal a partir de un modelo de ER.

Los pasos que componen las metodologías para desarrollo de ontologías identifican generalmente el método de trabajo partiendo desde cero, es decir, sin la previa

existencia de ontologías. En METHONTOLOGY [Gom04], los autores plantean en su proceso de modelización conceptual, la necesidad de organizar y convertir el conocimento informal de un dominio en una especificación semi-formal usando un conjunto de lo que los autores denominan representaciones intermedias (IR), basadas principalmente en notaciones gráficas. Por otra parte, en el capítulo 4 de la presente memoria se describió el proceso de reingeniería inversa propuesto por METHONTOLOGY, el cual toma como origen del conocimiento implementaciones de ontologías para luego a partir de ellas obtener esquemas conceptuales, los cuales serán posteriormente extendidos o modificados.

Sin embargo, podemos encontrarnos en una situación intermedia entre las anteriormente expuestas, ya que muchos de los sistemas de información existentes en la actualidad han sido o están siendo estudiados desde la perspectiva de modelos de datos semánticos, y el uso de esta información se convierte en una valiosa fuente de cara a su proceso de formalización. Por esta razón, a continuación se propone una extensión a las metodologías previas para detallar la forma en que puede ser integrado conocimiento existente en el análisis y diseño de bases de datos. Esta extensión metodológica se aplicará por tanto, a aquellos sistemas en los cuales ya existan estudios previos basados en modelos semánticos de datos. En la presente aproximación, se integran por tanto, diferentes aspectos de las metodologías existentes, y de lecciones aprendidas de la experiencia.

Tal y como podemos apreciar en la figura 6.3, se trata de “traducir” modelos semánticos de datos en modelos semánticos formales y a su vez, siempre que sea posible, proceder a su mejora.

6.5.2.2 Fases en el método propuesto.

Distinguimos 8 fases claramente diferenciadas.

1. Adecuación de propósito y alcance.
2. Normalización y modularidad. Creación de subdominios.
3. Reutilización de ontologías
4. Traducción básica
5. Refinamiento
6. Extensión del conocimiento: Adición de instancias.
7. Pruebas o Evaluación.
8. Documentación.

1) Adecuación de propósito y alcance.
Adecuación del propósito y alcance dados para el desarrollo del modelo de datos a nuestro modelo semántico formal. Análisis de nuevos requerimientos que el nuevo modelo deberá cumplir.

2) Normalización y modularidad. Creación de subdominios.
Análisis del esquema conceptual y propuesta de creación de diferentes módulos o subdominios:

a) Identificación de todas las entidades existentes en el modelo ER, con sus atributos y con las relaciones que posean con el resto de entidades del modelo.
b) En base a la anterior clasificación, se construye una tabla, en la que serán insertadas las entidades mediante un orden de prioridad establecido por el factor de número de relaciones, y en caso de empate por número de atributos. Esta tabla deberá mostrar los enlaces entre cada una de las entidades.
c) Extracción de la tabla la entidad con mayor prioridad y que se convertirá en esta primera extracción, en la entidad central del modelo, la cual constituirá la base para el desarrollo del principal dominio del modelo, a partir del cual girará el resto.
d) Intentar establecer que otras entidades podrían formar parte del dominio en cuestión y extraerlas de la tabla. Para ello se escogerán de entre aquéllas que están relacionados con la entidad escogida en el paso c), las que por sus características y aplicando lógica común deban de permanecer juntas en el mismo dominio.
e) Volver al paso c) y extraer de la tabla la siguiente entidad por orden de prioridad para la creación del resto de subdominios periféricos, hasta que la tabla quede vacía.

El propósito de la normalización es la sencillez la cual permitirá la reutilización, mantenimiento y evolución. Ante la necesidad de evitar descripciones complejas, esta fase de normalización consistirá en crear módulos simples que puedan ser combinados para alcanzar las descripciones complejas en las cuales estábamos interesados. Se establecen por tanto las razones para clasificar y se impone una disciplina para establecer cada definición considerada relevante de forma explícita. La razón para clasificar bajo los distintos módulos debe darse de forma explícita para que el razonador pueda realizar las correctas inferencias.
Los módulos consistirán generalmente en jerarquías simples donde los conceptos primitivos generalmente tendrán un único padre. Las definiciones y descripciones permitirán enlazar las diferentes jerarquías y permitirán al razonador inferir la estructura compleja. A su vez deberá existir una estructura taxonómica que relacione los diferentes módulos, especificando las diferentes categorías [Ste03] [Rec03]. En nuestro modelo, denominadas como Traffic Ontology o Concepts Ontology. (Ver figura 6.1).
La modularidad es un aspecto muy importante a tener en cuenta, ya que mantener ontologías independientes significa que cada una puede ser modificada separadamente. Inclusive dentro de una misma ontología es sumamente conveniente modificar las jerarquías en base a sus estructuras, roles etc., de tal forma que se puedan tener diferentes tipos de clasificaciones dentro de una misma ontología (subtaxonomías).
En esta tesis, los diferentes tipos de vías especificados en la ontología de clasificación de vías, han sido clasificadas según varios criterios, por ejemplo atendiendo al criterio de situación o al criterio de destino etc.

3) Reutilización de ontologías
Una vez determinados los diferentes subdominios, considerar la reutilización de ontologías ya disponibles y públicas que se puedan tener en cuenta en cada uno de éstos. El uso de éstas puede llegar a simplificar las fases siguientes de la metodología.

Una vez finalizada la fase de normalización y analizando el posible uso de ontologías ya creadas, se pasará a la siguiente fase en la metodología, que se aplicará a cada uno de los subdomínios establecidos.

4) Traducción básica

Traducción de los diferentes elementos del esquema conceptual a modelo semántico formal.

Entidades

Transformación de las entidades en clases y/o individuales según criterio y establecimiento de su jerarquía.

- Hay dos formas de representar los elementos de una clase, como individuales o como subclases [Cec02]. El uso de clases es semánticamente más rico y hace que la ontología se pueda extender más fácilmente. Incluso aún existiendo mayor complejidad, el uso de clases es mejor salvo en los casos en donde los individuales sean necesarios como por ejemplo, al establecer miembros de una clase enumerada.

Atributos de entidades

Transformación de los atributos de las entidades que aparecen en el modelo semántico de datos en propiedades, teniendo en cuenta que cada una de estas entidades posee una serie de atributos que a menudo solo tienen utilidad en el modelo de datos semántico, debido a las características intrínsecas de los modelos de datos relacionales. Por ejemplo, los atributos que hacen referencia a las versiones del Diccionario de Datos. Estos atributos quedarán por tanto excluidos de la especificación formal. Pasos a seguir:

a. Estas propiedades podrán ser dos tipos: propiedades objeto (ObjectProperty) o propiedades de tipos de datos (DatatypeProperty), lo cual determinará el rango de valores que puede tomar.
    i. Propiedades objeto son usadas para relacionar un recurso a otro recurso.
    ii. Propiedades de tipos de datos únicamente relacionan un recurso a un *rdfs:literal* o a un tipo de datos perteneciente a *XML Schema*.

b. Se estudiará según criterios dados la conveniencia de restringir estas propiedades globalmente: Rango y Dominio.
    i. Más tarde en el proceso, será posible restringir el rango especificado en una propiedad de una clase padre, cuando dicha propiedad afecta a clases descendientes, siempre y cuando ese nuevo conjunto de valores que la propiedad pueda tomar sea un subconjunto del rango de valores original de la clase ascendente. Esto se logrará mediante operadores de cuantificación generalmente.

c. Definición de características de las propiedades: Transitividad, Inversa, Simétrica.
    i. Hay que tener en cuenta que las características de transitividad o simétrica solo podrán ser aplicadas en propiedades de tipo objeto (relacionan dos recursos).
d. Inclusión de dicha propiedad en una jerarquía de propiedades (elección de superpropiedad(es)) que ayudará en la aplicación de posibles métodos de inferencia.

Relaciones

Tratamiento de las relaciones que aparecen en el esquema, y distinción según el grado de la relación y la posesión o no de atributos propios, en:

a) Relaciones Binarias, las cuales directamente se podrán traducir al lenguaje de especificación formal mediante propiedades de tipo objeto.
b) Relaciones n-arias, cuya traducción no se podrá realizar directamente.
c) Relaciones con atributos propios.

a) Para el primer caso, serán introducidas como propiedades, al igual que fueron introducidos los atributos, con la salvedad de que ahora solo pueden ser de tipo objeto, puesto que deben de relacionar dos recursos o entidades. Se tomarán los mismos criterios que los tomados para especificar los atributos, en cuanto a las características de dichas propiedades.

b) Debido a que en DL tradicional (y por tanto en los lenguajes formales basados en ella) únicamente son considerados relaciones unarias o binarias, se tomará en consideración el método propuesto por Calvanese et al. [Sat03], el cual consiste en "reificar" las relaciones, generalmente mediante la traducción de cada relación en un concepto cuyas instancias representan las tuplas de la relación. Mayor detalle de la definición de este tipo de relaciones y su uso mediante individuales puede ser encontrado en el informe técnico de Alan Rector en W3C [Rec04a].

c) Para el caso de relaciones con atributos, se tomará de nuevo en consideración si dicho atributo (propiedad), puede ser asumido por alguna de las entidades relacionadas, de lo contrario se procederá a crear una nueva clase que establecerá el rango de una nueva relación (propiedad objeto), tomando como base este nuevo atributo. (Ver sección 6.6.2.5.1)

Grados en las relaciones

a) Observación del modelo para determinar los grados de cardinalidad existentes en cada una de las relaciones. La traducción al lenguaje formal se hará mediante el uso de restricciones de cardinalidad a cada una de las clases afectadas o en su lugar restringiendo globalmente la propiedad mediante las características funcionales o inversamente funcionales.

i. Las características de Funcional o Inversamente Funcional pueden ser aplicadas a cualquier tipo de propiedad (objeto y datatype).

b) Especificación de las restricciones locales a cada una de las clases mediante el uso de operadores de cuantificación existencial y/o universal (ver documento "**Detalles de la especificación en las ontologías creadas**", en la URL http://robotica.uv.es/tesis/javi/document.html y en el CD-ROM que acompaña esta memoria).

Definiciones completas/primitivas. Uso de axiomas.

a) Determinación de que clases deberían formar parte en una definición completa o cuales simplemente deberían ser simplemente clases primitivas. (ver documento "**Detalles de la especificación en las ontologías creadas**", en la URL http://robotica.uv.es/tesis/javi/document.html y en el CD-ROM que acompaña esta memoria).

b) Teniendo en cuenta el punto anterior, para aquella parte de la especificación de una clase que no deba formar parte de una definición (completa o parcial), se hará uso de axiomas de cubrimiento, disyunción etc. (ver documento "**Detalles de la especificación en las ontologías creadas**", en la URL http://robotica.uv.es/tesis/javi/document.html y en el CD-ROM que acompaña esta memoria). Los axiomas proveerán información adicional sobre las clases.

5) Refinamiento

a) Revisión de entidades y de propiedades, ya que quizás sea necesario una mayor expresividad o base de conocimiento mucho más elaborada, de tal forma que algunas de estas propiedades o entidades necesiten de la elaboración de nuevos dominios específicos no contemplados en principio en el modelo semántico de datos. Pensemos por ejemplo en atributos del modelo de datos los cuales toman valores de su diccionario de datos. En estos casos puede ser conveniente la introducción de nuevos dominios correspondientes en principio a simples taxonomías que más tarde pueden convertirse en complejas ontologías con la adición de nueva información proveniente de otras fuentes, distintas al diccionario. Por ejemplo, de nuevo en el desarrollo de la ontología "Vías", ciertos atributos como "carretera" de las entidades Punto y Tramo del modelo de datos semántico o en la ontología de Vehículos (atributo "tipo" de la entidad Vehículo) o Geografía (atributos provincia, población etc. de las entidades Área y Punto), fueron convertidos a ontologías.
b) Una vez, establecida la primera "traducción" se procederá a estudiar las posibles aplicaciones de nuestro modelo. Quizás tras un estudio exhaustivo de éstas se llegue a la conclusión de que el modelo formal actual no contemple ciertas propiedades o entidades que serán necesarias para un adecuado uso en dichas aplicaciones. (por ejemplo, las propiedades específicas de elementos como Previsión que simplemente formaban parte del DD del modelo de datos original).

6) Extensión del conocimiento: *Adición de instancias.*

Adición de conocimiento extensional es decir aporte de instancias o individuales. La adición no tiene porque ser en el mismo fichero físico que en el que resida el sistema terminológico (términos y relaciones), aunque tendrán que estar ligados.

A partir de esta fase se considerará el sistema globalmente.

7) Pruebas o Evaluación: Pruebas de requerimientos sobre la base de conocimiento que muestren la eficacia de éste. En este punto se tratará de comprobar si la base de conocimiento es capaz de satisfacer todos los requisitos especificados en la fase inicial de propósitos y alcance. Es decir, responder a las cuestiones que inicialmente se abordaron. El uso de un sistema de inferencia, posiblemente mediante la elaboración de consultas permitirá resolver esta cuestión.

8) Documentación
La especificación ontológica debe ser ampliamente documentada, de tal forma que cualquiera puede entender fácilmente su composición y estructura. De esta forma las tareas de mantenimiento y escalabilidad del modelo podrán ser llevadas de forma más adecuada. En esta fase se considera útil aunque no necesario especificar la correspondencia entre ambos modelos.

A partir del paso 3 será útil comprobar en cada paso la consistencia del modelo e inclusive una vez determinadas aquellas clases que deban de poseer una definición completa, ayudarse del razonador para establecer posibles clasificaciones en la jerarquía.

6.6 DESARROLLO DE LA ONTOLOGÍA.

6.6.1 Fuentes de información utilizadas

La identificación de los diferentes términos, se realizó a partir de un estudio o análisis de fuentes de tráfico heterogéneas: sitios web de administraciones, bases y modelos de datos sobre tráfico, informes sobre accidentes, planes de gestión, reglamentos generales, ficheros que sirven como entrada o salida a simuladores de tráfico etc., lo que nos ha permitido establecer que conceptos y relaciones son los apropiados para cada dominio de discurso.
Hay que hacer constar tal y como se mostrará a continuación que el origen principal de las ontologías creadas ha sido el modelo conceptual semántico y diccionario de datos, correspondiente a la reingeniería inversa del actual Concentrador de Información de Tráfico (de aquí en adelante CIT) de la DGT. Partiendo del modelo conceptual allí creado se ha pretendido establecer lazos o correspondencias entre el modelo conceptual y el modelo ontológico, para explotar todas las características que este último puede aportarnos. Para poder aprovechar toda la expresividad que nos permite este tipo de modelo, se ha recurrido a una serie de fuentes ajenas al diseño conceptual que nos permitirán obtener bases de conocimiento mucho más elaboradas y por tanto la aplicación de métodos de inferencia sobre éstas.

En el desarrollo de esta infraestructura ontológica, se tuvieron en consideración dos aspectos de suma importancia:

En primer lugar, la clasificación taxonómica de los conceptos y sus relaciones (conocimiento intensional) y en segundo lugar la instanciación de sus términos (conocimiento extensional). La distinción de estos dos aspectos es trascendental porque en la mayoría de los casos, el origen o fuente de información varió según este tratamiento.

Como ejemplos específicos, citar que para la realización de la ontología sobre la clasificación de la red de carreteras, se tomaron en consideración las siguientes fuentes:

Para la especificación intensional, la principal fuente fue el "Reglamento General de Carreteras" donde queda establecida una clasificación de éstas y sus elementos, mientras que para la especificación del conocimiento extensional dos fuentes fueron principalmente utilizadas: El Real Decreto 1231/03 de 26 de septiembre de 2003, por el que se modificó la nomenclatura y el catálogo de las autopistas y autovías de la red de carreteras del estado así como también el diccionario de datos provisto para este tipo de información en el actual estudio sobre la reingeniería del CIT.

De igual manera para el desarrollo de la ontología sobre clasificación de vehículos se tomó como base el "Reglamento General de Vehículos, Anexo II Definiciones y categorías de vehículos".
Citaremos de nuevo, tal y como ya sido abordado en el capítulo 3, la documentación proveniente del grupo de trabajo ISO TC204 WG1 en el cual ejerzo tareas de supervisión, y concretamente la definición del conjunto de Grupos de Servicio ITS y Categorías, actualmente en vías de convertirse en un estándar y originalmente referenciada en TR 14813-1 [ISO04]. Esta fuente de información ha sido providencial para la especificación taxonómica de los servicios de tráfico, y su uso en la búsqueda de perfiles de éstos, tal y como abordaremos más tarde en el capítulo 8. De igual forma cabe destacar la reutilización de ontologías ya creadas como ontologías de especificación temporal que a su vez han sido ampliadas o modificadas para un diseño más acorde con nuestro sistema.

Analizando estas fuentes nos damos cuenta que podemos establecer una taxonomía de los diferentes elementos que lo constituyen así como sus relaciones e instanciación, pero que hay que tomar una serie de decisiones en cuanto su desarrollo.

6.6.2 Diferentes aspectos a tener en cuenta en la especificación

Para más detalles sobre la implementación mediante lenguajes semánticos referirse al documento "**Detalles de especificación en las ontologías creadas**" que se encuentra en la URL http://www.robotica.uv.es/tesis/javi/document.html y en el CD-ROM que acompaña esta memoria.

6.6.2.1 Características de los conceptos o términos.

En la enumeración de términos han sido determinados todos aquellos conceptos que en un principio se consideraban importantes. Posteriormente fueron tomadas una serie de decisiones en torno a ellos que han caracterizado nuestra ontología.

- **¿Se trata de conceptos asumidos como clases denominadas primitivas o definidas?**

Teniendo en cuenta la distinción entre clases o conceptos primitivos y clases definidas [Rec02] y [Rob02], ¿de qué modo especificaremos cada uno de nuestros elementos? Para responder a esta pregunta, deberemos tener en cuenta una serie de consideraciones:

- El uso de *razonadores lógicos* permitirá clasificar de forma automática clases primitivas bajo otras clases definidas, pero no será posible clasificar clases primitivas como subclases de otras clases primitivas[28].
- Las definiciones son muy costosas computacionalmente ya que el razonador tiene que considerar que cosas deben de ser clasificados bajo ellas al igual que el lugar donde ellas deberían ser clasificadas.
- Se deben especificar clases primitivas cuando existan multitud de definiciones sobre el mismo concepto, que harán que el uso de una definición sea completamente inadecuada para una definición completa y además cuando las características individuales casi nunca sean usadas, por lo que la ontología nunca necesitará reconocer el concepto dado.

Por tanto y llegando a un punto de conveniencia en nuestros objetivos se establecen los diferentes conceptos y relaciones en entre ellos. Debido al gran coste computacional que suponen las clases definidas, se han considerado la mayoría de conceptos como primitivos y solo en algunos casos se ha especificado una definición completa.

- **Distinción entre *conceptos independientes* y *modificadores o valor de refinamiento. Uso de axiomas de disyunción.***

En nuestra ontología podemos distinguir los denominados conceptos *independientes* o *significantes* por si mismos, de los conceptos denominados de *refinamiento, modificadores o características* como lo son la severidad, dimensión, rango etc. tal y como aparece en [Rec02].
En el desarrollo de la ontología, se han utilizado individuales disjuntos para describir ciertas clases mediante restricciones sobre propiedades cuyo rango sólo podrá pertenecer a un conjunto enumerado de valores determinado por esos individuales (*value sets*). Ese rango por tanto, estará determinado por una clase o concepto de refinamiento de tal forma que el concepto solo podrá tener valores pertenecientes a ese rango o dicho de otra forma valores correspondientes a individuales que forman la clase enumerada (concepto característica). Sin embargo, el problema consiste en que muchos

[28] En OWL o DAML+OIL definir un concepto como una clase primitiva es decir que nosotros no podemos (o elegimos no hacer) darle una definición completa sino que únicamente elegimos darle una descripción por la cual pueda ser clasificado bajo otras clases definidas.

razonadores o repositorios tienen limitaciones en cuanto al tratamiento de enumeraciones (operador "one-of") por lo que se hace indispensable encontrar otras formas alternativas para el conocimiento expuesto anteriormente. La solución aportada en esta tesis y que quedó reafirmada en agosto de 2004 por el informe técnico de Alan Rector en W3C [Rec04b], radica en el uso de valores como subclases que particionan un concepto característica (value partitions). El lector podrá encontrar información detallada de este tipo de problemas en el documento "**Problemática en la representación de valores de refinamiento mediante el lenguaje semántico**", accesible mediante la URL http://robotica.uv.es/tesis/javi/problemas.html y a través del CD-ROM que acompaña esta memoria.

6.6.2.2 Cubrimiento de clases (Covering).

Tomando como punto de referencia, el subdominio de Localización, hay que destacar que los servicios de información de tráfico son servicios que tratan con las condiciones o estado de carreteras, información de tráfico e información sobre accidentes. El método habitual, consiste en la creación de mensajes de tráfico por parte de los Centros de Información de Tráfico (CIT), para posteriormente distribuirlos. Generalmente estos mensajes consisten en dos partes bien diferenciadas: la descripción del evento y la descripción de la localización de éste. Atendiendo a la especificación de esta última parte, y basándonos en el modelo de datos preestablecido podemos hacer constar que la localización de un evento puede estar definida de tres formas distintas (figura 6.4):

- Puntos (Puntos kilométricos o PKs, coordenadas geográficas etc.)
- Tramos (Tramos de carreteras formados por puntos)
- Áreas (Formadas por puntos o tramos)

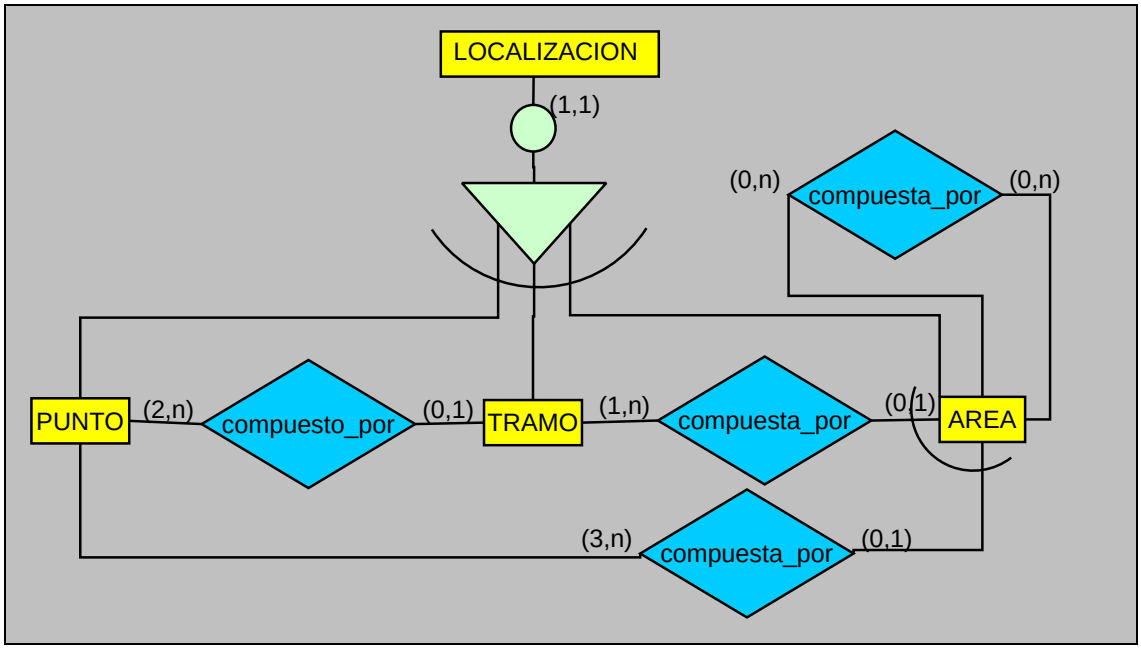


Figura 6.4 Esquema EER: Localización

Considerando únicamente la descripción del concepto Localización, se debe establecer que todos los subtipos (Punto, Tramo, Área) de "Localizacion" completan su definición de tal forma que el concepto estará completamente descrito. Además se deberá añadir en nuestra ontología el hecho de que estos conceptos o subclases son disjuntos. Para poder hacer lo anterior se establecen tres fases:

a) En la definición de clase, se especifica que "Localizacion" es subclase de cada uno de sus subtipos (denominados también valores de refinamiento).

b) Con axiomas de tipo subclase (covering), establecemos que cada uno de los subtipos es a su vez subclase del concepto al que cubren (Localizacion).

c) Por último, especificamos la disyunción entre los diferentes valores de refinamiento mediante un axioma disyuntivo.

Haciendo uso de Lógica Descriptiva:

a) $Localizacion \doteq Punto \cup Tramo \cup Area$
b) $\bot \doteq Punto \cap Tramo \cap Area$

6.6.2.3 Múltiples rangos y dominios y característica de alcance global

Si observamos el dominio de Sucesos (figura 6.5), nos damos cuenta que ciertos conceptos o clases pueden contener múltiples rangos, y una vez más en este caso se hace patente la necesidad de uso del lenguaje elegido, debido a la limitación de RDF(S) para poder expresar este conocimiento (ver sección 4.5.2.1 para más detalle).
En un Suceso, no solo están implicados vehículos sino también personas, por lo cual expresar esto formalmente, consiste en establecer dos rangos para una propiedad a la que denominaremos "implica" y cuyo dominio es el concepto o clase "Suceso".

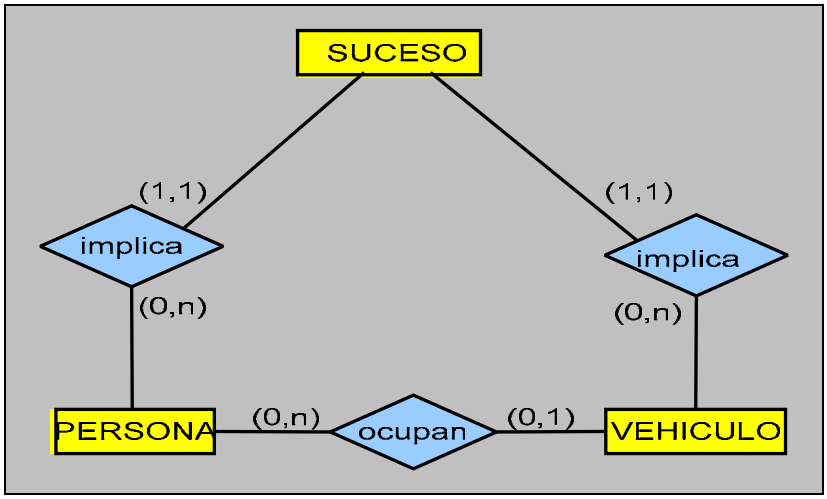


Figura 6.5 Múltiples rangos y dominios

Formalmente:

a) $implica \subseteq \Delta \times Persona$
b) $implica \subseteq \Delta \times Vehiculo$
c) $Vehiculo \subseteq (= 1\ implica^{-}.Suceso)$
d) $Persona \subseteq (= 1\ implica^{-}.Suceso)$

Sin embargo, el uso de rangos en la definición de propiedades tiene un efecto colateral muy importante y es el hecho de que estamos limitando la propiedad afectada ("implica") en un ámbito global, ya que siempre deberá de tener valores pertenecientes a las clases especificadas ("Vehículo" y "Persona").

6.6.2.4 Alcance local. Uso de restricciones sobre propiedades. Cuantificador existencial y cuantificador universal

En lugar de decir que el rango de una propiedad debe tener siempre valores pertenecientes a una clase determinada, podemos establecer la restricción solo para valores de una clase particular. Esta restricción de tipo local nos permitirá hacer uso de las propiedades en cualquier otra parte con restricciones diferentes. En estas restricciones podemos hacer uso de los operadores de cuantificación existencial y universal. A su vez, en algunos casos podremos hacer uso de las denominadas restricciones cualificadas, que como ya vimos al hablar de lenguajes para el desarrollo de ontologías (sección 4.5.2) no están permitidas en OWL, pero sí en DAML+OIL.

6.6.2.5 Agregación, Especialización/Generalización

6.6.2.5.1 Agregación

En el Modelo de Datos, la *agregación* es una abstracción que permite representar entidades compuestas que se obtienen por unión de otras más simples. A la entidad compuesta nos referimos como el *todo,* mientras que los componentes son las *partes.*
En aproximaciones orientadas a objetos, la agregación es una relación de tipo estándar. El operador agregación combina clases componentes, y genera una nueva clase con propiedades emergentes (propias de la clase agregada como nueva clase) y heredadas (de las participantes en la agregación).
En nuestro esquema conceptual inicial aparecen algunas relaciones que poseen atributos propios, como la relación "compuesto_por" entre las entidades Escenario y Suceso (ver figura 6.6). Para representar este tipo de relaciones se hará uso de la técnica que aparece en [Tab02]. (Ver documento "**Detalles de la especificación en las ontologías creadas**", en la URL http://robotica.uv.es/tesis/javi/document.html y en el CD-ROM que acompaña esta memoria).

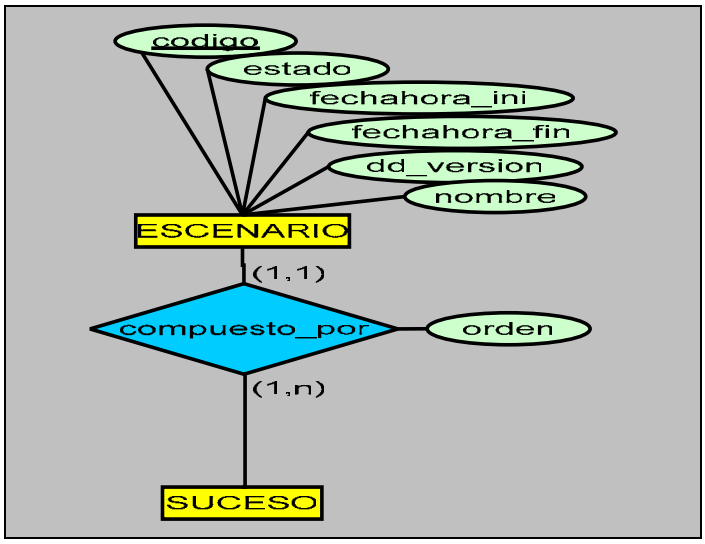


Figura 6.6 Relación entre entidades Escenario y Suceso

6.6.2.5.2 Especialización / Generalización, Herencia

Hay que tener en cuenta que uno de los principales mecanismos para inferir conocimiento basado en DL, es la herencia.

Los modelos semánticos han ido incluyendo en sus propuestas progresivamente los mecanismos de herencia / especialización en forma de relaciones IS-A, hasta tal punto que no se concibe en la actualidad modelo semántico alguno que no soporte tal expresividad. En los Modelos Orientados a Objetos, la generalización se usa para elaborar una vista uniforme de diferentes clases base. En la especialización se definen un número de subclases para una clase base dada mientras que en la generalización se define un superclase común a partir de un conjunto de subclases [Pas92].
Bajo el punto de vista del Modelo de Datos, esta clase de relación tiene la característica de que todo elemento de un subtipo es también un elemento del supertipo, aunque no sucede lo contrario. La aparición de estas jerarquías en dicho modelo puede surgir de dos formas distintas:

- *Generalizació*n: Se observa que dos o más entidades comparten varios atributos y/o relaciones, de donde se deduce la existencia de una entidad de nivel superior (supertipo) que contiene los atributos y las relaciones comunes a todos los subtipos.
- *Especializació*n: Se observa que una entidad tiene ciertos atributos y/o relaciones que tiene sentido para unos elementos pero no para otros, por lo que es conveniente definir uno o varios subtipos que contengan estos atributos y/o relaciones específicas, dejando en el supertipo los que son comunes.

Por tanto, si nos movemos de los subtipos hacia el supertipo, se trata de una generalización; mientras que si primero identificamos el supertipo y, a partir de él, llegamos a los subtipos, se trata de una especialización.

OWL permite establecer jerarquías de generalización/especialización. Una clase puede especificar su superclase(s) haciendo uso de *subClassOf* o *intersectionOf*, y puede especificar la subclase(s) con *unionOf*. Por ejemplo, nosotros podemos definir "CiclomotorTresRuedas" como *subClassOf* de "Ciclomotor", de modo que cada "CiclomotorTresRuedas" es también un "Ciclomotor".
También podemos establecer (por su definición) "VehiculoArticulado" como la *intersectionOf* de dos clases "VehiculoMotor" y "Semirremolque". De este modo, cada "VehiculoArticulado" es también un "VehiculoMotor" y un "Semirremolque".

Por lo tanto haciendo que una clase X sea una subclase de Y, podremos afirmar que todos los miembros de la clase X pertenecerán también a la clase Y.

Atendiendo al primero de los ejemplos, cualquiera de las siguientes afirmaciones es completamente válida:

- Ciclomotores de tres ruedas *es un tipo de* ciclomotor,
- *si* algo es un ciclomotores de tres ruedas *entonces* es un ciclomotor,
- ser un ciclomotores de tres ruedas *implica* ser ciclomotor,
- ciclomotores de tres ruedas *es una subclase de* ciclomotor,
- ciclomotor *incluye* ciclomotores de tres ruedas,
- ciclomotores de tres ruedas → ciclomotor.

Ser un *tipo de*, o *subclase de* alguna cosa tiene un específico y fuerte significado en DAML + OIL u OWL:
"Todos los miembros de un subclase, sin excepción, son miembros de la superclase" o "ser un miembro de la subclase implica ser un miembro de la superclase."
Este fuerte significado lógico de "subclase" tiene una serie de consecuencias:

- No habrá ninguna excepción
- Será posible probar cosas mediante el uso de la lógica. En particular será posible probar que una clase es subclase de otra sobre las bases de la definición y axiomas. Esto es uno de los aspectos más importantes del lenguaje.

Herencia:
Supongamos una clase hijo con sus propias restricciones sobre propiedades, como subclase de una clase padre (puede existir más de una clase padre), que a su vez posee sus propias restricciones y es hija a su vez de otra clase padre. La herencia del hijo estará compuesta tanto por las propiedades y restricciones propias de sus padres como de las que éstos también heredaron de sus respectivos padres. Por tanto la clase hijo inicial dispondrá de un conjunto de restricciones (R) formado tanto por las suyas propias, como las de su(s) padre(s) directo(s) y las de sus antecesores. Denominaremos a estas restricciones sobre las propiedades como Ri, Rp y Ra respectivamente. Por tanto podemos establecer la siguiente relación:

$$\mathbf{R = Ri + Rp + Ra}$$

Determinando la clase "VehiculoPersonasMovilidad" como subclase de "CiclomorTresRuedas", podemos afirmar todo lo anteriormente dicho para

CiclomotorTresRuedas y Ciclomotor respectivamente. Es decir: Todos los vehículos para personas de movilidad reducida son ciclomotores de tres ruedas y por lo tanto también serán Ciclomotores (VehiculoPersonasMovilidad → CiclomotorTresRuedas → Ciclomotor).

Si suponemos una restricción para esta clase: "Tener un tara (peso o masa) límite de 350 Kg.", podremos afirmar lo siguiente:

"Si alguna cosa es un vehículo de personas de movilidad reducida deberá ser un ciclomotor de tres ruedas y por tanto cumplir sus restricciones y además tener una tara de 350 Kg. como máximo".

Esta afirmación significa dos cosas:

- "Todos los *vehículos de movilidad reducida* son *ciclomotores de tres ruedas*"
- "Todos *los vehículos de movilidad reducida* tienen una *tara* inferior a 350 Kg."

Sin embargo mediante estas afirmaciones el sistema utilizado será incapaz de reconocer y clasificar otras cosas como pudieran ser tipos de vehículos de movilidad reducida u otra clase de ciclomotores de tres ruedas, si previamente las clases no han sido especificadas como clases definidas en lugar de primitivas.

La herencia a veces desencadena algunos problemas, al arrastrar hacia clases descendientes ciertas restricciones sobre propiedades que pueden dar lugar a una inconsistencia del modelo. Por ejemplo, clases descendientes que hereden restricciones incompatibles de más de un clase ascendente.
El lector podrá encontrar un ejemplo práctico de este tipo de problemas en el documento "**Problemas derivados de la herencia. Un caso práctico**", accesible mediante la URL http://robotica.uv.es/tesis/javi/problemas.html y a través del CD-ROM que acompaña esta memoria.

6.6.2.6 Uso de diferentes espacios de nombres.

Es importante destacar ciertos aspectos en el desarrollo de la ontología de tráfico. Por ejemplo, al intentar especificar determinados acontecimientos como puedan ser los relativos a una determinada incidencia ocurrida en la carretera, podemos describir con cierto detalle determinadas características de los elementos (conceptos) a describir. Para lograr este objetivo haremos uso no solo de las propiedades propias del modelo sino de *namespaces* mediante el uso de URIs que nos aportarán una información más enriquecida del elemento en cuestión. De la combinación de éstos surgirá una descripción mucho más elaborada para cada uno de los conceptos expresados en el modelo. Por ejemplo, para la descripción de una determinada persona involucrada en un accidente, podemos hacer uso de *namespaces* como FOAF [FOAF04] o DC [DC05] que nos aportarán un alto nivel de detalle en la descripción de dicha persona (Ver mi descripción o currículo en http://alba.uv.es/javi/foaf.rdf).
El uso de RDF Schema nos permite combinar objetos desde el mismo o diferentes *namespaces* usando los axiomas *subClassOf* y *subPropertyOf*, así como las restricciones globales de rango y dominio para una determinada propiedad. Es decir

podemos referirnos a clases, propiedades e incluso tipos definidos por el usuario o definidos en XML Schema haciendo uso del URI que identifica la ubicación exacta del concepto en cuestión. De la misma forma se pueden tomar distintas ontologías y conectarlas minuciosamente entre sí mediante el uso del abanico de constructores del lenguaje.
Cuando existan más de una ontología especificando distintas definiciones para un mismo concepto, a menudo es útil hacer uso de los axiomas *sameClassAs* y *samePropertyOf* de esa forma no hará falta reescribir las instancias porque se entenderá que las diferentes definiciones son válidas y representan el mismo concepto.
El uso de este tipo de axiomas puede ser utilizado en el tratamiento de ontologías equivalentes pero descritas en lenguas diferentes (ver figura 6.24).

6.6.2.7 Especificando relaciones en nuestro conocimiento base.

La especificación de relaciones se convierte a veces en una tarea no trivial. Sin embargo, DAML+OIL u OWL nos permiten en cierta manera representar un abanico muy amplio de éstas.
Deberemos distinguir dos casos diferentes: Las relaciones binarias entre objetos pertenecientes a un mismo dominio y las relaciones entre conceptos u objetos pertenecientes a los diferentes subdominios.

6.7 MODELADO FORMAL

Tomemos como base la arquitectura general de un sistema de representación de conocimiento basado en lógica descriptiva adaptado a nuestro modelo parcial (Sucesos). Ver figura 6.7:

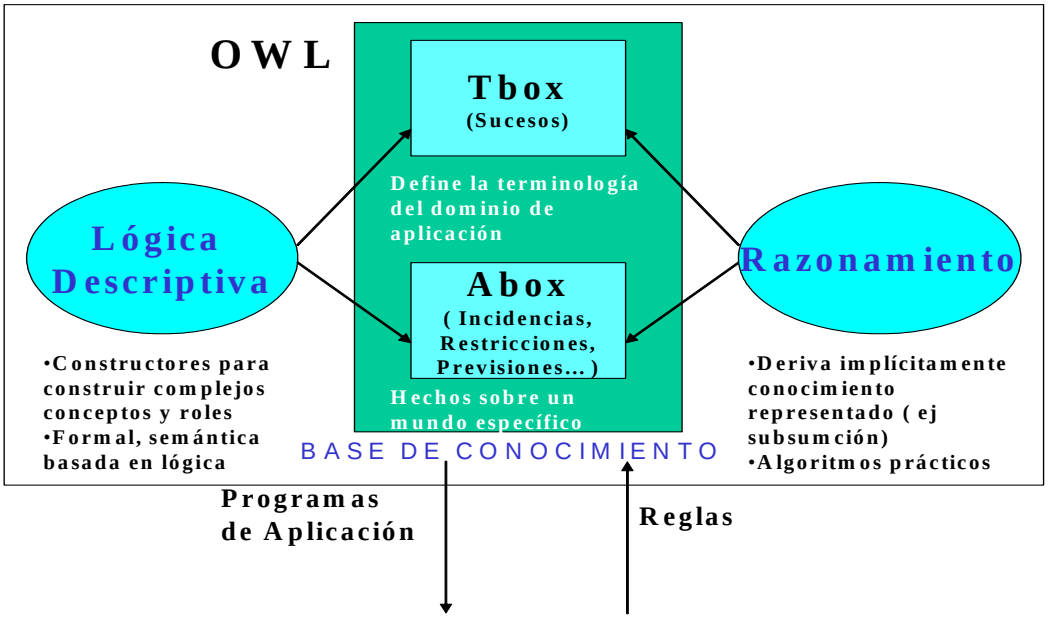


Figura 6.7 Arquitectura de un sistema de representación de conocimiento de "Sucesos" o eventos de tráfico basado en Lógica Descriptiva.

Como podemos apreciar en la figura 6.7, hay dos partes claramente diferenciadas, Tbox, formado por los términos y sus relaciones y el ABox que estará formado por instancias de los conceptos definidos en la parte terminológica. Dichos elementos pueden estar en el mismo fichero, aunque es práctica recomendable que se encuentren separados, debido principalmente a que las instancias generalmente serán modificadas periódicamente.

6.7.1 TBox: modelando términos y relaciones

Tomando como punto de referencia la entidad SUCESO, eje central del modelo, dicha entidad establece una serie de relaciones con otras entidades (Localizacion, Informacion, Causa, Vialidad, Accion y Escenario) que bajo el criterio de modularidad elegido, estarán especificadas en otros subdominios.

Observando de nuevo en la figura 6.6 la relación entre la entidad Suceso y la entidad Escenario, éstas pasarán a convertirse en clases del nuevo modelo semántico formal.

Como ya se comentó en la propuesta metodológica, algunos de los atributos de las entidades del modelo de datos semántico no tendrán representación en la especificación formal.

La relación binaria que aparece, puede ser vista de dos formas según se aplique como dominio o como rango una entidad o la otra, de ese modo, nos encontramos con que la relación “compuesto_por” tiene su correspondiente inversa en “es_contenido_en” y ambas expresan la relación que existe entre ambas entidades. El uso de ambas puede aportar una mayor expresividad a nuestro modelo.

Como queda patente en el diagrama de ER, y tras una revisión de su análisis de dominio, hay una serie de requerimientos o restricciones que el modelo semántico de datos cumple y por lo tanto el modelo formal también tiene que hacerse eco de ello.

Veamos cuales son las restricciones que conciernen a esta parte del esquema:

- Un escenario está compuesto por al menos un suceso y además en un orden concreto.
- Un suceso siempre es contenido por un escenario (pero solo por uno). Es decir, no pueden existir dos escenarios que contengan al mismo suceso.

Hay que considerar, tal y como se comentó al hablar de agregación en el punto 6.6.2.5, que la relación binaria “compuesto_por”, posee un atributo que la caracteriza, denominado “orden”, el cual expresa la lista ordenada de sucesos que componen un escenario. Este tipo de atributos, obliga a la creación de una nueva clase “ListaOrdenadaSucesos”, que en realidad estará formada por todos aquellos sucesos que de forma ordenada han ido sucediendo en el escenario común. Para poder especificar este conocimiento, se creará una nueva relación binaria cuyo dominio sea la entidad “Escenario” y cuyo rango la lista ordenada de sucesos, representada por “ListaOrdenadaSucesos”. (Ver documento “**Detalles de la especificación en las**

ontologías creadas", en la URL http://robotica.uv.es/tesis/javi/document.html y en el CD-ROM que acompaña esta memoria).
Estudiemos ahora el grado de la relación(es) o en su lugar las restricciones impuestas o requerimientos de esta parte del modelo. Lo expresamos en DL.

- **Restricción 1:**

La relación "compuesto_por" es la relación inversa de "es_contenido_en".
compuesto_por $\doteq es\_contenido\_en^{-}$

- **Restricción 2:**

Un escenario está compuesto por al menos un suceso.
Escenario $\sqsubseteq (\geq 1\ compuesto\_por.Suceso)$

- **Restricción 3:**

Un suceso siempre es contenido por un escenario (pero solo por uno).
Suceso $\sqsubseteq (=1\ es\_contenido\_en.Escenario)$

Proceso de edición[29]

Para explicar con mayor detalle como se lleva a cabo el proceso de edición, remitámonos a la figura 4.13, la cual expresa la relación de causalidad en un suceso. En esta relación, además de la entidad Suceso, aparece la entidad Causa (figura 6.8), la cual debe estar especificada formalmente.

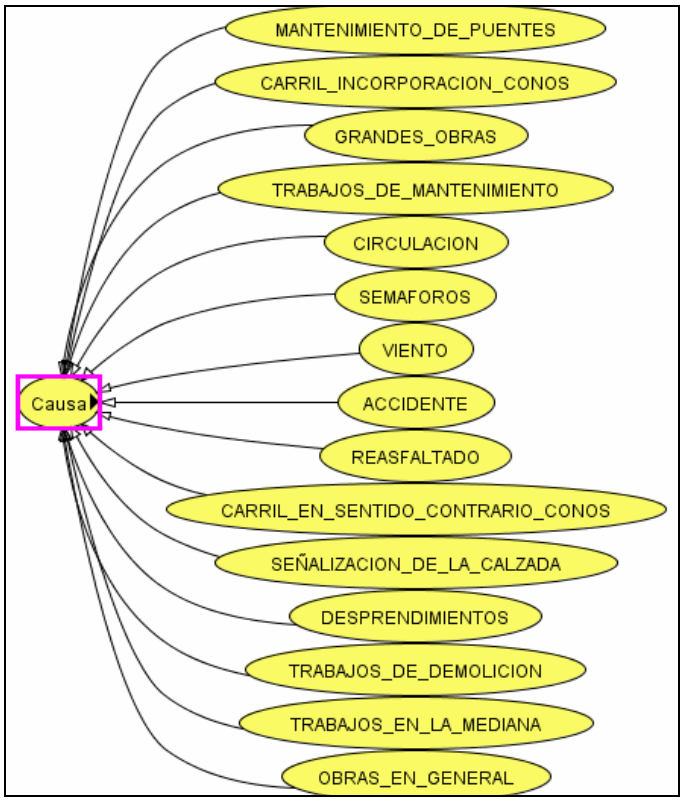


Figura 6.8 Parte de la subtipación de la entidad Causa

[29] Las capturas mostradas en esta sección correponden al uso del editor OilEd.

La siguiente captura (figura 6.9) muestra la propiedad "causa" que representa la anterior relación, de tal forma que dicha propiedad especificada en el dominio de "sucesos" establece que las únicas causas posibles de la aparición de un suceso (Accidente, Incidente, Medida) serán valores de la entidad "Causa", especificada en otro subdominio u ontología diferente (de ahí #5 que corresponde al subdominio "Relaciones_Sucesos").

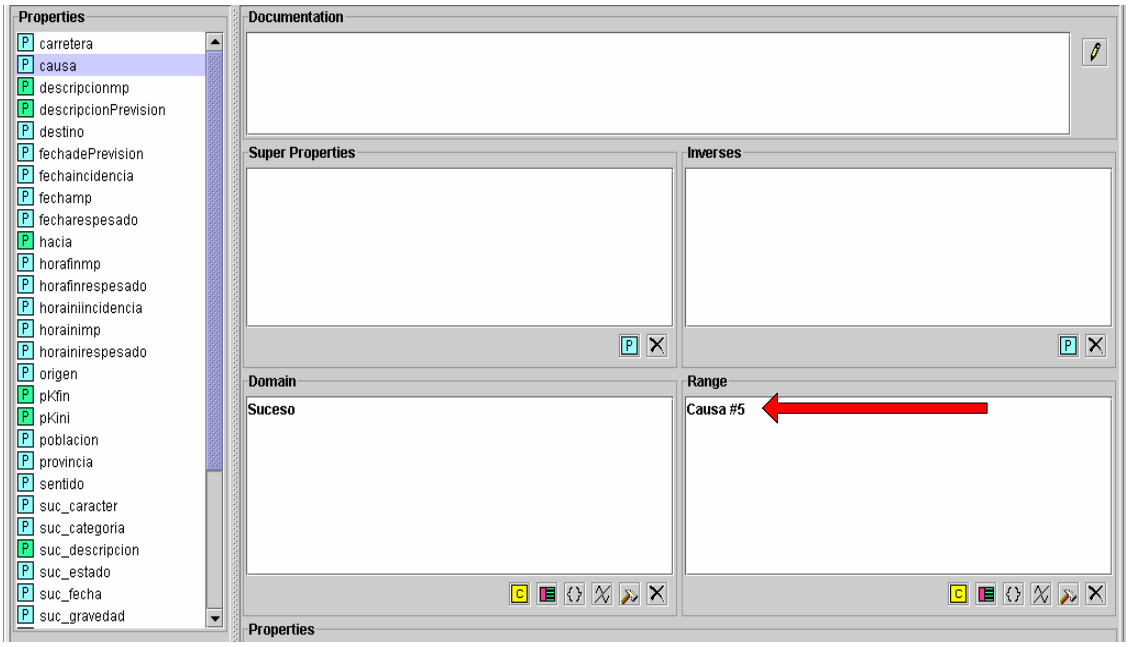


Figura 6.9 Edición de la propiedad "causa", en el subdominio "sucesos".

Por lo que a la hora de expresar nuestro conocimiento base en extensión, o dicho de otro modo, al completar nuestra base de conocimiento dinámicamente mediante instancias de sucesos, y más concretamente como veremos a continuación, instancias de incidente, podemos especificar sus diferentes atributos haciendo referencia a valores concretos de la entidad Causa, especificada en una ontología diferente a la de Sucesos.

Subtipación de la entidad Suceso: Concepto de Incidente

En la figura 6.10 podemos apreciar una vista parcial de la jerarquía de clases elaborada para el subdominio de "Sucesos":

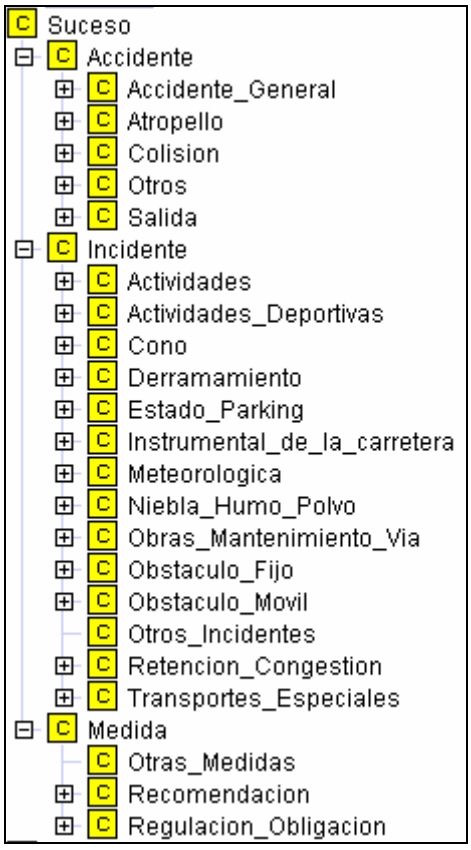


Figura 6.10 Vista parcial de la jerarquía de clases en el dominio de "Sucesos"

Para comprender mejor, el desarrollo y especificación de los conceptos, tratemos de ver un concepto determinado, el concepto "Incidente".

Requerimientos de incidente:

- Ocurre en una de las vías.
- Tiene lugar en una fecha dada.
- Cada incidente está caracterizado por un nivel de servicio.
- Tiene lugar en una población.
- Tiene lugar en una provincia.
- Está localizado en un punto (latitud, longitud).
- Ocurre en uno de los sentidos de la vía.
- Se inicia en un punto kilométrico de la carretera y finaliza en otro punto kilométrico de la carretera.
- Sucede hacia algún lugar.
- Puede ser que haya sido originado por alguna causa.
- Puede ser clasificado por el tipo de incidente.

Los requerimientos que poseen los conceptos principales de la ontología de Sucesos, como es el caso del concepto Incidente, desde el punto de vista de las ontologías serán tratados como restricciones de los conceptos, en su mayoría de cardinalidad.

Incidente → Suceso ∧ {Restricciones} ó
Incidente ⊆ Suceso ∧ {Restricciones}

Donde *Restricciones* equivale a la conjunción de todas aquellas restricciones sobre las propiedades que de manera local afectan al concepto de Incidente.

Por tanto, cualquier sistema de razonamiento podrá inferir que si "algo" es un incidente, de igual manera será también un Suceso y además deberá cumplir las restricciones (requerimientos) que completan su especificación.

En la figura 6.11 se describe la especificación del concepto "Incidente".

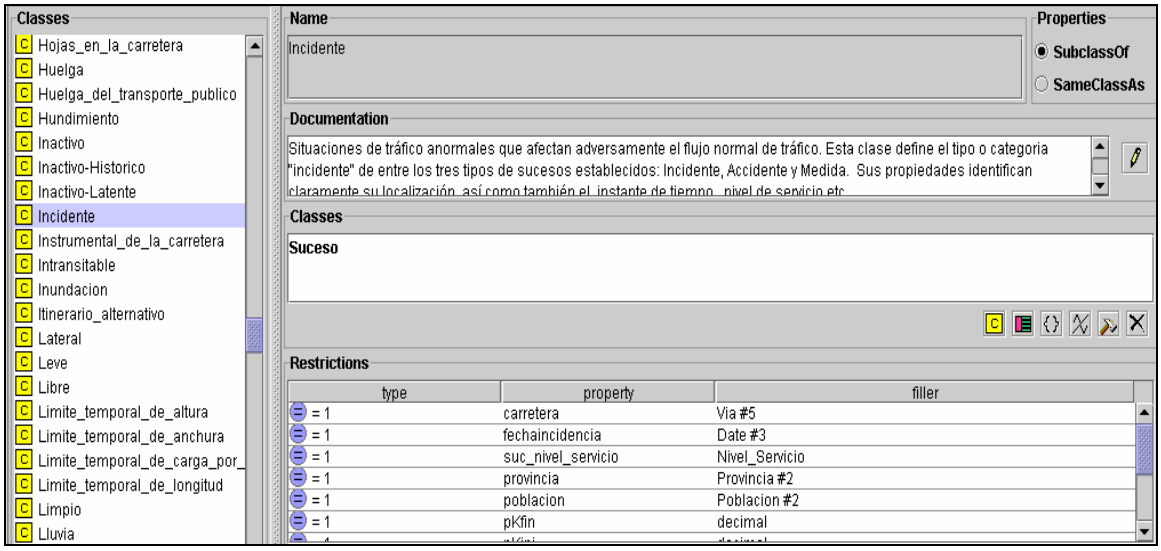


Figura 6.11 Definición del término "Incidente"

Como se observa, se han empleado los dos tipos de propiedades que permite OWL para representar los parámetros que describen una incidencia: propiedades de tipo objeto, las cuales toman como rango otras clases de la misma ontología (suc_nivel_servicio) o clases definidas en diferentes ontologías de conceptos de tráfico (por ejemplo provincia o causa), y también propiedades de tipo *datatype* de XML Schema.
El hecho de hacer uso de *namespaces* o URI´s hará posible la reutilización o relación de conceptos ya especificados en otros subdominios, de tal manera que estos URI´s deberán de darse explícitamente en la especificación.

La figura 6.12 muestra la especificación de concepto "Incidente" en la ontología de "Sucesos":

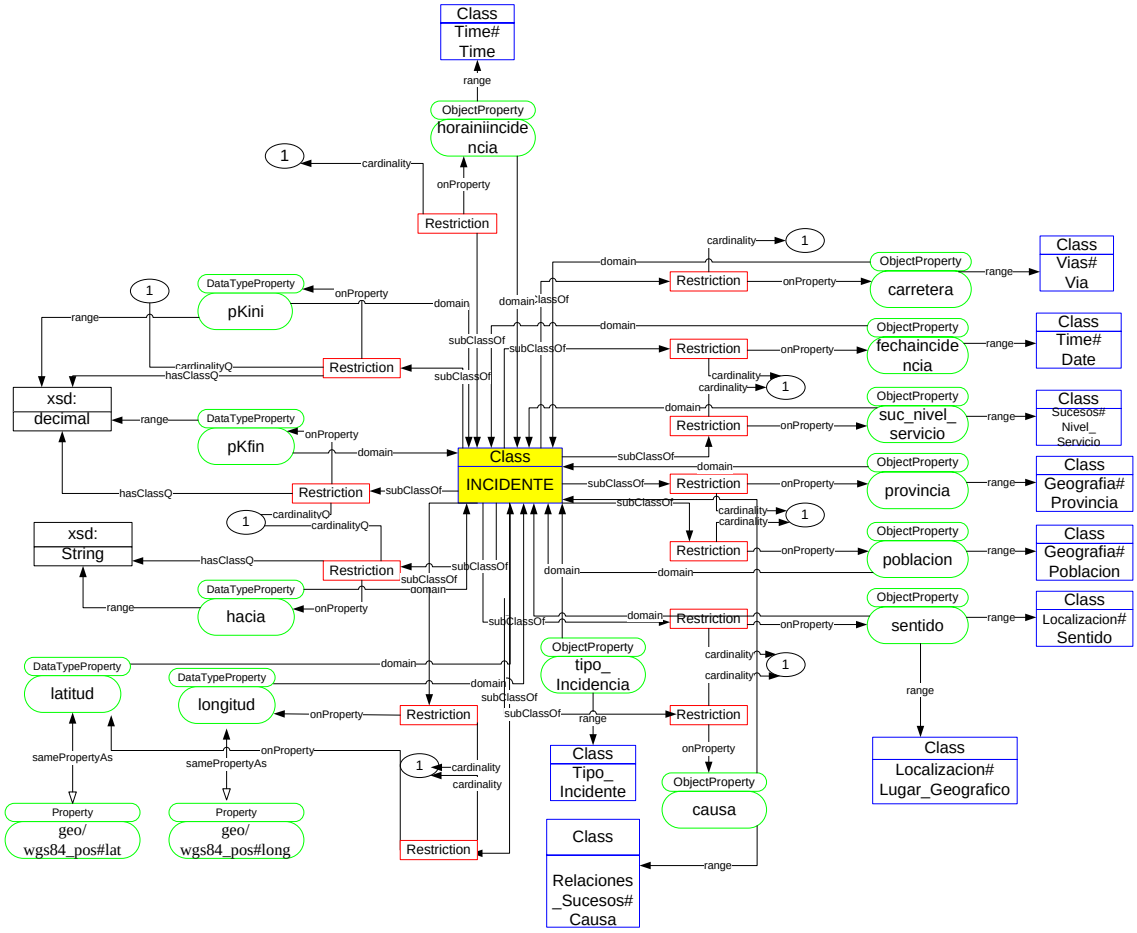


Figura 6.12 Esquema “Incidente”.

En el esquema anterior se aprecian las numerosas relaciones entre diferentes subdominios.
De esta forma y tal y como se comentó al hablar de la entidad Causa (especificada en un subdominio diferente al de Incidente), en la implementación podemos encontrarnos con la siguiente especificación:

<sucesos:causa rdf:resource= …/Relaciones_Sucesos#DESPRENDIMIENTOS>

En la figura 6.13 podemos apreciar una representación parcial del concepto “Incidente”:

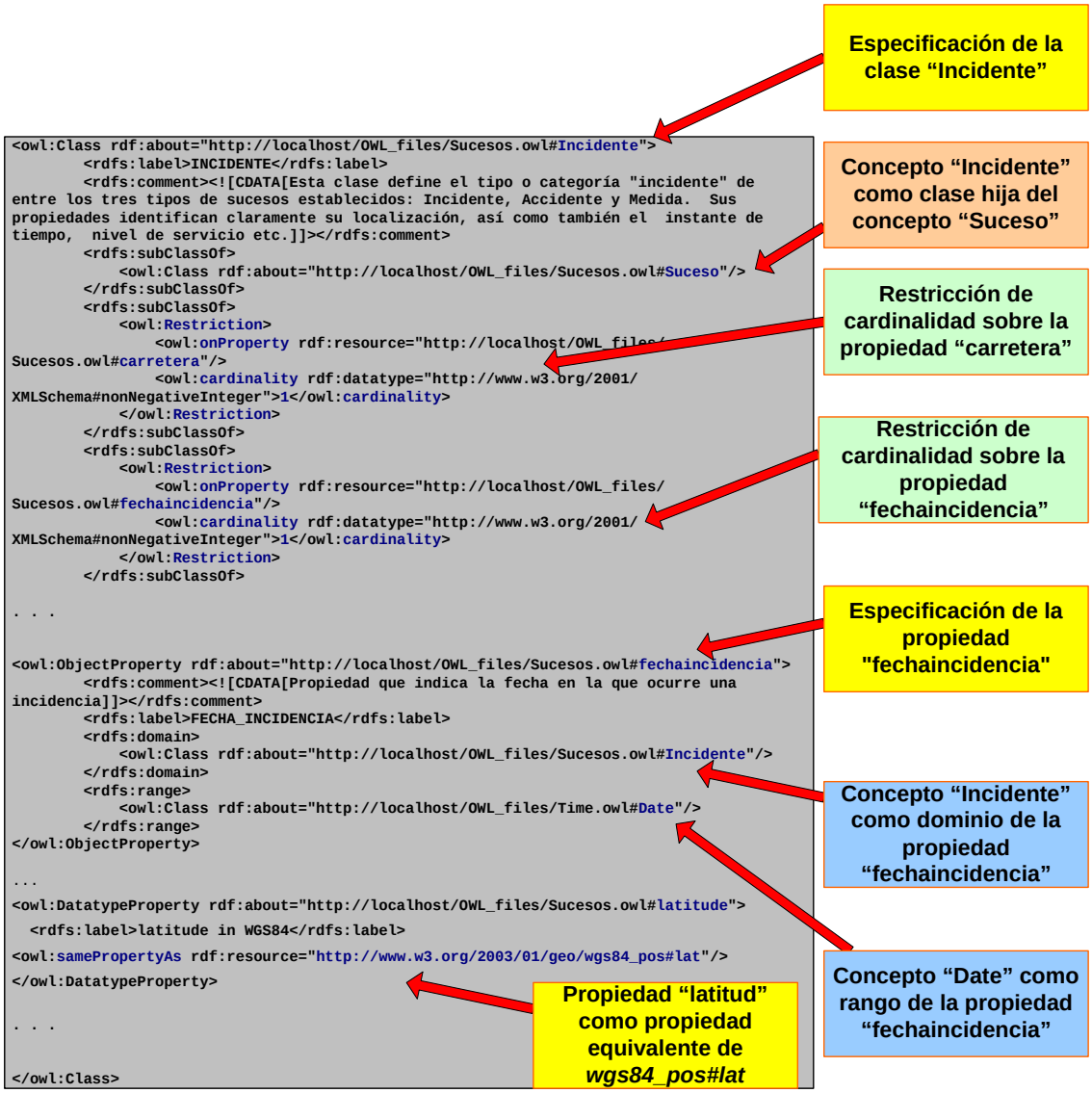


Figura 6.13 Especificación en OWL del subdominio ontológico “Sucesos”.

Esta ontología, también recoge otros conceptos importantes, como son el concepto Prevision (carácter de un suceso) y los conceptos Restricciones_Mercancias_Peligrosas y Restricciones_Vehiculos_Pesados como subclases del concepto Restricciones definido en la ontología. Ver figuras 6.14, 6.15 y 6.16.

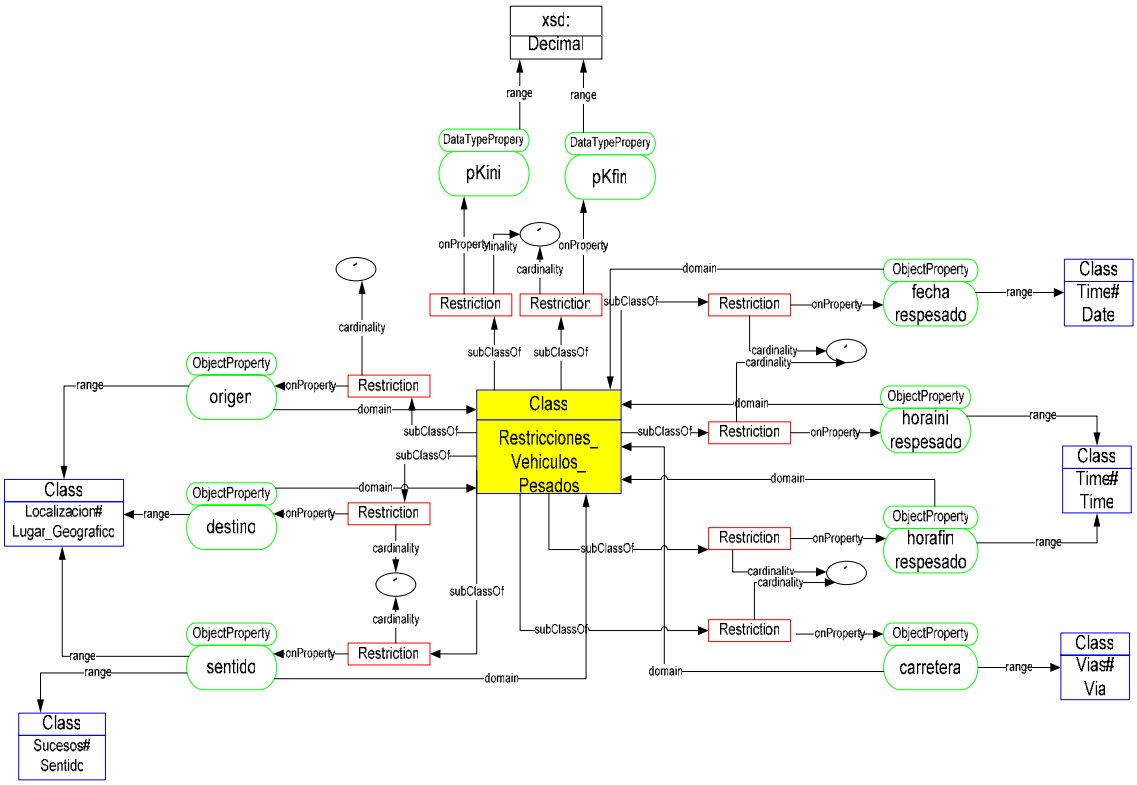


Figura 6.14 Esquema Restricciones a vehículos pesados.

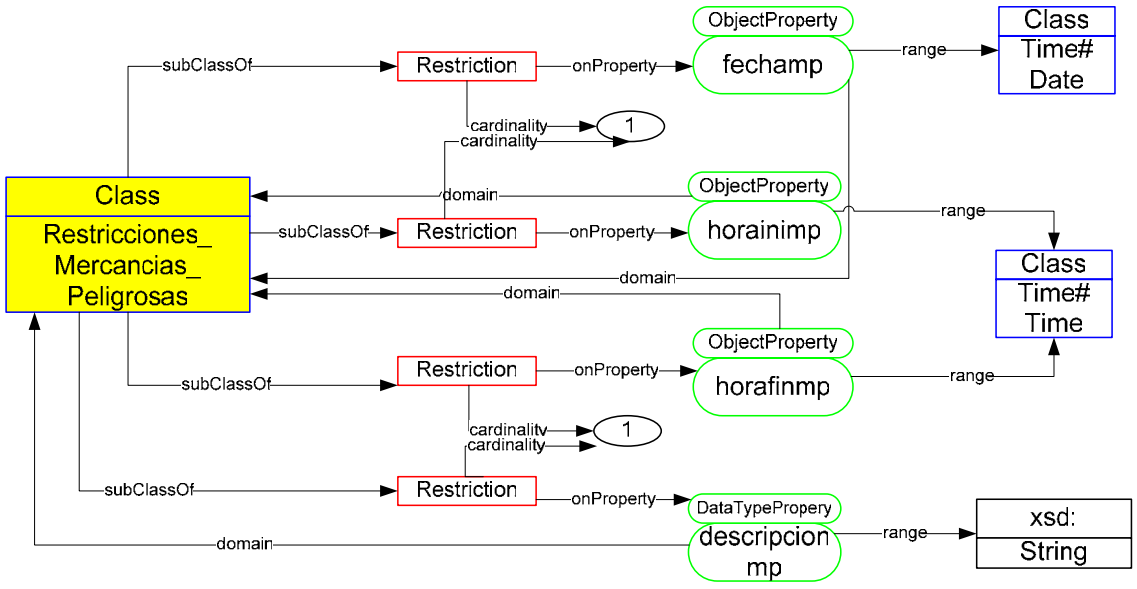


Figura 6.15 Restricciones referentes a mercancías peligrosas.

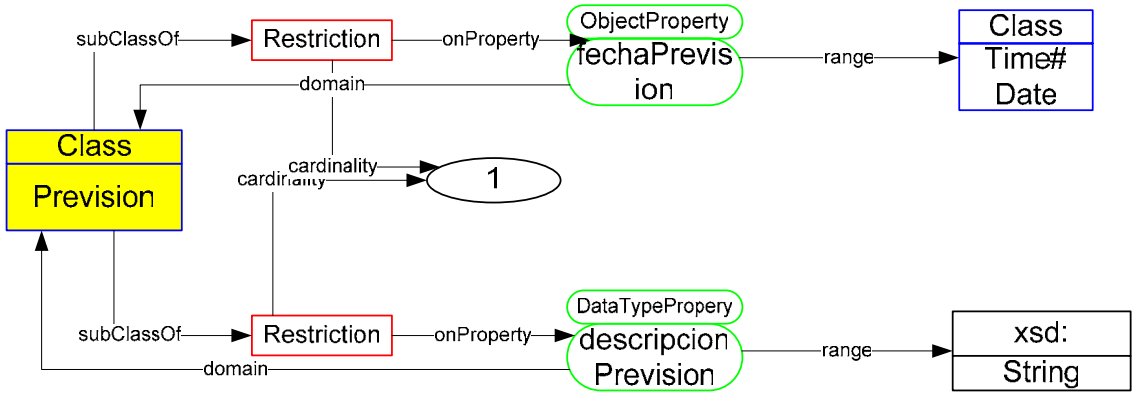


Figura 6.16 Esquema del concepto Previsión de tráfico.

6.7.2 Versatilidad y reutilización de ontologías

6.7.2.1 Ontología de Rutas

Se corresponde con la subontología de conceptos relacionados con los servicios de cálculo de rutas (urbanas e interurbanas). Ver figura 6.17.

La clase principal de esta ontología es Ruta, la cual tiene dos subclases que especifican los dos tipos de rutas más importantes, como son *Ruta_Urbana* y *Ruta_Interurbana*. El resto de los conceptos como *ItemRuta* y *Categoría_Ruta* que aparecen descritos en esta ontología sirven para dar valor semántico a los atributos (propiedades) que poseen estas tres clases principales. Por ejemplo, con las propiedades *item* y *categoría* se describen cuales son las diferentes partes que componen la ruta y cual es el tipo o categoría de ésta.

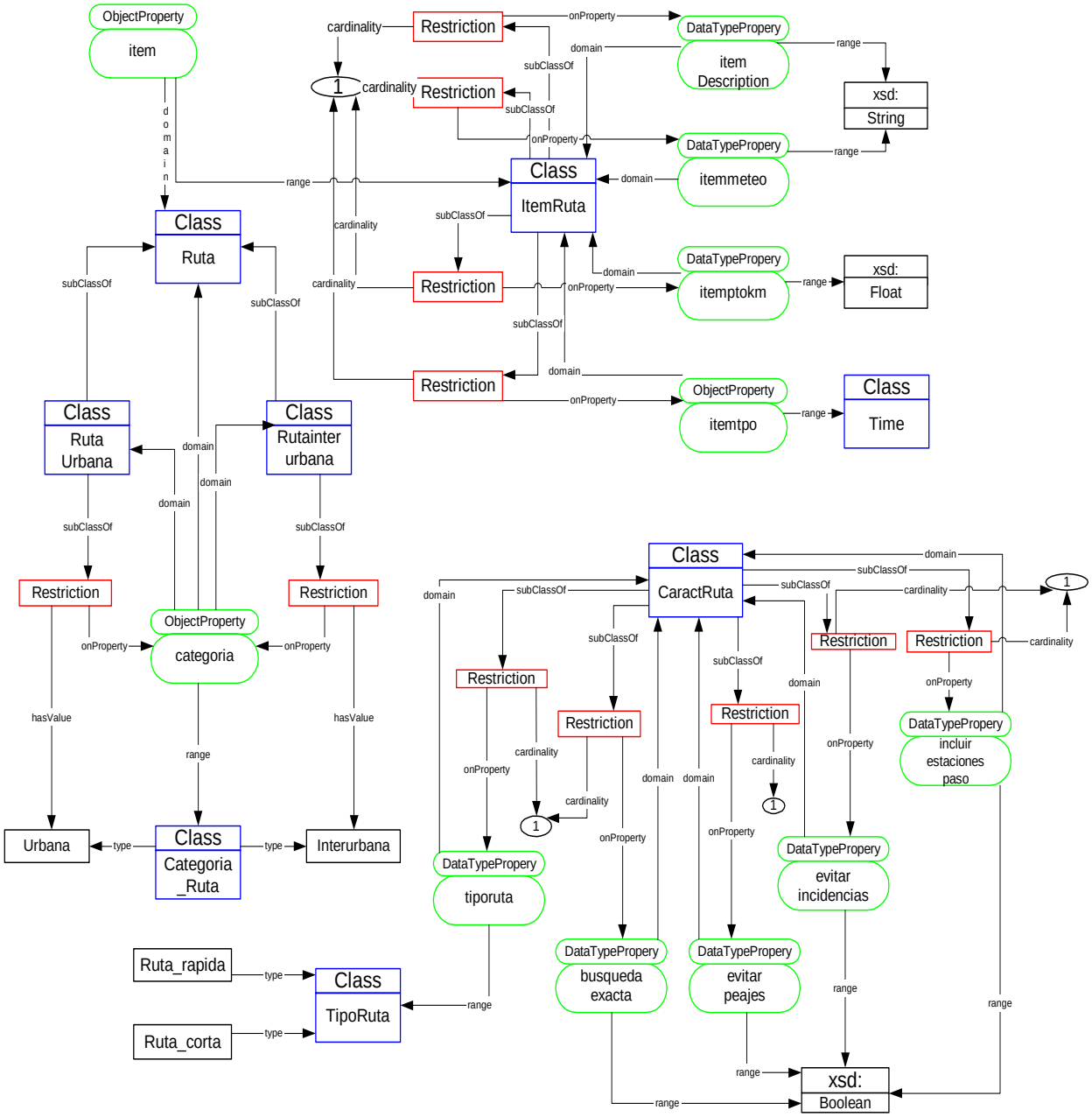


Figura 6.17 Diagrama de la ontología de Rutas

6.7.2.2 Ontología de Geografía

La ontología de geografía (figura 6.18) desarrollada posee un número limitado de clases. Mediante éstas se especifica el valor semántico para conceptos como *País, Comunidad_Autónoma, Provincia, Comarca, Población* y *Nacionalidad* además de indicar cuales son las relaciones que guardan entre sí (*ser_parte_de*$^{+}$ y *estar_compuesta_por*). Esta ontología incluye un gran número de instancias para estos conceptos. Es de gran utilidad en otras ontologías, ya que muchos de los conceptos que se definen en las ontologías de conceptos de tráfico hacen uso de los conceptos definidos en la ontología de *Geografía*. Por ejemplo, conceptos como *Incidente* de la

ontología de *Sucesos*, poseen propiedades como población, provincia y sentido, para localizar la incidencia de tráfico; y en la ontología de conceptos de ruta, una ruta interurbana pasará por diferentes ciudades, por lo que deben de ser identificadas como instancias del concepto *Poblacion*.

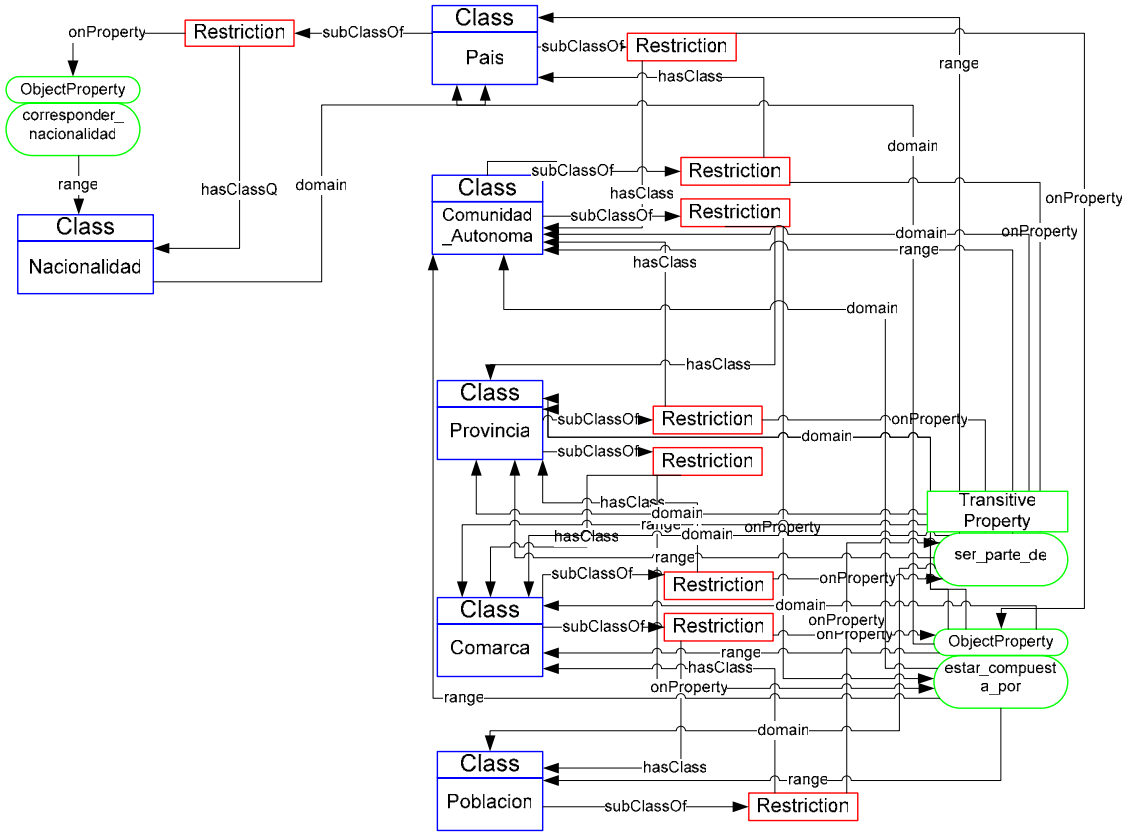


Figura 6.18 Diagrama de la ontología de Geografía

6.7.2.3 Ontología Time

Debido al interés en especificar parámetros temporales en los requerimientos, surge la necesidad de utilizar ontologías que representen entidades referentes a conceptos como Fecha y Hora. (Figura 6.19)

El diseño de esta ontología tuvo como base principal la ontología *Calendar* desarrollada por Luigi Ceccaroni [Cec02].

La ontología *Time*[30] tiene los conceptos temporales básicos como son *Date* y *Time* para que sean utilizados como conceptos semánticos de los parámetros temporales de los SW. Aunque también se han definido otros conceptos como los días de la semana, una clase *Calendar* que posee dos propiedades para representar *Date* y *Time* en una única clase, y propiedades extra para la clase *TimeFormat* que permitan indicar la franja horaria y el formato en el que está escrita (AM, PM).

Para el concepto *Date* se han definido las propiedades *day*, *month* y *year*, y en el concepto *Time* se han definido *timeHour*, *timeMinute* y *timeSecond*.

[30] Debido a la modificación de una ontología ya creada, inicialmente en inglés, se ha preferido conservar la especificación de los términos en su lengua origen.

Existen otras ontologías de tiempo como la desarrollada por Feng Pan [Pan04] del *Information Sciences Institute* en el que basan sus descripciones de conceptos temporales en dos tipos de entidades: intervalos e instantes.

El concepto de *intervalo temporal*, puede ser utilizado por ejemplo, para determinar la duración de un viaje. Este concepto también se ha tenido en cuenta en la construcción de la ontología. Para ello se ha definido un concepto *Duration* en el que se expresa el intervalo de tiempo en horas, minutos y segundos.

El diagrama que describe los conceptos más importantes de esta ontología es el siguiente:

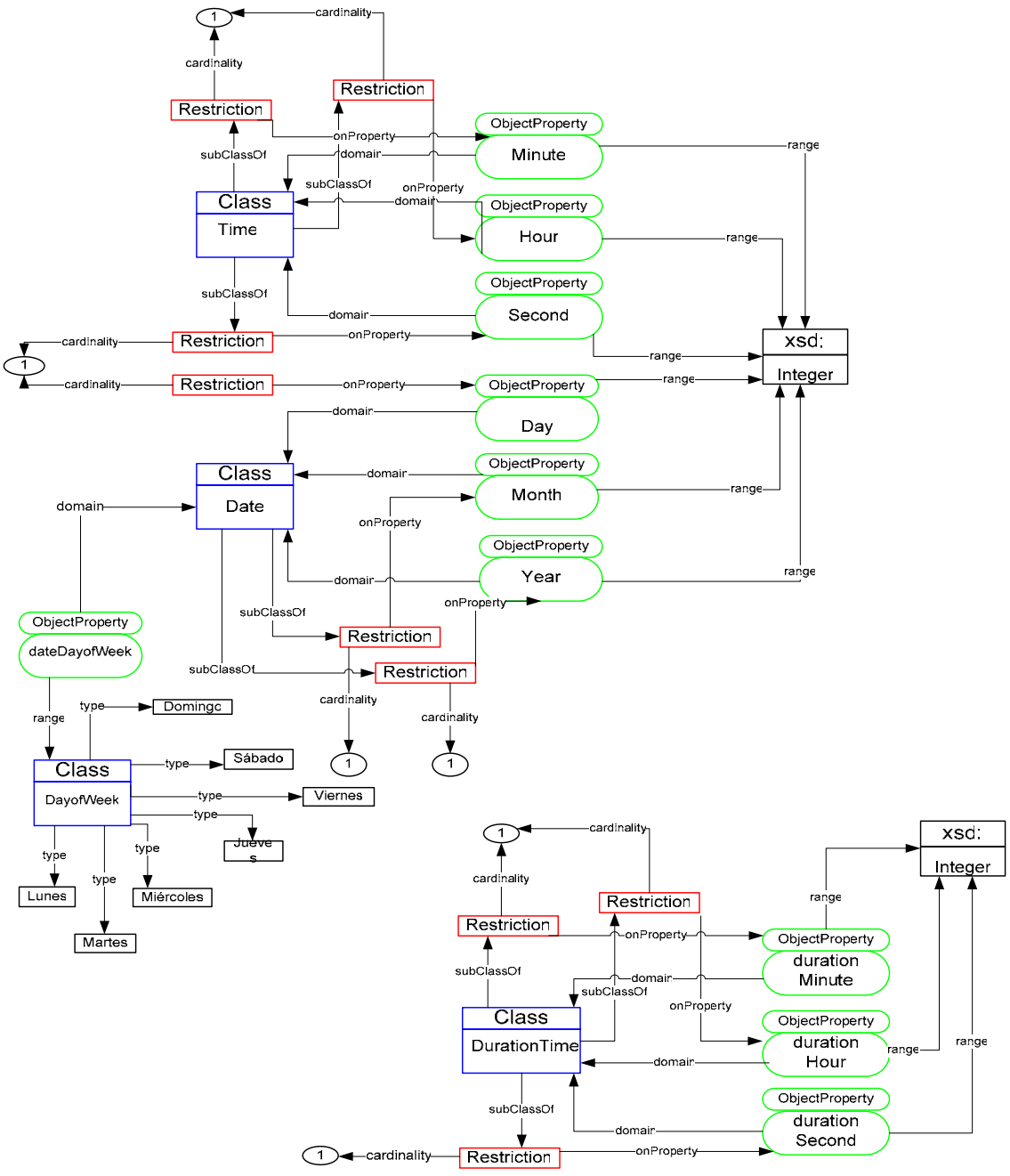


Figura 6.19 Diagrama de la ontología Time

Como se puede observar en el diagrama anterior, para poder realizar comparaciones numéricas con tipos de datos *Integer* del XML Schema, las propiedades de las conceptos de las ontologías no deben de ser de tipo *DatatypeProperty*, sino que deben de apuntar a objetos de Xml Schema *Integer*, y por tanto ser propiedades de tipo objeto (*ObjectProperty*)[31]. Esto no lo cumplía la ontología de tiempo escogida, por lo que se modificaron estas propiedades para poder hacer uso de esta funcionalidad.

6.7.2.4 Ontología GEO WGS84

Otra de las ontologías que han sido reutilizadas en la modelización de una incidencia es el vocabulario definido en RDFIG *Geo vocab workspace* donde determinan un vocabulario para representar información sobre latitud, longitud y altitud de un punto, referenciadas mediante coordenadas geodésicas en el datum WGS84 (*World Geodesic Survey*) en RDF [GEO03]. Mediante este vocabulario se puede especificar la localización de una incidencia utilizando este tipo de coordenadas. (Figura 6.20)

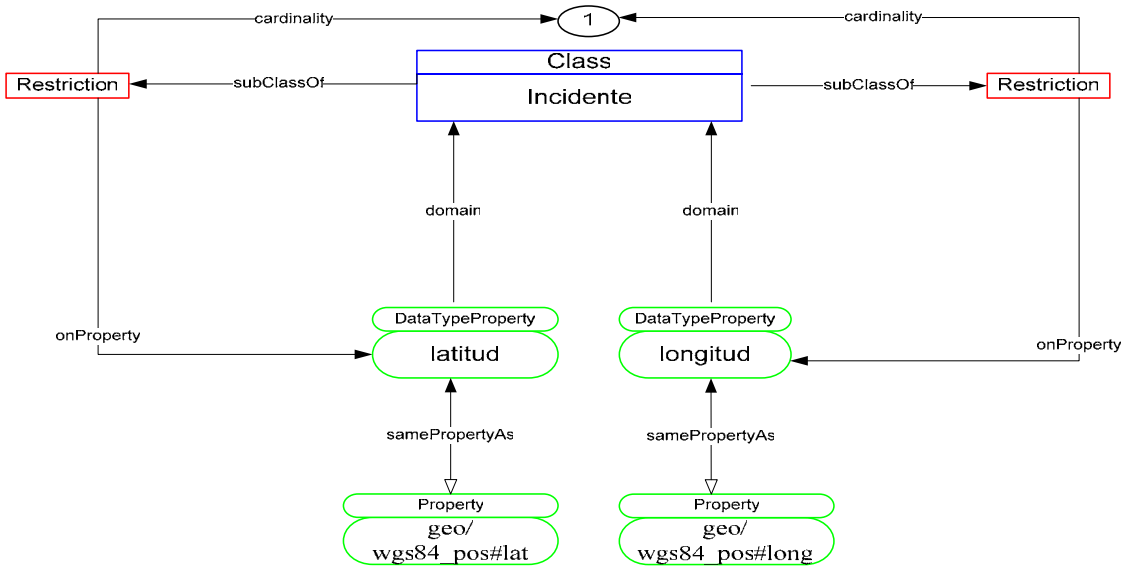


Figura 6.20 Uso de WGS84 definido en RDFIG.

6.7.3 ABox: modelando estados específicos del mundo.

Una vez especificada la parte principal del conocimiento (términos y relaciones), se procede a la adición de instancias, es decir a la concretización o extensión del conocimiento. En nuestro caso de estudio (instanciación de incidencias), y tal y como ya habíamos apuntado en la introducción de este punto, el fichero que contenga las instancias sobre incidencias, será un fichero distinto (incidencias.owl) al que contiene los conceptos en los que se basa (sucesos.owl). (Ver figura 6.21)

Al contrario, de lo que ocurre con el TBox, donde se recurre al uso de editores que faciliten su especificación, en el caso de las instancias, se han hecho uso de agentes "wrapper" (agente que extrae la información desde la Web) mediante *Scripts*

[31] Funcionalidad del Repositorio escogido.

desarrollados en *WebL* [Mar99]. (Ver URL http://robotica.uv.es/tesis/javi/scripts.html o CD-ROM que acompaña esta memoria) Estos agentes, extraen la información de Internet ayudándose del soporte de las ontologías.
Estas instancias tienen el aspecto que aparece en la figura 6.22.

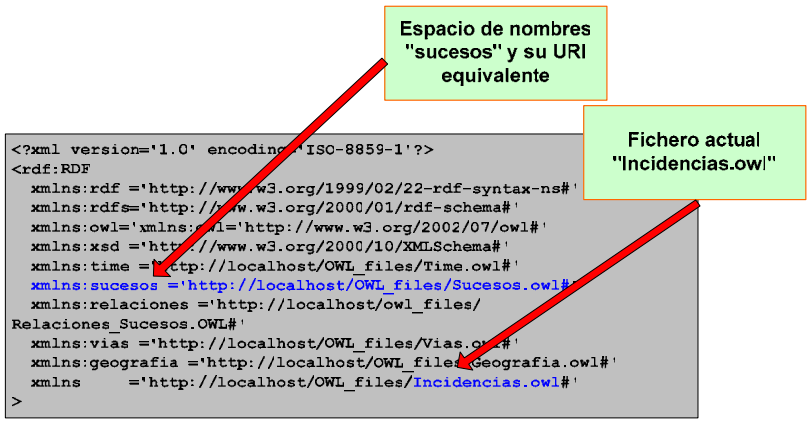


Figura 6.21 Espacios de nombres en el fichero de instancias de incidencias "Incidencias.owl"

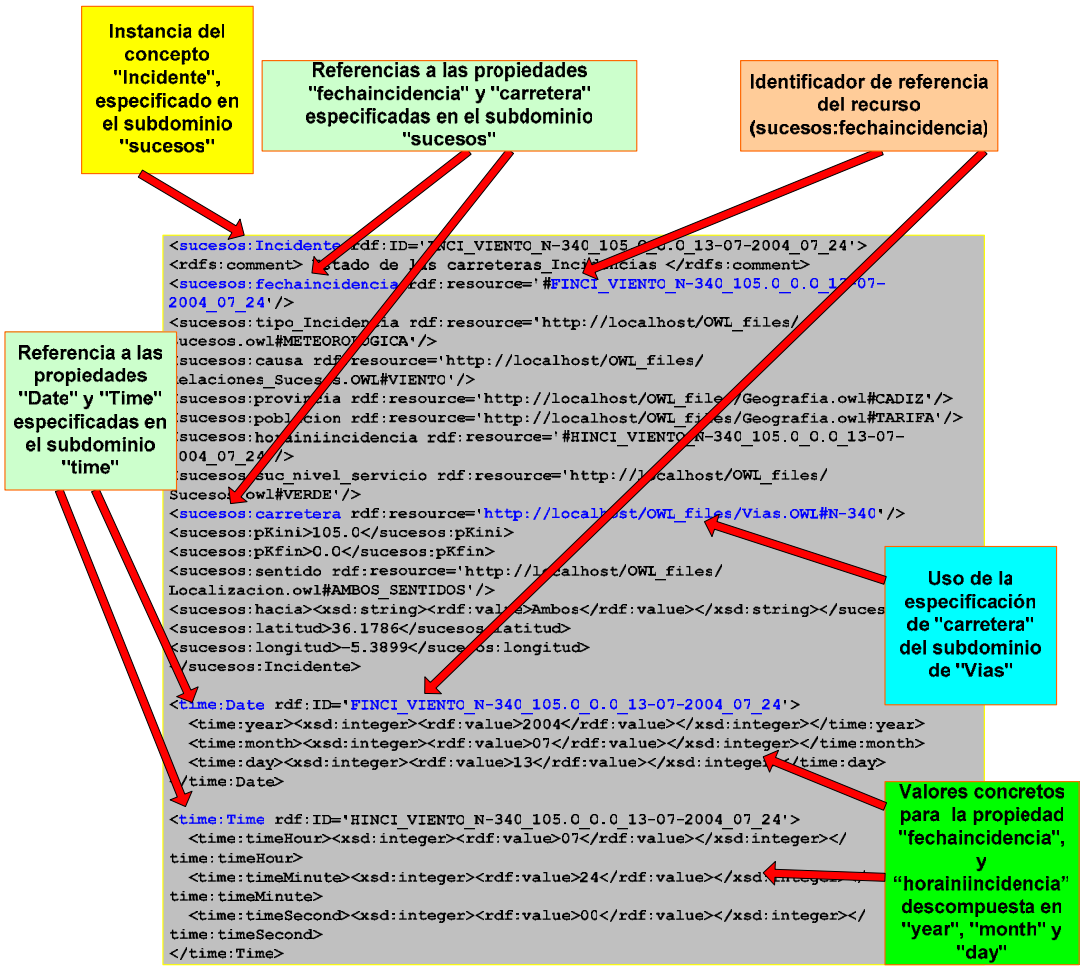


Figura 6.22 Instancia de la clase "Incidente" especificada en "Sucesos.owl"

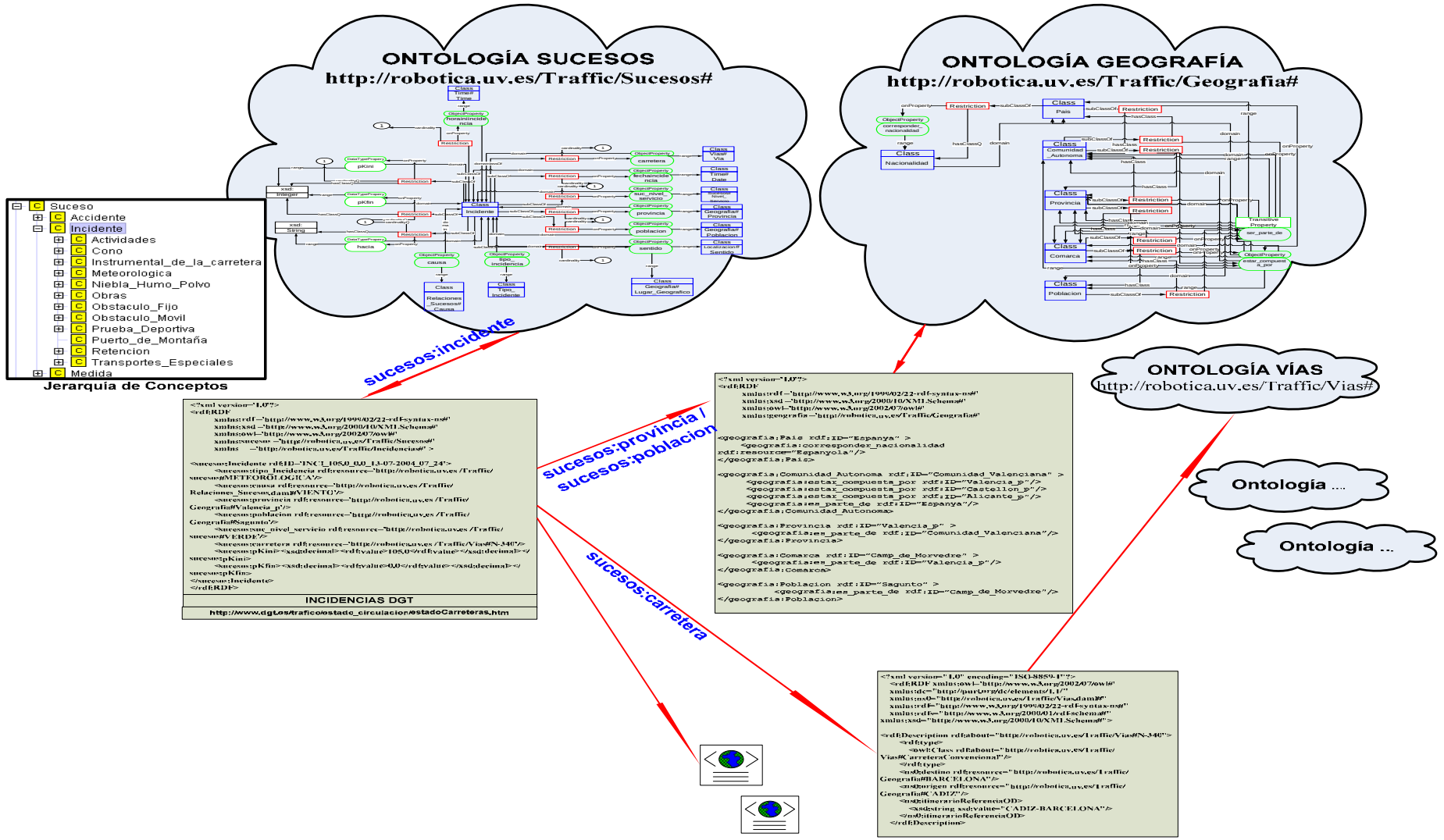


Figura 6.23 Relaciones entre la diferentes ontologías, a través de los atributos de una instancia de Incidente

Podemos apreciar en la figura 6.23 las relaciones entre las diferentes especificaciones así como la naturaleza distribuida del conocimiento, a partir de una vista de una instancia determinada.

Una vez instanciado, podemos probar nuestro sistema de inferencia mediante cualquier razonador como RACER o SeBOR, para recuperar de la base conocimiento las instancias de un determinado concepto etc.

Para lograr el objetivo de **ontología multilingüe** se pueden usar varias técnicas como describir el mismo conocimiento en varias ontologías, cada una en un idioma distinto, manteniendo toda la estructura y semántica descrita en la ontología original, y haciendo uso del operador de equivalencia entre términos y relaciones (Ver figura 6.24).

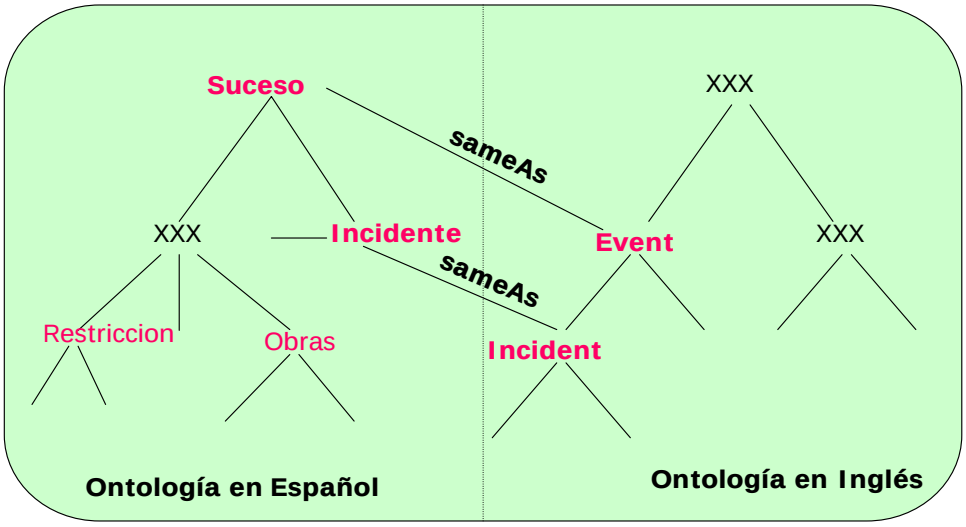


Figura 6.24 Equivalencia entre ontologías

Otra posibilidad es hacer uso de atributos *XML:lang* en los comentarios, que posibiliten en una misma ontología la descripción de sus términos en varios idiomas. Sin embargo, esta última técnica no funciona en algunos razonadores como RACER debido a la limitación de su lenguaje de requerimientos para acceder a determinados campos como *rdf:comment* de la descripción del concepto.

En la figura 6.25 podemos ver como a partir de una instancia de “Incidente” y haciendo uso de esta última técnica, es posible obtener información no explícita a priori (en este caso el significado de nivel de servicio “verde”) tanto en castellano como en inglés, mientras que en la figura 6.26 debido a la especificación en la consulta de que el idioma requerido sea el español, solo se obtiene información en este idioma.

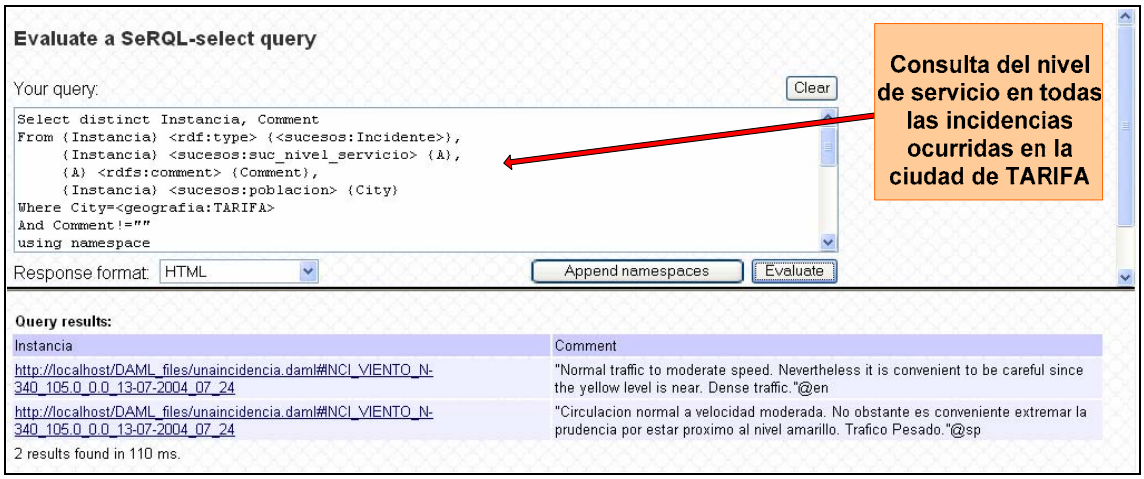


Figura 6.25 Consulta sin especificar el idioma en una base de conocimiento con dos idiomas (SeBOR)

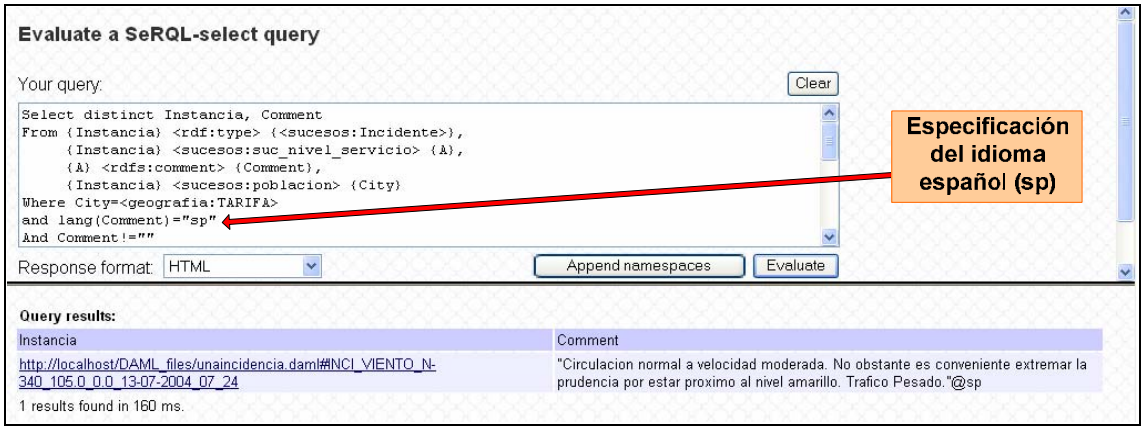


Figura 6.26 Consulta con especificación del idioma español

6.8 CONCLUSIONES

Se ha propuesto y validado una ontología para manejo de información de tráfico vial, campo en el cual según lo descrito en el estado del arte, no se habían encontrado vocabularios de tipo semántico.

A partir del estudio de metodologías y propuestas para la construcción de ontologías se ha presentado una extensión a los trabajos ya existentes, los cuales a juicio del autor de este trabajo en su mayoría no están enfocados a la existencia de un estudio previo que se pueda tomar como base. Debido al diferente origen de las fuentes de información, y a que a veces éste puede ser un modelo de datos semántico, se ha propuesto una extensión metodológica para la traducción de un modelo de este tipo a modelos semánticos formales, basada en metodologías tradicionales, pero con la adición de nuevos elementos fruto de nuestra experiencia.

La extensión plantea el uso de modelos de datos (ER) preexistentes y para su descripción se ha detallado a partir de la construcción de una base de conocimiento.

A su vez, se han extendido ontologías que ya existían como Time y reutilizado otras como GEO.

En base a la representación explícita de la estructura y semántica de los vocabularios de tráfico vial y de los conceptos a los que se refiere, es posible desarrollar herramientas informáticas, agentes inteligentes u otras aplicaciones, que procesen la información y conocimiento representados para realizar diversas tareas relacionadas con la distribución por parte de las autoridades o administraciones pertinentes, así como garantizar un acceso más eficiente y eficaz por parte de los usuarios.

Para abordar el problema de posibles aplicaciones multilingües, se han empleado varias técnicas. La primera de ellas es hacer uso de atributos *XML:lang* en los comentarios, que posibiliten en una misma ontología la descripción de sus términos en varios idiomas. Otra posibilidad se basa en la creación de correspondencias entre conceptos entre diferentes ontologías equivalentes en su estructura y contenido pero traducidas a distintos idiomas.

Tras la obtención de una instancia de un portal web de tráfico española, y por tanto en castellano, será posible determinar, por el URI que especifica un determinado concepto, su definición en la ontología a la que pertenece, así como la equivalencia con el término perteneciente a otra ontología (su correspondiente en otro idioma), por lo que podremos obtener información en cualquier lengua.

Por ejemplo, a través de un requerimiento de idioma en inglés y a partir de una instancia en castellano, seremos capaces de obtener una definición en el idioma requerido. Si un cliente interacciona con el sistema, determinando que el idioma elegido es el inglés, la petición especificará este requerimiento, de tal forma que la información resultado será traducida (en realidad recuperada en castellano, pero documentada en inglés).

Otro problema distinto será el hecho de recuperar información desde servicios o portales web de otros países o administraciones. La solución a este problema pasa por describir estos portales como SWS específicos, donde el uso de ontologías en dicho idioma ayudarán tanto en la búsqueda como en el resto de tareas.

CAPÍTULO 7

EXTENSIÓN DE ALGORITMOS DE EMPAREJAMIENTO DE SERVICIOS WEB SEMÁNTICOS.

7.1 RESUMEN

Este capítulo describe las principales características de un algoritmo de emparejamiento de SWS. El algoritmo aprovecha al máximo las capacidades proporcionadas por la ontología de descripción de servicios y constituye una mejora en relación con propuestas existentes. Además se describen los principales componentes relacionados con el proceso de implementación de éste. El sistema desarrollado interactúa con ontologías de descripciones de conceptos desarrolladas en DAML+OIL y descripciones de servicios en DAML-S, usando como repositorio/razonador el sistema *Sesame+BOR*. Finalmente, se detalla el desarrollo de comparaciones entre parámetros de *profiles*[32] (perfiles) de servicio mediante consultas realizadas a la base de conocimiento, así como las diferentes pruebas orientadas a la comprobación funcional del emparejador semántico y al contraste con una versión del emparejador semántico que implementa el algoritmo original descrito por Paolucci et al. De este modo, se justifica la eficiencia y mejora del algoritmo propuesto en diferentes escenarios.

7.2 INTRODUCCIÓN

El sistema emparejador es una aplicación (biblioteca) que interactúa con un repositorio de perfiles de proveedores, y que como se verá en el capítulo 10, será utilizado por el agente Matchmaker dentro de la plataforma.. Los clientes por tanto envían peticiones mediante agentes que los representan al agente emparejador. La información intercambiada entre estos dos agentes, es la URI del *profile* .que describe el servicio buscado por el cliente.

Una vez el sistema emparejador dispone del *profile* del cliente, pasa a ejecutar el algoritmo de emparejamiento descrito en esta tesis. Este algoritmo está subdividido en varias fases, las cuales van filtrando los diferentes SW para obtener el servicio que más se parezca semánticamente a lo que solicitó el cliente.

[32]Profile provee una forma de describir los servicios ofrecidos por los proveedores y los servicios necesitados por los solicitantes. Esencialmente, es una pequeña descripción de lo que hace el servicio, describiendo el servicio como una función de tipos básicos de información (Sección 4.7.4).

La propuesta dada en esta tesis, toma como base la aproximación dada por CMU y concretamente el algoritmo de Paolucci et al.[Pao02a]. La principal característica que comparten, es el uso de una ontología de perfiles de servicios. Sin embargo, las principales diferencias tienen que ver más con la ampliación y división de algunos de los grados de similitud usados en el algoritmo tal y como veremos a continuación. El hecho de tomar las relaciones "fraternarles", es decir relaciones entre conceptos hijos de un mismo padre, con restricciones comunes, se convierte en una de las principales aportaciones en el algoritmo.

7.3 ALGORITMO DE EMPAREJAMIENTO PROPUESTO

Para el desarrollo de este algoritmo se han tomado como base los siguientes interrogantes:

- ¿Qué parámetros de la clase Profile podrían ser usados para la búsqueda de servicios?
- ¿Qué grados de emparejamiento se deberían utilizar en las comparaciones de los parámetros funcionales?

A partir del análisis de estos requisitos se ha desarrollado el algoritmo que se describirá en los siguientes apartados.

7.3.1 Algoritmo Base

El algoritmo consiste en buscar emparejamientos de valores semánticos que fueron utilizados para describir cada una de las capacidades del servicio, tanto por parte del proveedor como por parte del cliente en su petición. Por este motivo, las peticiones del cliente serán perfiles como los que utilizan los proveedores para anunciarse.

El algoritmo planteado en esta tesis toma como base el algoritmo de Paolucci et al., utilizado en su sistema emparejador, pero posee algunas diferencias, las cuales se centran en cuatro aspectos:

1. Filtrado en función del perfil de la petición, de los perfiles de proveedores utilizados para el emparejamiento, para reducir el coste del algoritmo.
2. Uso de parámetros no funcionales[33].
3. Obtención del grado de emparejamiento mediante el uso de nuevos grados.
4. Mejoras en la ordenación de resultados.

[33] Son atributos que no son obligatorios. Suelen utilizarse para que los propios proveedores añadan las características especiales que creen que ofrecen sus servicios Web. Se pueden añadir más a los ya definidos gracias a otras propiedades definidas para la clase Profile, pero los principales son: ServiceCategory, QualityRating y GeographicRadius.

7.3.1.1 Fases del algoritmo:

Las fases del algoritmo propuesto son:

- ***Fase 1***: Filtrar entre todos los anuncios de servicios, aquéllos que pertenecen a la misma categoría que la solicitada por el cliente:

El hecho de utilizar nuevos filtros nos permite aprovechar otras características que posee la clase *Profile* para anunciar SW y que no fueron explotadas por anteriores algoritmos.
El algoritmo de emparejamiento tiene un coste considerable ya que intenta emparejar la petición con cada uno de los proveedores disponibles, y para hacer esto cada una de sus entradas o salidas se contrasta con todas las de cada proveedor para encontrar la que más se le parezca semánticamente. Por esta razón, incluimos un filtrado anterior al emparejamiento, cuya finalidad es descartar proveedores que no cumplan unas determinadas características definidas también en los perfiles de la ontología.

El filtrado está basado en la categoría de servicio y es de carácter obligatorio en la formulación de la petición de servicio por parte del usuario. Servirá al sistema *emparejador* para obtener del repositorio de perfiles (*profiles)* aquella lista de servicios que pertenezcan a la categoría que busca el cliente.

- ***Fase 2***: De la lista resultante, combinar todas las posibles parejas entre la petición del cliente y cada uno de los anuncios de proveedores:

Esta fase consiste en establecer las distintas combinaciones posibles entre la petición dada por el cliente y los anuncios publicados por los proveedores, que por la fase anterior, pertenecen a la misma categoría de servicio.

- ***Fase 3***: Cada vez que se obtiene una pareja en la fase anterior, aplicar sobre ella los diferentes grados de similitud para los parámetros funcionales y calcular los pesos relativos para:

i. Parámetros funcionales correspondientes a salidas,
ii. parámetros funcionales correspondientes a entradas y
iii. parámetros no funcionales

El cálculo de pesos relativos a las entradas y parámetros no funcionales (ii e iii) se hará siempre que éstos hayan sido especificados por el cliente y si el grado de similitud de una pareja en cuanto a sus salidas es positivo, es decir, si tienen alguna salida en común.

El riesgo de obtención de falsos positivos o negativos será menor dependiendo de la precisión en la asignación de pesos a las parejas "*petición cliente – anuncio proveedor"*

Los dos principales procesos que caracterizan esta fase son:

a) **Obtención del grado de emparejamiento para parámetros funcionales**:

En [Pao02a] se establece una función de ranking (*DegreeOfMatch*()) en la que se diferencian cuatro posibles casos además del de fallo. Además se le asigna el mismo peso (exact) no solo a conceptos iguales sino también al caso en que los conceptos de la petición son subclase (relación directa) de los del anuncio.

En [Lei03] además de diferenciar los grados de Paolucci en cuanto a emparejamiento *exacto* y *ser subclase de*, destaca otro posible emparejamiento que ya fue anteriormente expuesto por [Gon01]:

- Grado de *intersection* cuando un anuncio y una petición son compatibles, es decir mantienen algo en común.

En esta tesis se propone un algoritmo con características similares a los aquí expuestos pero con importantes modificaciones en los grados de emparejamiento ya que en estos trabajos eran demasiado generales, por lo que se propone una versión compuesta por siete grados de emparejamiento que detallamos a continuación, en orden descendente de importancia:

- *Exacto:* los conceptos definidos por el cliente y por el proveedor son los mismos.

- *CsubclaseP:* dentro del árbol de la taxonomía de conceptos la distancia entre el concepto demandado por el cliente y el ofrecido por el proveedor es igual a 1, siendo por tanto, el descrito por el cliente, subclase directa del que proporcionó el proveedor. Ver figura 7.1.En este caso el proveedor ofrece un concepto más general que el que pide el cliente. El concepto del cliente es más restrictivo pero está incluido en el que proporciona el proveedor.

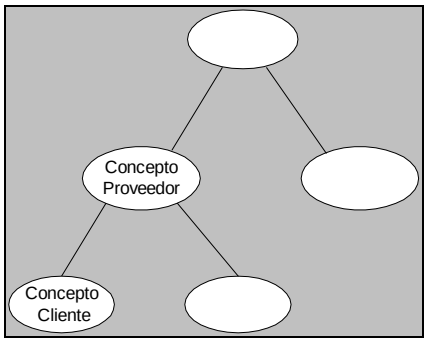


Figura 7.1 Grafo que representa el caso CsubclaseP.

- *PsubclaseC:* el concepto descrito por el proveedor es subclase directa del que pide el cliente, es decir, el proveedor ofrece un concepto más restrictivo que el demandado por el cliente. Ver figura 7.2.

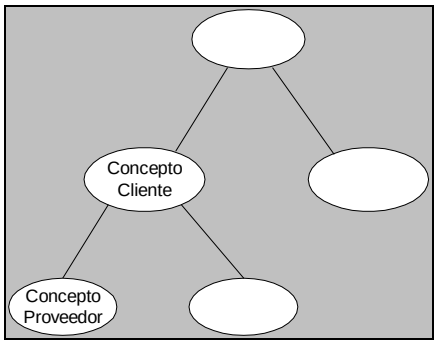


Figura 7.2 Grafo que representa el caso PsubclaseC.

- *PsubsumeC:* el concepto descrito por el cliente se encuentra dentro del subárbol de conceptos descendiente del definido por el proveedor (figura 7.3). Sería equivalente al *Plugin* definido en [Pao02a] aunque se diferencia de él en que no permite que el cliente especifique el nivel de profundidad máximo hasta el que llegará la búsqueda. Se ha optado por no permitir al cliente especificar esto porque consideramos que conceptos a distancias mayores o iguales a 3 niveles no tienen prácticamente relación semántica, debido al modo en que se construyen las jerarquías de conceptos. Por ello se ha decidido fijar la profundidad máxima en dos niveles, por lo que solo se comprueba si el concepto definido por el cliente es como máximo "nieto" del concepto del proveedor. De no hacerlo así, aumentaría el riesgo de falsos positivos y por tanto, podríamos obtener servicios como válidos, cuando la relación semántica entre el concepto que se provee y el que pide el cliente es tan lejano semánticamente que no responde a las expectativas de éste.

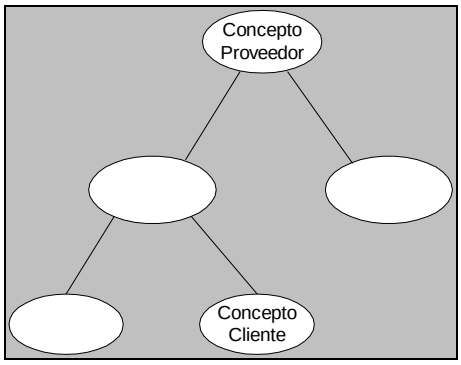


Figura 7.3 Grafo que representa el caso PsubsumeC

- *CsubsumeP:* El concepto descrito por el proveedor se encuentra dentro del subárbol de conceptos descendiente del definido por el cliente (figura 7.4), es decir, es el caso inverso al anterior, y es equivalente al grado *Subsume* definido en [Pao02a]. Como en el caso anterior, aquí también se limita a dos niveles de profundidad la comprobación de si el concepto demandado por el cliente incluye (*subsume*) al ofertado por el proveedor.

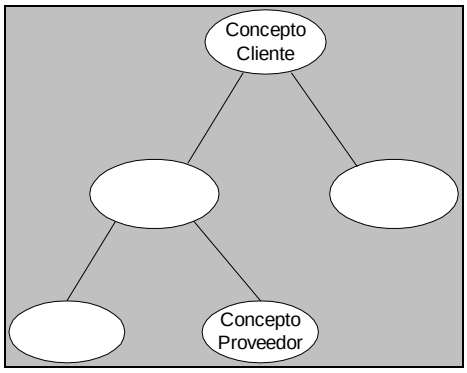


Figura 7.4 Grafo que representa el caso CsubsumeP.

- *ChermanoP:* El concepto proporcionado por el cliente y el del proveedor tienen restricciones en común en alguna propiedad que ambos poseen, y además, tanto el concepto ofertado por proveedor como el demandado por el cliente son hijos del mismo padre, es decir, son conceptos ”hermanos” (figura 7.5).

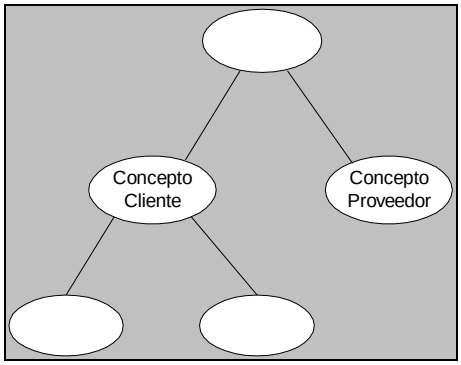


Figura 7.5 Grafo que representa el caso PhermanoP.

En una primera versión del algoritmo no era considerado el número de restricciones que tenían en común ambos conceptos, por lo que para mejorar este grado de familiaridad y así obtener una mejor diferenciación en el emparejamiento de los conceptos semánticos, se tuvo en cuenta este factor. Veámoslo con un ejemplo mediante la figura 7.6:

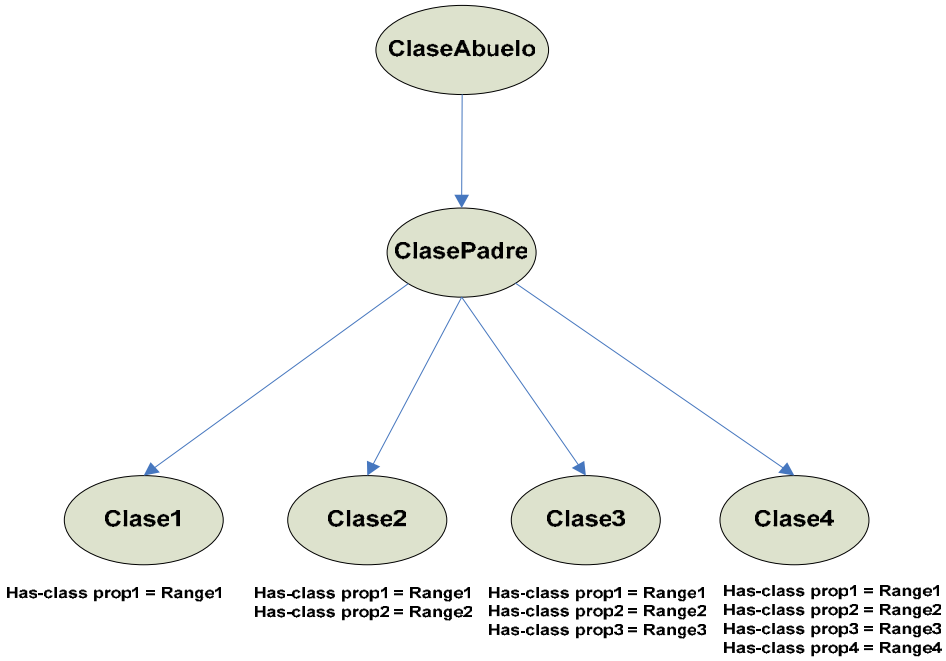


Figura 7.6 Factor: número de restricciones común.

Supongamos que el cliente requiere un parámetro del tipo *Clase2*, y dos proveedores ofrecen servicios parametrizados por conceptos pertenecientes a *Clase1* y *Clase3*. En la primera versión del algoritmo, se otorgaba el mismo peso a ambos proveedores, ya que ambos poseen restricciones comunes con el concepto proporcionado por el cliente y son hijos de un mismo padre, pero es lógico pensar que es mejor dar una mejor puntuación al proveedor de *Clase3*, ya que posee dos restricciones iguales para dos propiedades en común (*prop1*, *prop2*), mientras que el proveedor de *Clase1* solo una (*prop1*).

- *Fail:* Cuando no se cumple ningún caso de los anteriores, es decir el concepto del proveedor y el de cliente no tienen relación alguna.

El motivo de este orden para los casos de subclase, es decir, porque CsubclaseP es mejor grado que PsubclaseC, es debido a que consideramos más importante que el proveedor ofrezca un producto menos restrictivo que el cliente, ya que en este caso puede suceder que también ofrezca lo que solicita el cliente. En caso contrario, si un proveedor ofrece algo más restrictivo que lo que solicitó el cliente, puede que a éste no le interese. El caso del orden entre PsubsumeC y CsubsumeP es análogo solo que considerando una mayor distancia en el árbol de la taxonomía de conceptos especificados en la ontología.

El pseudocódigo de nuestro método para la asignación de grados de emparejamiento se puede observar en la figura 7.7:

```
DegreeOfMatch(outR,outA)
    {
        if Iguales(outA, outR)
            return EXACTO
        else if SubClassOf(outR,outA)
            return CSUBCLASEP
        else if SubClassOf(outA,outR)
            return PSUBCLASEC
        else if Subsumes(outA,outR)
            return PSUBSUMEC
        else if Subsumes(outR,outA)
            return CSUBSUMEP
        else if Hermanos(outR,outA)
            return CHERMANOP
        else
            return FALLO
```

Figura 7.7 Método de asignación de grados

b) Coincidencia exacta en parámetros no funcionales.

El algoritmo de Paolucci et al. no aprovecha otras propiedades de la descripción del perfil (*Profile)*, como son los parámetros no funcionales, que también aportan información semántica del servicio web, por lo que uno de los objetivos fue cubrir este vacío. El algoritmo emparejador explota al máximo las posibilidades de *Profile*, con el fin de hacer uso de todas las capacidades de los servicios, descritas semánticamente.

A continuación veamos qué filtros se han desarrollado para los diferentes parámetros no funcionales:

filtro para región: con él se comprobará si el radio geográfico dado por el cliente es el mismo que el proporcionado por el proveedor.

filtro para calidad de servicio*:* consiste en comprobar en el repositorio si la calidad que aportan los SW es la misma que la pedida por el cliente.

filtro para nombre de servicio*:* en esta consulta se comprueba si el cliente encuentra algún servicio en particular con el nombre que proporcionó. Este emparejamiento es sintáctico, ya que, tal y como está definida la propiedad *serviceName* en la subontología Profile, tiene como rango de valores el *datatype String* del *XML Schema*, con el que solo se pueden hacer comparaciones sintácticas.

filtro para nombre del proveedor*:* permite comprobar si el servicio web lo provee la empresa en la que está interesado el cliente. Es un parámetro como el anterior, es decir, únicamente se puede hacer un emparejamiento sintáctico debido a que esta propiedad de Profile también está definida como un *XML Schema String*.

Hay que destacar que los anteriores parámetros no funcionales van a incidir en el peso de distinta manera. Por ejemplo se ha querido premiar a aquellos parámetros de tipo semántico, de tal forma que si su comparación es positiva cada uno de ellos

acumulará 5 puntos en el cómputo de peso relativo a este tipo de parámetros, mientras que si tratamos con los emparejamientos de *serviceName* y *ActorName* (tipo sintáctico) aportan 3 y 2 puntos respectivamente.

- *Fase 4*: Teniendo en cuenta los pesos anteriormente calculados, ejecutar el proceso de ordenación para insertar en el lugar correcto de la lista de servicios resultado de la consulta. El servicio que encabeza la lista será aquél que se considere el más óptimo.

Las parejas obtenidas deberán ser ordenadas en función de su peso. La inserción de parejas en la lista ordenada se realiza cada vez que se construye una pareja, de tal forma que se va comparando el peso actual de la pareja recién encontrada con los pesos de cada una de las parejas ya almacenadas.

El mecanismo de inserción ordenada consiste en tomar la pareja recién encontrada y comparar su peso con el de la primera de las parejas que aparece en la lista y que será aquélla que hasta el momento poseía un mejor peso.

El hecho de que ciertos parámetros tengan mayor relevancia que otros para la búsqueda del servicio más adecuado, obliga a que la variable peso no sea manejada como algo general sino que ésta sea utilizada independientemente para cada tipo de emparejamiento. De tal forma que se acumulan pesos para salidas, entradas o parámetros no funcionales. En este último caso se considera un único valor resultado de la distinta aportación de la similitud entre parámetros dependiendo de si ésta es de tipo semántico o sintáctico.

La ordenación propuesta da más importancia al peso de las salidas, igual que en el método *sortRule()* de [Pao02a], ya que lo más importante es que el cliente obtenga lo que quiere, que debe ser lo que le proporciona el proveedor mediante las salidas (*outputs*) del servicio. A continuación, no se considera el peso de las entradas sino que se compara antes la suma de los pesos obtenidos con los parámetros de tipo no funcional que son: la proximidad en función de la región o radio geográfico, calidad de servicio, y coincidencia sintáctica del nombre del servicio y del nombre del proveedor (ver figura 7.8).

Posteriormente, se compara el peso de las entradas, ya que éstos se consideran parámetros menos importantes para poder encontrar el SW que aporte lo que el cliente busca, son solamente valores de entrada antes de ejecutar el servicio y obtener de esta manera el beneficio, producto o información que el cliente espera en realidad, es decir, las salidas.
Por último, para deshacer un posible empate en este punto, se considerará el número de veces que se ha obtenido el grado de emparejamiento *FALLO*.

El proceso de **ordenación** se inicia con la comparación entre la pareja recién encontrada y la de la cabeza de la lista. Si el resultado es *false*, querrá decir que el peso no es mejor, por lo que no se habrá encontrado una mejor solución, y por tanto deberá compararse con el resto de la lista e insertarse en el lugar adecuado. En este caso, no se habrá

conseguido encontrar una solución mejor, pero el registro de esta nueva pareja servirá para realizar un sistema tolerante a fallos, ya que en el caso de un fallo en el uso del servicio obtenido como mejor solución, se podrá recurrir al uso de información provista por el elemento siguiente de la lista.

```
ORDENACION(peso1,peso2)
{
      si peso1.pesosalidas > peso2.pesosalidas
            devuelve true
      sino si peso1.pesosalidas < peso2.pesosalidas
            devuelve false
      si peso1.sumaotrosparametros > peso2.sumaotrosparametros
            devuelve true
      sino
            si peso1.pesoentradas > peso2.pesoentradas
                        devuelve true
            sino si peso1.pesoentradas <= peso2.pesoentradas
                        devuelve false
            sino si peso1.numfallos < peso2.numfallos
                        devuelve true
            sino
                        devuelve false
}
```

Figura 7.8 Proceso de ordenación

7.3.1.2 Descripción del proceso completo y diagrama de actividades

El emparejador actúa como un consumidor en un esquema básico productor/consumidor. Permanece dormido en *Match* (figura 7.9) mientras no detecte que se haya insertado un elemento nuevo en la lista de URIs de peticiones. Cuando esto suceda, comienza el proceso de emparejamiento que se inicia con la extracción de la lista de peticiones (URIs), de aquélla que encabece la lista, el matchmaker o emparejador accede al *Profile* del cliente y extrae mediante un *parser*, los conceptos que definen cada uno de los parámetros. Una vez extraídos tanto los parámetros funcionales como los no funcionales, comienza el proceso de emparejamiento (Match) que se describe a continuación. (Ver Figura 7.9).

El proceso consiste en establecer todas las posibles parejas petición-anuncio entre la petición cursada por el cliente y cada uno de los anuncios tomados de la lista de perfiles de SW que resultó del filtrado por el parámetro categoría de servicio. Cada vez que se establece una pareja, se le aplica el método *match* (figura 7.10) que consiste en averiguar cada uno de los pesos parciales relativos a los parámetros de salidas, entradas y parámetros no funcionales, que nos indiquen el grado de similitud semántica entre anuncio y petición. Tanto en la comparación de las salidas como en las entradas (parámetros funcionales) se van comparando cada uno de los parámetros de la petición con los parámetros de su pareja anunciante, haciendo uso de los diferentes grados de similitud.

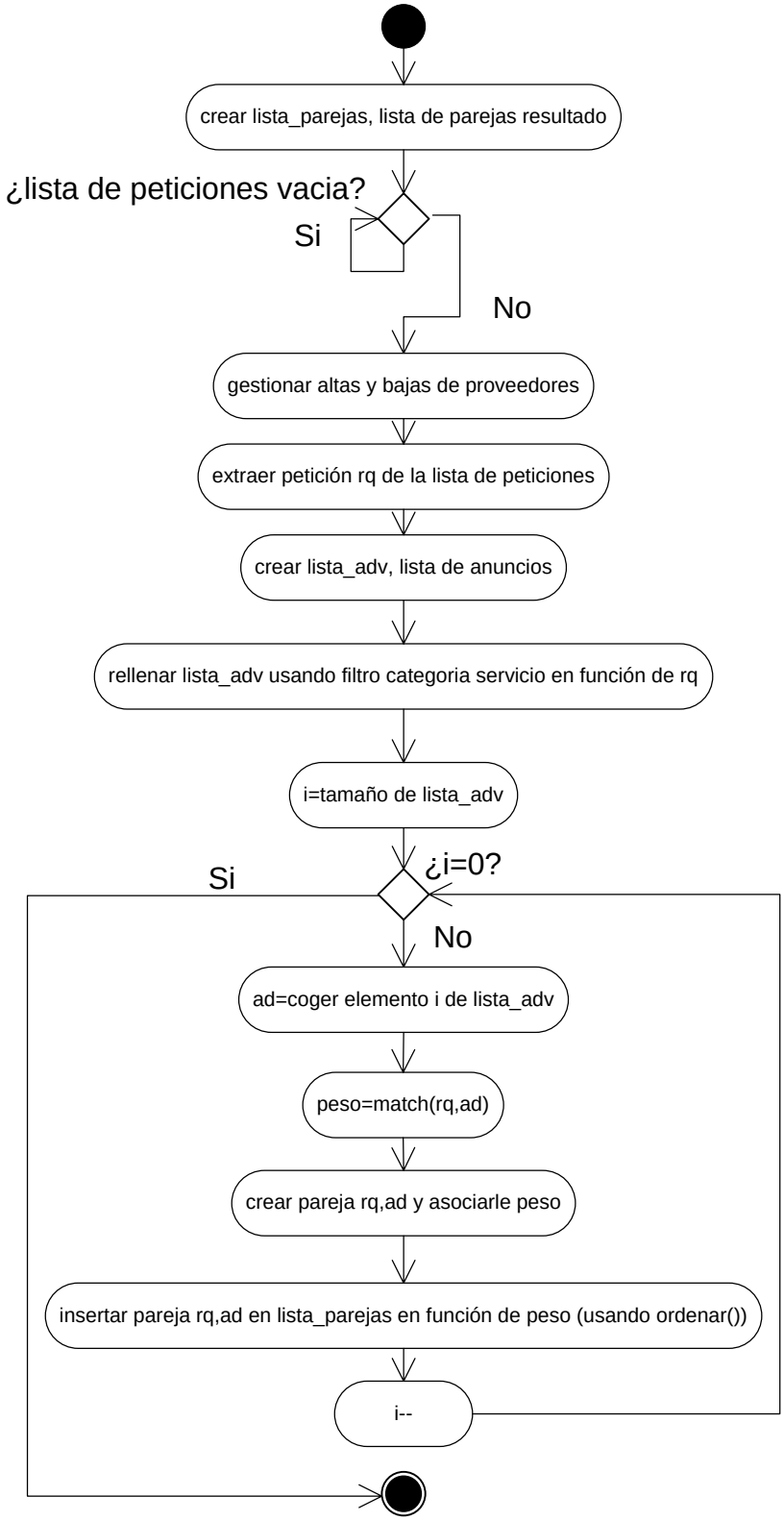


Figura 7.9. Diagrama de actividades del método Match

El proceso de cálculo de pesos se realiza de la siguiente forma: (Ver Figuras 7.10, 7.11)

Partiendo de que el sistema ha cargado inicialmente en un repositorio la ontología de conceptos asociada a la categoría de servicio definida en la petición y en los SW y una vez obtenido un *output* del anuncio y un *output* de la petición, se averigua cuál es el grado de emparejamiento, para lo cual se sigue el siguiente proceso:

- Ejecución de cada una de las consultas (*DegreeOfMatch*, figura 7.7), de mayor a menor grado, esperando a que alguna de las consultas devuelva resultado positivo, o, si no se consigue resultados en las consultas, se llegará al grado de emparejamiento "*Fail*" que representará que no hay emparejamiento entre los parámetros dados. Cada uno de los grados se ha cuantificado de tal forma que el grado obtenido se suma a la variable que mantiene la suma de todos los grados de similitud de los *outputs*.

Al terminar ese proceso para todos los *outputs* definidos por el usuario, y emparejados con todos los *outputs* de los SW mantenidos en el repositorio, los valores obtenidos como resultado son almacenados en la variable **peso** (*peso.salidas*).

En este punto y como mejora de anteriores propuestas, se comprueba si el peso obtenido tiene valor nulo en cuyo caso se abandona el proceso comparativo con el resto de parámetros. Si no fuera así se realizará el proceso anterior pero esta vez para los parámetros funcionales de entradas si los hubiera. El valor resultante será almacenado en *peso.entradas*.

Por último, una vez ya emparejados todos los SW en sus parámetros funcionales de entrada y salida, el siguiente paso es comprobar si estos servicios tienen definidos los parámetros no funcionales que haya especificado el usuario en su petición.

Mediante la implementación de consultas se comparan los parámetros no funcionales de radio geográfico (método *RadioGeoMatch*), calidad del servicio (método *CalidadServicioMatch*) ambos con una aportación de 5 y nombre de servicio (método *NombreServMatch*) y proveedor (método *NombreProvMatch*) con valores de 3 y 2 respectivamente. El resultado global de estas comparaciones es almacenado en la variable *peso.otros*. Al final, cada pareja tendrá asociado un peso que es el que se tomará como medida de idoneidad para la petición cursada.

A continuación y tal como ya hemos comentado, se inserta esta pareja de forma ordenada, haciendo uso del método *ordenacion()* (figura 7.8), en la lista de parejas resultado. Cada elemento de la lista contendrá tres valores que se corresponden con las URI´s tanto de la petición como del anuncio, así como el peso de la pareja.

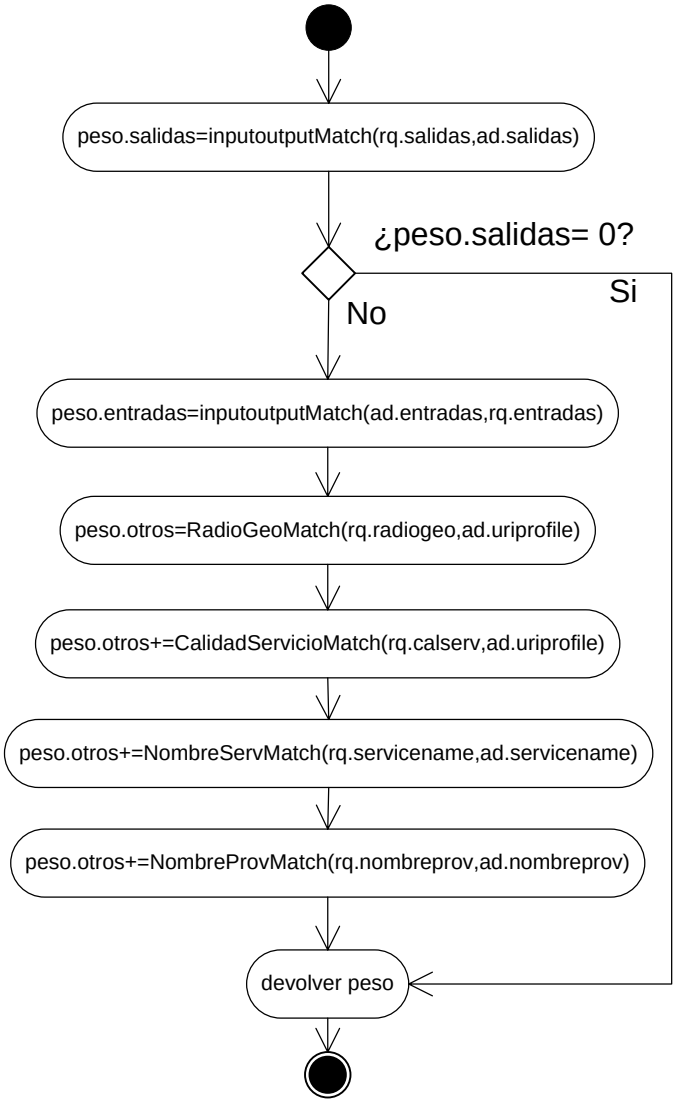


Figura 7.10. Cálculo de pesos del método match

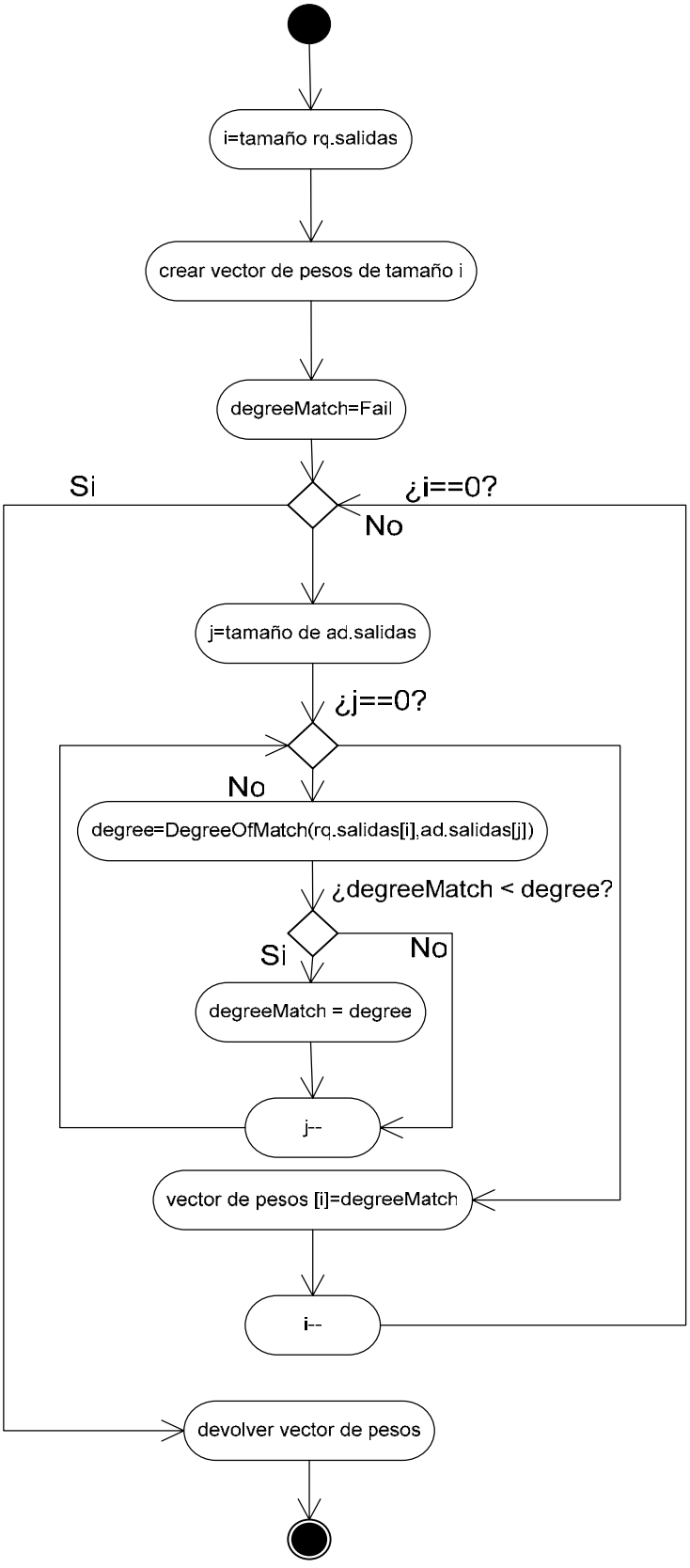


Figura 7.11 Diagrama de actividades del método InputOutputMatch(), para cálculo de “peso.salidas”

7.4. IMPLEMENTACIÓN

El algoritmo fue implementado dentro de un sistema multiagente de descubrimiento de SW de información de tráfico, como parte del trabajo de investigación de esta tesis. Para su desarrollo se utilizaron los siguientes componentes:

- Como repositorio se utilizó Sesame [Bro02], debido a que permite añadir y eliminar información escrita en RDF en los repositorios, y puede almacenar esta información en cualquier base de datos. Sesame soporta como lenguajes de consulta a RQL, RDQL y SeRQL (Sesame RDF Query Language) [SeRQL04], para acceder al conocimiento. Permite la interoperabilidad con un razonador de lógica descriptiva, el cual debe ser otro de los participantes en cualquier sistema emparejador.

- Como razonador se utilizó BOR [BOR02], el cual está basado en lógica descriptiva y tiene soporte para inferencias sobre instancias y sobre conceptos. Este razonador puede ser usado tanto con ontologías escritas en DAML+OIL (con algunas restricciones), y con ontologías escritas en la especificación OWL Lite. Además se puede incorporar a la aplicación Sesame, para dar soporte a ontologías DAML+OIL y poder realizar razonamiento y todas las funciones descritas en la información almacenada en los repositorios.

- Para dar soporte a DAML+OIL se utilizó el plugin de Sesame llamado SeBOR (Sesame+BOR) [BOR02] que acepta modelos de datos semánticos de DAML+OIL en Sesame.

- El sistema de agentes que integra los diferentes componentes fue desarrollado utilizando la metodología FIPA y se implementó en lenguaje Java.

7.4.1 Consultas para los grados de similitud en parámetros funcionales.

En las fases de diseño y posteriormente implementación se especificaron las consultas necesarias para emparejar cada uno de los grados de similitud en la fase de emparejamiento de parámetros funcionales, así como aquéllas utilizadas para comprobar el emparejamiento de los parámetros no funcionales que se emplean en el algoritmo.

A partir del estudio realizado de los lenguajes disponibles se utilizó SeRQL, debido a que era el que mejor se integraba con el razonador y repositorio seleccionado.

Las consultas implementadas para obtención de grados de similitud e implementadas en los métodos Iguales(), SubClassOf(), Subsumes() y Hermanos() (figuras 7.1-7.6) y utilizadas en el proceso de asignación de grados *DegreeOfMatch* (ver figura 7.7) son las siguientes:

– Iguales(),:

```
Select X, Y
From {Aux1} <rdfs:subClassOf> {X},
     {Aux2} <rdfs:subClassOf> {Y}
Where  Aux1 = Aux2
and  X = Y
and  X = <URI del concepto semántico 1>
and  Y = <URI del concepto semántico 2>
```

Este método devolverá *cierto* si esta consulta devuelve alguna tupla, lo que indicará que el concepto cliente es equivalente al concepto proveedor.

– *SubClassOf**(concepto1, concepto1)***, que comprueba si dados dos conceptos, el primero es subclase directa del segundo :

```
Select X, Y
From {X} <daml:subClassOf> {Y}
Where X = <URI del concepto semántico 1>
  And Y = <URI del concepto semántico 2>
```

Este método devolverá *cierto* si esta consulta devuelve alguna tupla.

– *Subsumes**(concepto1, concepto2)***,:

```
Select X, Padre, Abuelo
From {X} <daml:subClassOf> {Padre},
     {Padre} <daml:subClassOf> {Abuelo}
Where X = <URI del concepto semántico 1>
And Abuelo=<URI del concepto semántico 2>
```

Este método devolverá *cierto* si esta consulta devuelve alguna tupla, lo que indicará que el concepto 1 se encuentra dentro del subárbol de conceptos que cuelga del concepto 2.

- *ChermanoP*: Esta consulta consiste en tomar las restricciones del concepto del cliente, para después navegar por las restricciones de los hermanos de esta clase, para encontrar restricciones que emparejen.

Para comprobar este tipo de grado, se han implementado dos consultas, según el cuantificador utilizado en la restricción.

```
CONSULTA PARA COMPROBAR LOS MATCH DE LOS HERMANOS CON COINCIDENCIAS
DEL TIPO TOCLASS

Select distinct X, Y, Prop1, Z
From {X} <daml:subClassOf> {Padre},
{Y} <daml:subClassOf> {Padre},
{X} <rdfs:subClassOf> {Rest},
{Y} <rdfs:subClassOf> {Rest2},
{Rest} <rdf:type> {<daml:Restriction>},
{Rest2} <rdf:type> {<daml:Restriction>},
{Rest} <daml:onProperty> {Prop1},
{Rest2} <daml:onProperty> {Prop1},
{Rest} Z {A},{Rest2} Z {B}
Where X = <URI del concepto semántico del proveedor>
AND Y = < URI del concepto semántico del cliente >
And A = B
And Z = <daml:toClass>


CONSULTA PARA COMPROBAR LOS MATCH DE LOS HERMANOS CON COINCIDENCIAS
DEL TIPO HASCLASS

Select distinct X, Y, Prop1, Z
From {X} <daml:subClassOf> {Padre},
{Y} <daml:subClassOf> {Padre},
{X} <rdfs:subClassOf> {Rest},
{Y} <rdfs:subClassOf> {Rest2},
{Rest} <rdf:type> {<daml:Restriction>},
{Rest2} <rdf:type> {<daml:Restriction>},
{Rest} <daml:onProperty> {Prop1},
{Rest2} <daml:onProperty> {Prop1},
{Rest} Z {A},{Rest2} Z {B}
Where X = <URI del concepto semántico del proveedor>
AND Y = < URI del concepto semántico del cliente >
And A = B
And Z = <daml:hasClass>
```

Las consultas comprueban que ambos conceptos son hijos del mismo padre, que ambas poseen restricciones y que estas restricciones son sobre una propiedad común a los dos conceptos, a su vez comprueban que el predicado y el valor objeto que poseen estas restricciones coincidan. Una consulta nos sirve para comprobar coincidencias en las restricciones del tipo *hasClass* y la otra consulta nos sirve para comprobar las coincidencias en restricciones del tipo *toClass*. Dadas estas premisas y condiciones, ambas consultas nos devuelven una lista con las URI's de ambos conceptos, y con las URI's de las propiedades comunes que poseen las mismas restricciones.

ChermanoP aporta como máximo 100 puntos en la suma total. De esta forma, podemos dar hasta 99 puntos, ya que cada restricción común valdrá 1 punto (cada línea de resultado en cualquiera de las dos consultas), de tal forma, que aquel concepto hermano que tenga más restricciones en propiedades comunes respecto un concepto dado por el cliente será elegido antes que otro concepto hermano con menos coincidencias con el concepto cliente.

7.4.2 Consultas para parámetros no funcionales. Coincidencia exacta.

En relación con los parámetros no funcionales se implementaron consultas relacionadas con categoría de servicio, radio geográfico, calidad de servicio, nombre de servicio y nombre de proveedor. Han sido utilizadas en los métodos auxiliares de *match* (Ver figura 7.11)

filtro para región:

```
Select Region
From {ProfileProveedor} <profile:serviceParameter> {Geopar},
     {GeoPar} <profile:sParameter> {Region}
Where Region = <URI de la región geográfica dada por el
                cliente>
And ProfileProveedor = <URI del Profile del servicio Web al
                   que se esta comparando con la petición>
```

Si el resultado es positivo, la consulta nos devuelve la URI de la región geográfica, y si es negativo, la consulta no devuelve resultado alguno.

filtro para calidad de servicio:

```
Select Calidad
From {A} <profile:qualityRating> {Ratparam},
           {RatParam} <profile:rating> {Calidad}
Where Calidad = <URI de la calidad especificada por el
                  cliente>
And A = <URI del Profile del servicio Web al que se esta
         comparando con la petición >
```

Si se obtiene como resultado de la consulta el grado que proporcionó el cliente, es debido a que coincide con el que aporta el proveedor.

filtro para nombre de servicio:

```
Select  Nombre
From {A} <profile:serviceName> {Nombre}
Where  A = <URI del Profile del servicio Web al que se esta
           comparando con la petición >
And Nombre = "Nombre dado por el cliente"
```

Como en las anteriores consultas, si devuelve como resultado el nombre de servicio que le proporciona el cliente es porque coincide con el del proveedor.

filtro para nombre del proveedor:

```
Select  Nombre_empresa
From {A} <profile:contactInformation> {Empresa},
     {Empresa} <profile:name> {Nombre_empresa}
Where  A = <URI del Profile del servicio web al que se esta
           comparando con la petición >
And Nombre_empresa = "Nombre de la empresa dado por el cliente"
```

Si como resultado se devuelve el nombre de la empresa, es porque la empresa proveedora es la que indicó el cliente.

7.5 PROPUESTA DE MEJORA PARA EMPAREJAMIENTO DE CONCEPTOS HERMANOS EN EL ALGORITMO.

Una mejora que se puede dar al algoritmo de emparejamiento, en la fase de emparejamiento de conceptos hermanos consiste en diferenciar el tipo de restricciones en las que son comunes dos hermanos, y dar más prioridad a un tipo de restricción respecto al resto.

Recordemos que hay dos tipos de restricciones principales que se pueden hacer sobre las propiedades que caracterizan a las clases: *daml*:*hasClass / owl:allValuesFrom* (cuantificador existencial) y *daml:toClass* / *owl:someValuesFrom* (cuantificador universal).

El operador *daml:toClass* indica que los valores que se pueden dar para una determinada propiedad deben de ser únicamente de ese tipo (el especificado en el rango de la propiedad), o dicho de otro modo, la propiedad es restringida a valores de ese tipo, pero pueden existir miembros de esa clase que no tengan esa propiedad y sin embargo hagan cierta esa restricción. Por otra parte el operador *daml:hasClass* es similar, pero menos restrictivo [OWL02]. El uso de este operador para construir la restricción de una propiedad dada, indicará que al menos existirá un valor del tipo dado como rango, pero el resto de valores para esa propiedad pueden ser de cualquier otro tipo.

Se pueden dar diferentes casos cuando dos conceptos hermanos comparan entre sí las diferentes restricciones que pueden tener en común, y hay que dar un peso u otro en el algoritmo de emparejamiento según el nivel de emparejamiento que tengan. Por este motivo, podemos establecer una clasificación dependiendo de las especificaciones de la petición y de las de los diferentes proveedores potenciales:

Caso	CLIENTE	PROVEEDOR1	PROVEEDOR2	PROVEEDOR3
Caso 1	Concepto R1: (hasClass)	ConceptoHermano R1: (hasClass)	ConceptoHermano R1: (toClass)	ConceptoHermano R1: (hasClass) R2: (toClass)
Caso 2	Concepto R1: (toClass)	ConceptoHermano R1: (hasClass)	ConceptoHermano R1: (toClass)	ConceptoHermano R1: (hasClass) R2: (toClass)
Caso 3	Concepto R1: (hasClass) R2: (toClass)	ConceptoHermano R1: (hasClass)	ConceptoHermano R1: (toClass)	ConceptoHermano R1: (hasClass) R2: (toClass)

Tabla 7.1: Casos de emparejamiento de conceptos hermanos

Para cualquiera de los tres casos identificados en la tabla 7.1 se toma la misma decisión de considerar siempre como mejor respuesta, aquélla que coincide exactamente con la de la petición cursada.

El ***caso 1*** y el ***caso 2*** de la tabla, coinciden en que el cliente solicita un requerimiento cuyo parámetro viene descrito por un concepto cuya definición posee una sola restricción en cuya construcción solo se ha utilizado en único cuantificador (existencial para el primero y universal para el segundo). En el lado de los proveedores pueden encontrarse tres tipos de conceptos hermanos, uno que posee el mismo tipo de restricción que el cliente, otro concepto hermano que posee una restricción con diferente operador sobre la misma propiedad y rango, y por último un concepto hermano que posee dos restricciones, una de cada tipo, sobre la misma propiedad que la restricción del concepto cliente. En estos casos, dado el concepto cliente, cuando el emparejador otorgue la puntuación en el emparejamiento con los diferentes proveedores en el grado hermano, tal y como ya hemos comentado, escogerá aquél con coincidencia exacta, y a continuación, el siguiente proveedor con mayor peso debería ser el correspondiente al concepto que posee las dos restricciones (una de las cuales coincide con la del cliente) sobre la propiedad, ya que cumple parte de las expectativas del cliente, el cual solo espera que exista al menos una.

Por último, en el ***caso 3***, el concepto proporcionado por el cliente tiene los dos tipos de restricciones (*daml:hasClass*, *daml:toClass*). Esto significa que "algún valor para la propiedad será del tipo especificado en el rango de valores" y además "los únicos valores para la propiedad serán del tipo dado en dicho rango".

Siguiendo el mismo criterio de "mayor peso al concepto exactamente igual", el algoritmo deberá dar más prioridad al concepto proporcionado por el Proveedor3. La decisión sobre que proveedor debe ser escogido a continuación, queda resuelta por el hecho que apuntábamos al principio de que las restricciones existenciales son menos restrictivas que las universales. Por este motivo, como ya se ha comentado anteriormente en este capítulo al describir los grados de similitud, se dará un mayor peso a conceptos (de proveedores) más generales o menos restrictivos (Proveedor1).

La tabla 7.2 sirve como base para definir el grupo de consultas SeRQL que se debe emplear para comprobar cada uno de los diferentes tipos de emparejamientos que se pueden dar. Los casos descritos en la tabla 7.2 son los mismos que los de la tabla 7.1 (Las consultas se detallan en el anexo.)

Caso	CLIENTE	PROVEEDOR1	PROVEEDOR2	PROVEEDOR3
Caso 1	Concepto R1: (hasClass) C1, C2, C3	ConceptoHermano R1: (hasClass) C2	ConceptoHermano R1: (toClass) C1	ConceptoHermano R1: (hasClass) R2: (toClass) C3
Caso 2	Concepto R1: (toClass) C1, C2', C3'	ConceptoHermano R1: (hasClass) C1	ConceptoHermano R1: (toClass) C2'	ConceptoHermano R1: (hasClass) R2: (toClass) C3'
Caso 3	Concepto R1: (hasClass) R2: (toClass) C3 ,C3', C4	ConceptoHermano R1: (hasClass) C3	ConceptoHermano R1: (toClass) C3'	ConceptoHermano R1: (hasClass) R2: (toClass) C4

Tabla 7.2: Consultas SeRQL para cada caso

Tomando como base la tabla descrita anteriormente y partiendo del concepto proporcionado es posible determinar las restricciones asociadas para poder determinar con cual de los casos definidos en las tablas anteriores nos encontramos, para cada una de las propiedades que posean restricciones.

Para cada uno de estos casos, el algoritmo de emparejamiento se ejecutaría según la prioridad o peso considerados, concediendo una puntuación de 3, 2 y 1 respectivamente, según el orden.

Caso 1: Consultas C2, C3 y por último C1.
Caso 2: Consultas C2', C3' y por último C1.
Caso 3: Consultas C4, C3 y en último lugar la consulta C3'.

Una vez estudiados todos los casos, se ha podido constatar la gran dificultad que lleva consigo, la distinción de las restricciones que puede tener definidas el concepto proporcionado por el cliente. Este análisis debe ser llevado a cabo, antes del comienzo de la fase de emparejamiento, lo cual a su vez ralentiza el proceso de búsqueda.

Para cada uno de los tres diferentes casos posibles que se pueden encontrar, se deben de ejecutar tres consultas diferentes siguiendo un orden, comparando con todos los proveedores en cada una de ellas.

Por tanto, el beneficio que se puede tener en eficacia con esta mejora, respecto la gran complejidad y pérdida en velocidad de ejecución, hace que por ahora esta segunda posibilidad de mejora de la fase de emparejamiento del grado hermano quede descartada.

7.6 PRUEBAS ORIENTADAS A LA COMPROBACIÓN FUNCIONAL DEL EMPAREJADOR SEMÁNTICO

7.6.1 Fundamentos de las pruebas

Se han realizado diferentes pruebas en las que se ha comprobado la mejor eficiencia del algoritmo de emparejamiento propuesto frente al algoritmo de Paolucci et al[34]. En estas pruebas se mostrará que, gracias a la mejor utilización de los parámetros de la clase *Profile* de la ontología de OWL-S / DAML-S, el algoritmo propuesto ofrece mejores resultados, en cuanto a precisión en las búsquedas de servicios, que el algoritmo proporcionado por Paolucci et al., y a su vez, los tiempos de respuesta no sufren cambios de consideración salvo en situaciones extremas (casos peor y mejor).

Para realizar las pruebas se construyen diferentes perfiles de servicios muy parecidos entre sí, y ante diferentes peticiones del cliente se estudian los distintos resultados dados por los dos algoritmos tras su ejecución.

[34] Implementación del pseudocódigo publicado en [Pao02a].

7.6.2 Definición de los casos de prueba.

Para comparar los dos algoritmos a continuación se explica en qué han consistido los diferentes casos de prueba y el objetivo perseguido:

Caso 1: Diferenciación grado exacto del de subclase directa.
Para comprobar que realmente nuestro algoritmo diferencia entre el grado exacto y el de subclase directa, se expondrá un ejemplo en el que el algoritmo de Paolucci et al. puede devolver un anuncio de un proveedor cuya salida es un concepto subclase directa, aún existiendo otros proveedores con grado de similitud exacto en los parámetros de salida. Con esta prueba se determina que nuestra propuesta da como resultado un servicio más adecuado a los requerimientos expresados en la petición que lo que pudiera dar el algoritmo comparado.

Caso 2: Relaciones de parentesco fraternales
Paolucci et al. no distinguen el grado *fraternal*, por lo que las pruebas han ido dirigidas a comprobar que aunque el servicio que más se asemeja al buscado posee parámetros que pertenecen a esta relación, el algoritmo estudiado será incapaz de obtener un resultado y por tanto, devolver un perfil correspondiente a un servicio. Se comprobará que el algoritmo propuesto sí que localiza un proveedor de un concepto hermano al solicitado por el cliente.

Caso 3: Relaciones de subclase.
Paolucci et al. en su algoritmo plantean el grado exacto como aquél en el que el anuncio del proveedor se corresponde con el mismo concepto que el del cliente, o cuando el concepto cliente es subclase directa del concepto proveedor. Por tanto el algoritmo propuesto se comportará de la misma forma en situaciones en las que el anuncio este compuesto por conceptos *padre* en la taxonomía, pero el comportamiento será diferente cuando no existan conceptos *padre* de anuncios y si que los haya simultáneamente de tipo *abuelo* e *hijo* del concepto cliente. En nuestro algoritmo consideramos que debe tener más prioridad aquel anuncio de proveedor cuyos conceptos sean subclase directa del concepto cliente respecto a los anuncios de proveedores más generales (abuelo, bisabuelo etc.), ya que se encuentran a una distancia mayor dentro de la taxonomía que posee la ontología.

Caso 4: Relaciones de subsumción[35].
Distinguir relaciones de subsumción teniendo en cuenta el factor de proximidad, entendida como la distancia en la taxonomía inferior o igual a dos.

Caso 5: Comparación de radio geográfico
En este escenario, se tendrá en cuenta el uso del parámetro no funcional de *radio geográfico*. Pensemos en la solicitud de un servicio de información de incidencias de tráfico. En este caso concreto, si en el repositorio están dados de alta los perfiles del servicio de información de incidencias de la Dirección General de Tráfico y el

[35] Plug-in o Contained: Cuando los conceptos en el anuncio A incluyen los de la petición P. (no se considera que exista una relación de subclase); Subsume o Container: Cuando los conceptos en la petición incluyen los del anuncio.

servicio de incidencias del Servei Català de Trànsit (SCT), ambos caracterizados por el parámetro de salida "Incidente", el algoritmo de Paolucci no permite distinguir entre ellos dos ante una petición del cliente. Por el contrario, el algoritmo propuesto, si que es capaz de diferenciarlos, mediante el uso del parámetro *radio geográfico.* Para ello, los proveedores de la DGT deben de haber previamente especificado este parámetro con valor de *España* correspondiente a la ontología de geografía, y el proveedor SCT con el valor de *Cataluña*. Gracias a estas especificaciones por parte de los proveedores, cada vez que un cliente determina este parámetro en sus requerimientos la búsqueda, ésta se podrá ajustar más a la petición cursada.

Caso 6: Exclusión de conceptos a distancias mayores de 2 en relaciones de subsumción. Validez de considerar tipo de relaciones fraternales en lugar de relaciones de subsumción con distancias grandes en el árbol taxonómico de conceptos.

Este caso guarda relación con el caso 4 anteriormente comentado.

El algoritmo de Paolucci otorga el mismo valor o grado de similitud a conceptos (descendientes en la jerarquía) cuya distancia en el árbol es de solamente dos saltos que aquéllos otros que estén a una distancia mayor, y que por lo tanto se consideren mucho más restrictivos.

En este escenario se comprobará que la devolución de ciertos resultados sin la imposición de un límite máximo en la distancia en el árbol, puede dar como resultado conceptos (perfiles de servicio) que apenas guarden características en común con el concepto requerido. Consideraremos en estos casos el hecho de ser más adecuado, la devolución como resultado de conceptos con relación fraternal.

Caso 7: Comparación por radio de calidad de servicio[36]

Aquí comprobaremos que el uso de un parámetro no funcional, como es la calidad de servicio, puede deshacer un posible empate entre dos proveedores seleccionados tras un emparejamiento previo de parámetros funcionales.

Por último, trataremos de abordar como el grado de similitud fraternal puede ser estudiado más ampliamente de manera que la búsqueda de perfiles dependientes de este tipo de relación, pueda ser todavía más precisa que el hecho de considerar simplemente que dos conceptos hermanos compartan una sola restricción sobre una propiedad. Este caso de prueba, no trata de comparar los dos algoritmos en cuestión, sino que únicamente pretende comprobar una optimización en este tipo de relación.

Caso 8: Diferenciación de grados en la relación fraternal

Se tratará de constatar que la consideración del número de restricciones en común entre dos conceptos hermanos, es un factor importante a tener en cuenta en la obtención de un perfil adecuado.

[36] Una propuesta de escala para este tipo de parámetro es descrita en el capítulo 8.

7.6.3 Distintos escenarios propuestos y resultados obtenidos.

A continuación vemos un cuadro resumen (tabla 7.3) de los perfiles de servicios usados en los distintos escenarios, identificando sus parámetros y tipos utilizados para cada uno de ellos. Dichos perfiles han sido utilizados en las diferentes pruebas realizadas, lo que nos han permitido validar nuestras hipótesis para cada uno de los casos de uso descritos anteriormente.

InfovozProfile
INPUTS: Usuario: *XMLSchema.xsd#string* Password: *XMLSchema.xsd#string* Carretera: *Vias.daml#Via* Kminicio: *XMLSchema.xsd#decimal* Kmfin: *XMLSchema.xsd#decimal* Sentido: *Relaciones_sucesos.daml#Sentido* Dias: *Time.daml#DayofWeek* Hora_ini: *Time.daml#Time* Hora_fin: *Time.daml#Time* **OUTPUTS:** Confirmacion: *XMLSchema.xsd#String* Alerta: *Cruta2.daml#Alerta_SMS* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Excelente*
TraficoCatProfile
INPUTS: Incidencia: *Sucesos.daml#Tipo_Incidente* Ordenación: *TraficoCatProcess.daml#Orden* Demarcacion*: Geografia.daml#Provincia* Comarca: *Geografia.daml#Comarca* **OUTPUTS:***Incidenciaout: Sucesos.daml#Incidente* **Radio Geográfico:** *Geografia.daml#Catalunya* **Calidad de Servicio:** *Concepts.daml#Excelente*
TraficoProfile
INPUTS: Incidencia: *Sucesos.daml#Tipo_Incidente* Provincia: *Geografia.daml#Provincia* Comunidades: *Geografia.daml#Comunidad_Autonoma* MostrarInc: *Concepts.daml#MostrarType* **OUTPUTS:** Incidencia: *Sucesos.daml#Incidente* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Excelente*
CaracterProfile
INPUTS: Dia: *Time.daml#Date* **OUTPUTS:**Caracter: *Sucesos.daml#Caracter* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*
PrevisionProfile
INPUTS: Dia: *Time.daml#Date* **OUTPUTS:** Prevision: *Sucesos.daml#Prevision* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*
HijoPrev_1Profile
INPUTS: Dia: *Time.daml#Date* **OUTPUTS:** Salida*: Sucesos.daml# HijoPrev_1* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*

PrognosisProfile **INPUTS:** Dia: *Time.daml#Date* **OUTPUTS:** Prognosis: *Sucesos.daml#Prognosis* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*
FechaAccidenteProfile **INPUTS:** Fecha: *Time.daml#Date* **OUTPUTS:** Salida: *Sucesos.daml#Accidente* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*
MeteorologicoProfile **INPUTS:** Meteorologicoin: *Sucesos.daml#Tipo_Incidente* Provincia: *Geografia.daml#Provincia* Comunidad: *Geografia.daml#Comunidad_Autonoma* Mostrarinc: *Concepts.daml#MostrarType* **OUTPUTS:** Meteorologicoout: *Sucesos.daml#Metereologico* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Excelente*
HijoMeteorologicaProfile **INPUTS:** HijoMeteorologicain: *Sucesos.daml#Tipo_Incidente* Provincia: *Geografia.daml#Provincia* Comunidad: *Geografia.daml#Comunidad_Autonoma* Mostrarinc: *Concepts.daml#MostrarType* **OUTPUTS:**HijoMeteorologicoout: *Sucesos.daml#HijoMetereologica* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Excelente*
SucesoProfile **INPUTS:** Incidencia: *Sucesos.daml#Tipo_Incidente* Provincia: *Geografia.daml#Provincia* Comunidad: *Geografia.daml#Comunidad_Autonoma* Mostrarinc: *Concepts.daml#MostrarType* **OUTPUTS:** Sucesoout: *Sucesos.daml#Suceso* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Excelente*
NietoPrev_1Profile **INPUTS:** Fecha: *Time.daml#Date* **OUTPUTS:** Salida: *Sucesos.daml#NietoPrev_1* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*
BisNietoPrev_1Profile **INPUTS:** Fecha: *Time.daml#Date* **OUTPUTS:** Salida: *Sucesos.daml#BisNietoPrev_1* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** Concepts.daml#Bueno
TraficoCADIZProfile **INPUTS:** Incidencia: *Sucesos.daml#Tipo_incidente* Provincia: *Geografia.daml#Provincia* Comunidades: *Geografia.daml#Comunidad_Autonoma* MostrarInc: *Concepts.daml#MostrarType* **OUTPUTS:** Incidencia: *Sucesos.daml#Incidente* **Radio Geográfico:** *Geografia.daml#CADIZ* **Calidad de Servicio:** *Concepts.daml#Excelente*

CBProfile
INPUTS: Fecha: *Time.daml#Date* **OUTPUTS:** Salida: *Concepts.daml#ConceptoB* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*
CCProfile
INPUTS: Fecha: *Time.daml#Date* **OUTPUTS:** Salida: *Concepts.daml#ConceptoC* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*
CDProfile
INPUTS: Fecha: *Time.daml#Date* **OUTPUTS:** Salida: *Concepts.daml#ConceptoD* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*
PrevisionProfileMalo
INPUTS: Dia: *Time.daml#Date* **OUTPUTS:** Prevision: *Sucesos.daml#Prevision* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Malo*

Tabla 7.3: Perfiles de servicios usados en las pruebas.

En las tablas siguientes se pueden apreciar los diferentes resultados obtenidos en cada uno de los escenarios creados. Es importante resaltar que tal y como se ha expuesto y queda reflejado en la figura 7.10 de este capítulo, el emparejador semántico, después de procesar el emparejamiento entre parámetros de salida, no seguirá procesando el resto de parámetros, si el resultado de los primeros ha sido de fallo. Sin embargo, para una comparación con más nivel de detalle, entre los dos algoritmos objeto de estudio, se ha preferido completar la fase de emparejamientos para el resto de parámetros.

ESCENARIO 1

CLIENTE	PROVEEDORES	PUNTUACIÓN (O/I/Otros)	
		Propuesto	Paolucci
ClientePrevision Profile **INPUTS:** Dia: *Time.daml#Date* **OUTPUTS:** Prevision: *Sucesos.daml#Prevision* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*	InfovozProfile	0/0	0/0
	TraficoCatProfile	0/0	0/0
	TraficoProfile	0/0	0/0
	CaracterProfile	5/6	**6/6**
	PrevisionProfile	**6/6**	6/6
	HijoPrev1_Profile	4/6	0/6

Tabla 7.4: Resultados escenario 1

En este caso se puede observar como ante la petición de un servicio de información sobre previsiones de tráfico, el algoritmo de Paolucci et al. otorga el mismo valor a los anuncios de previsiones de tráfico, que a los anuncios de servicios de

conceptos que son clase padre directa de las previsiones, por lo que ante el empate, el algoritmo de Paolucci no sabe distinguir este pequeño matiz (uno es más exacto que el otro), algo que si se realiza en el algoritmo propuesto, por lo que la distinción es mejor y más exacta.

En la figura 7.12, se puede observar dónde se encuentran localizados en la taxonomía los principales conceptos de los anuncios de los proveedores respecto el que solicita el cliente.

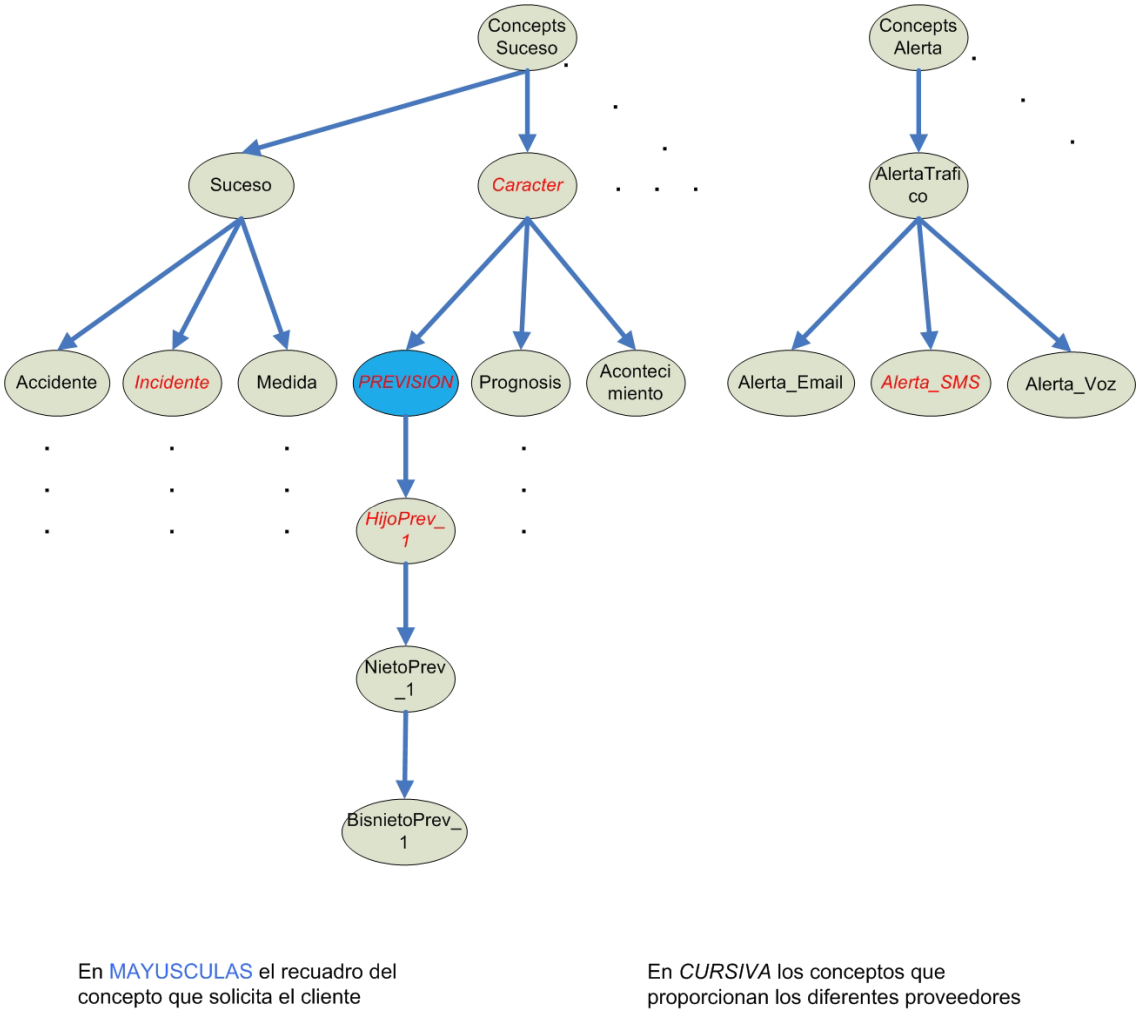


Figura 7.12 Jerarquía de conceptos. Escenario uno.

ESCENARIO 2

CLIENTE	PROVEEDORES	PUNTUACIÓN (O/I/Otros)	
		Propuesto	Paolucci
ClientePrevision Profile **INPUTS:** Dia: *Time.daml#Date* **OUTPUTS:** Prevision: *Sucesos.daml#Prevision* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*	InfovozProfile	0/0	0/0
	TraficoCatProfile	0/0	0/0
	TraficoProfile	0/0	0/0
	PrognosisProfile	**1/6**	0/0
	FechaAccidenteProfile	0/6	**0/6**

Tabla 7.5: Resultados escenario 2

Al no distinguir Paolucci et al. el grado fraternal o de hermanos, puede ocurrir que lo que solicite el cliente no se encuentre disponible mediante la búsqueda por antecedentes ni predecesores del concepto, por lo que en estos casos, es conveniente tener en cuenta en las taxonomías de conceptos, el grado de similitud entre nodos hermanos. En este caso de prueba, se observa su valor.

En la figura 7.13, se puede observar dónde se encuentran localizados en la taxonomía los principales conceptos de los anuncios respecto a lo que solicita el cliente.

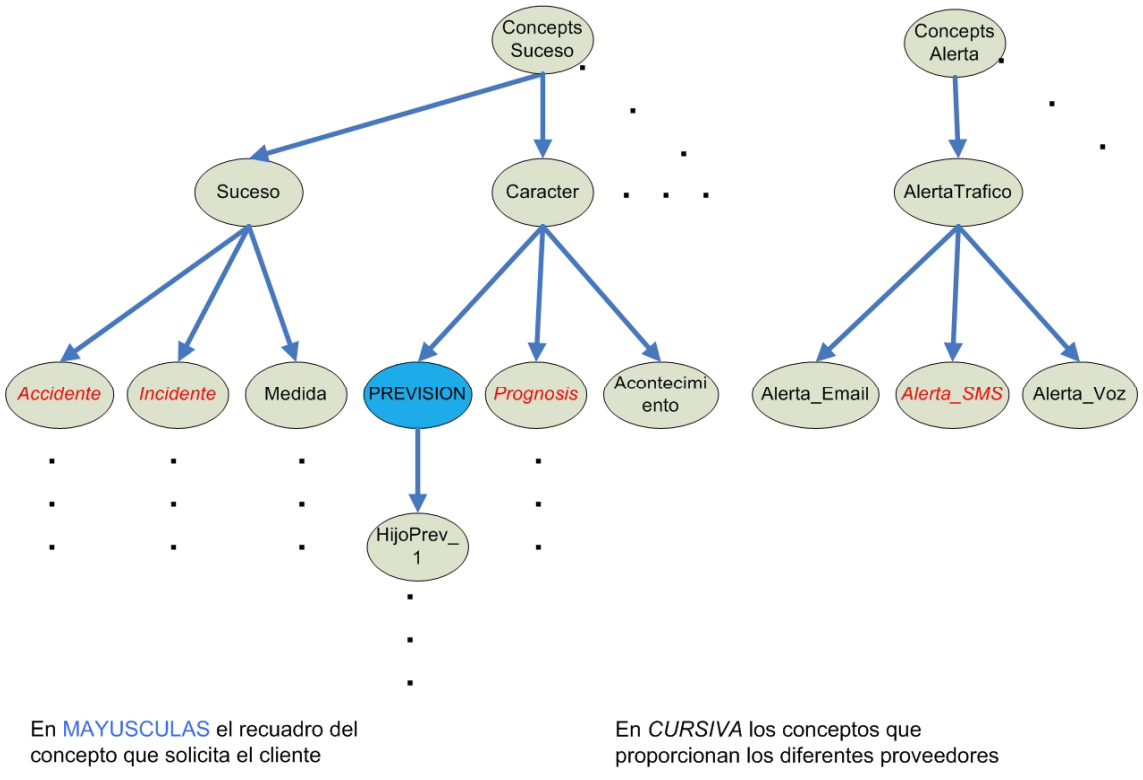


Figura 7.13 Jerarquía de conceptos. Escenario dos.

ESCENARIO 3

Se plantean dos escenarios distintos:

Escenario 3a:

CLIENTE	PROVEEDORES	PUNTUACIÓN (O/I/Otros)	
Meteorologico Profile **INPUTS:** Meteorologicoin: *Sucesos.daml#Tipo_Incidente* Provincia: *Geografia.daml#Provincia* Comunidad: *Geografia.daml#Comunidad_Autonoma* Mostrarinc: *Mostrar_type* **OUTPUTS:** Meteorologicoout: *Sucesos.daml#Metereologico* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Excelente*		Propuesto	Paolucci
	TraficoProfile	**5/6/5**	**6/6**
	PrevisionProfile	0/0/5	0/0
	HijoMeteorologicaProfile	4/6/5	0/6

Tabla 7.6: Resultados escenario 3a

En este caso se puede observar como los dos algoritmos se compartan de la misma manera, dando como resultado el anuncio que más se asemeja (*TraficoProfile*), cuyo concepto es *padre* en la taxonomía.

Escenario 3b*:*

Para este caso, se incluye un anuncio que es abuelo del concepto *Meteorologico* (eliminando *TraficoProfile*), de tal forma que comprobamos que "*Paolucci*" devuelve como resultado un anuncio de proveedor más alejado que el que solicita el cliente.

CLIENTE	PROVEEDORES	PUNTUACIÓN (O/I/Otros)	
MeteorologicoProfile **INPUTS:** Meteorologicoin: *Sucesos.daml#Tipo_Incidente* Provincia: *Geografia.daml#Provincia* Comunidad: *Geografia.daml#Comunidad_Autonoma* Mostrarinc: *Mostrar_type* **OUTPUTS:** Meteorologicoout: *Sucesos.daml#Metereologico* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Excelente*		Propuesto	Paolucci
	SucesoProfile	2/6/5	**2/6**
	PrevisionProfile	0/0/5	0/0
	HijoMeteorologicaProfile	**4/6/5**	0/6

Tabla 7.7: Resultados escenario 3b

En la figura 7.14, se puede observar dónde se encuentran localizados en la taxonomía los principales conceptos de los anuncios respecto a lo que solicita el cliente para el escenario 3b.

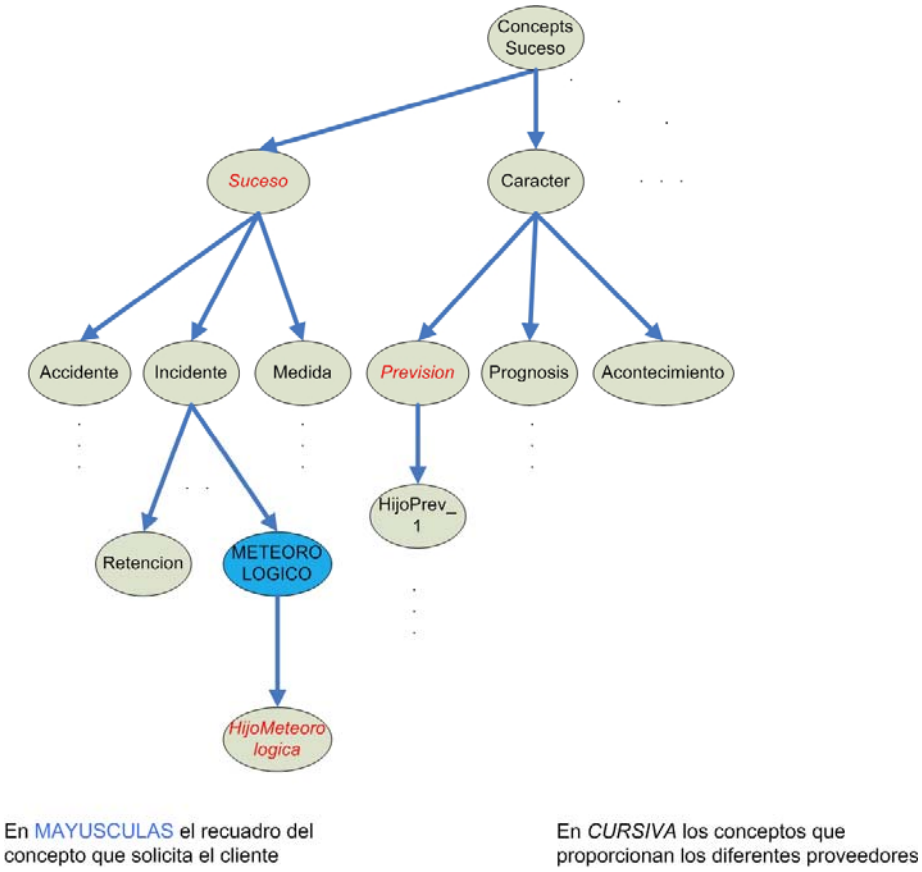


Figura 7.14 Jerarquía de conceptos. Escenario tres.

ESCENARIO 4

CLIENTE	PROVEEDORES	PUNTUACIÓN (O/I/Otros)	
		Propuesto	Paolucci
ClientePrevision Profile **INPUTS:** Dia: *Time.daml#Date* **OUTPUTS:** Prevision: *Sucesos.daml#Prevision* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*	PrognosisProfile	1/6/8	0/6
	InfovozProfile	0/0/5	0/0
	NietoPrev_1Profile	**3/6/8**	3/6
	BisNietoPrev_1 Profile	0/6/8	**3/6**

Tabla 7.8: Resultados escenario 4

Paolucci et al. no distinguen la proximidad en los grados *Subsume* y *Plugin*, les da a todos un mismo valor, mientras que en nuestro caso, se limita la proximidad a una distancia igual a dos. De esta forma, se observa como Paolucci puntúa igual a un concepto nieto que a un bisnieto, mientras que en nuestro caso, al limitar la distancia aseguramos un mayor grado de similitud entre conceptos y la obtención de resultados que no sean falsos positivos.

En la figura 7.15, se puede observar dónde se encuentran localizados en la taxonomía los principales conceptos de los anuncios respecto a lo que solicita el cliente.

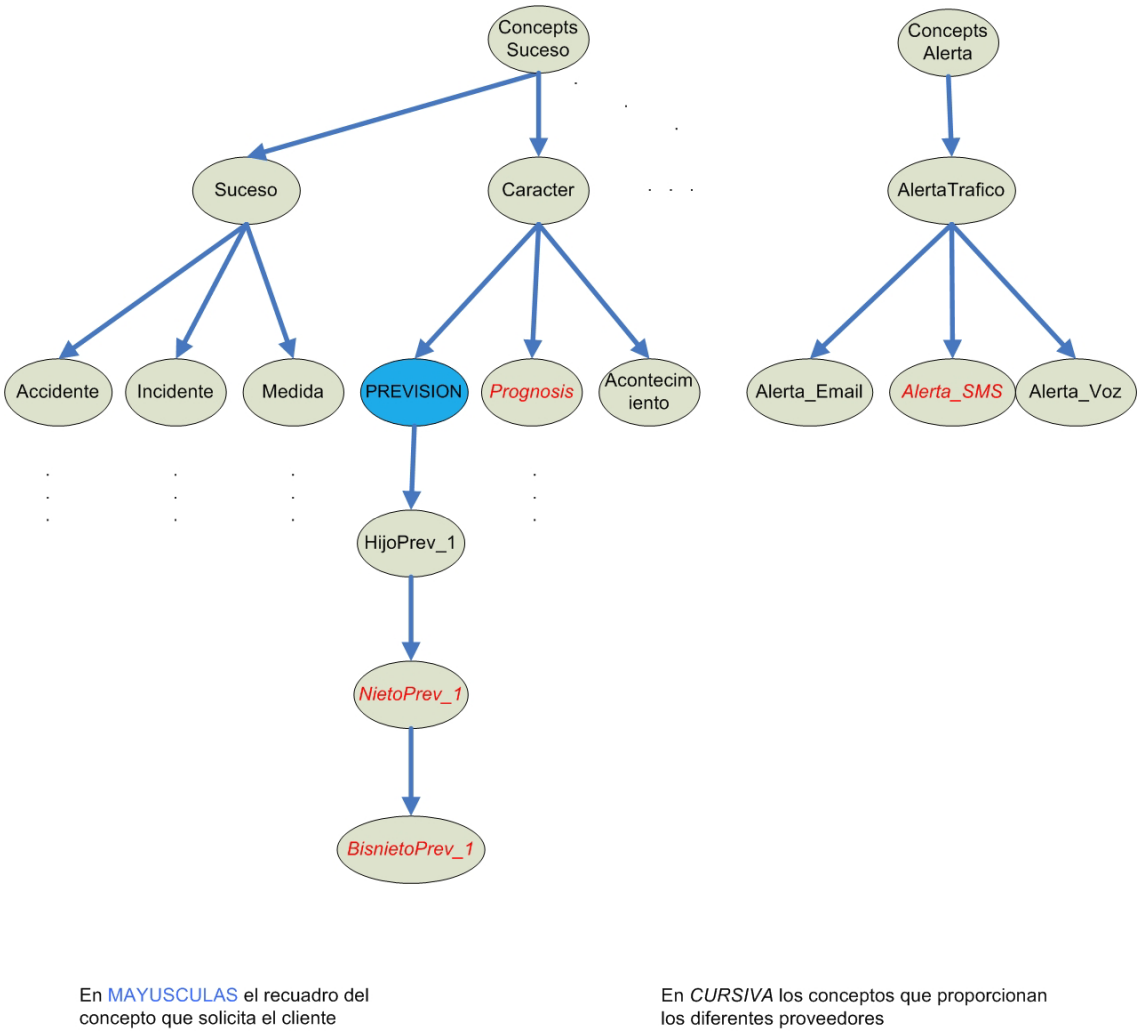


Figura 7.15 Jerarquía de conceptos. Escenario cuatro.

ESCENARIO 5

CLIENTE	PROVEEDORES	PUNTUACIÓN (O/I/Otros)	
TraficoCatProfile		Propuesto	Paolucci
INPUTS: Incidencia: *Sucesos.daml#Tipo_Incidente* Ordenación: *TraficoCatProcess.daml#Orden* Demarcacion*: Geografia.daml#Provincia* Comarca: *Geografia.daml#Comarca* **OUTPUTS:** *Incidenciaout: Sucesos.daml#Incidente* **Radio Geográfico:** *Geografia.daml#Catalunya* **Calidad de Servicio:** *Concepts.daml#Excelente*	TraficoProfile	6/6/0	6/6
	TraficoCatProfile	**6/6/5**	6/6
	TraficoCADIZ	6/6/0	**6/6**

Tabla 7.9: Resultados escenario 5

En este ejemplo se muestra como dado el caso de que se tengan anuncios de perfiles similares, es decir, que ofrezcan prácticamente lo mismo (en este caso información sobre incidencias de tráfico), como la utilización de un parámetro no funcional como es el radio geográfico de actuación del proveedor puede servir para romper el empate entre estos anuncios en el algoritmo de emparejamiento.

ESCENARIO 6

CLIENTE	PROVEEDORES	PUNTUACIÓN (O/I/Otros)	
ClientePrevisionProfile		Propuesto	Paolucci
INPUTS: Dia: *Time.daml#Date* **OUTPUTS:** Prevision: *Sucesos.daml#Prevision* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*	PrognosisProfile	**1/6/5**	0/6
	InfovozProfile	0/6/5	0/6
	BisNietoPrev_1 Profile	0/0/5	**3/6**

Tabla 7.10: Resultados escenario 6

Como ya se ha expuesto, el algoritmo de Paolucci et al. no impone un límite de distancia en los grados *subsume* y *plugin*, por lo que puede darse el caso de que algún proveedor proporcione un concepto muy alejado en la taxonomía, pero subsumido por el concepto que pide el cliente, por lo que su valor semántico pueda ser más distante que el que puede proporcionarnos un anuncio de un concepto hermano del que pide el cliente (el cual no sería considerado por su algoritmo).

En la figura 7.16, se puede observar dónde se encuentran localizados en la taxonomía los principales conceptos de los anuncios respecto a lo que solicita el cliente:

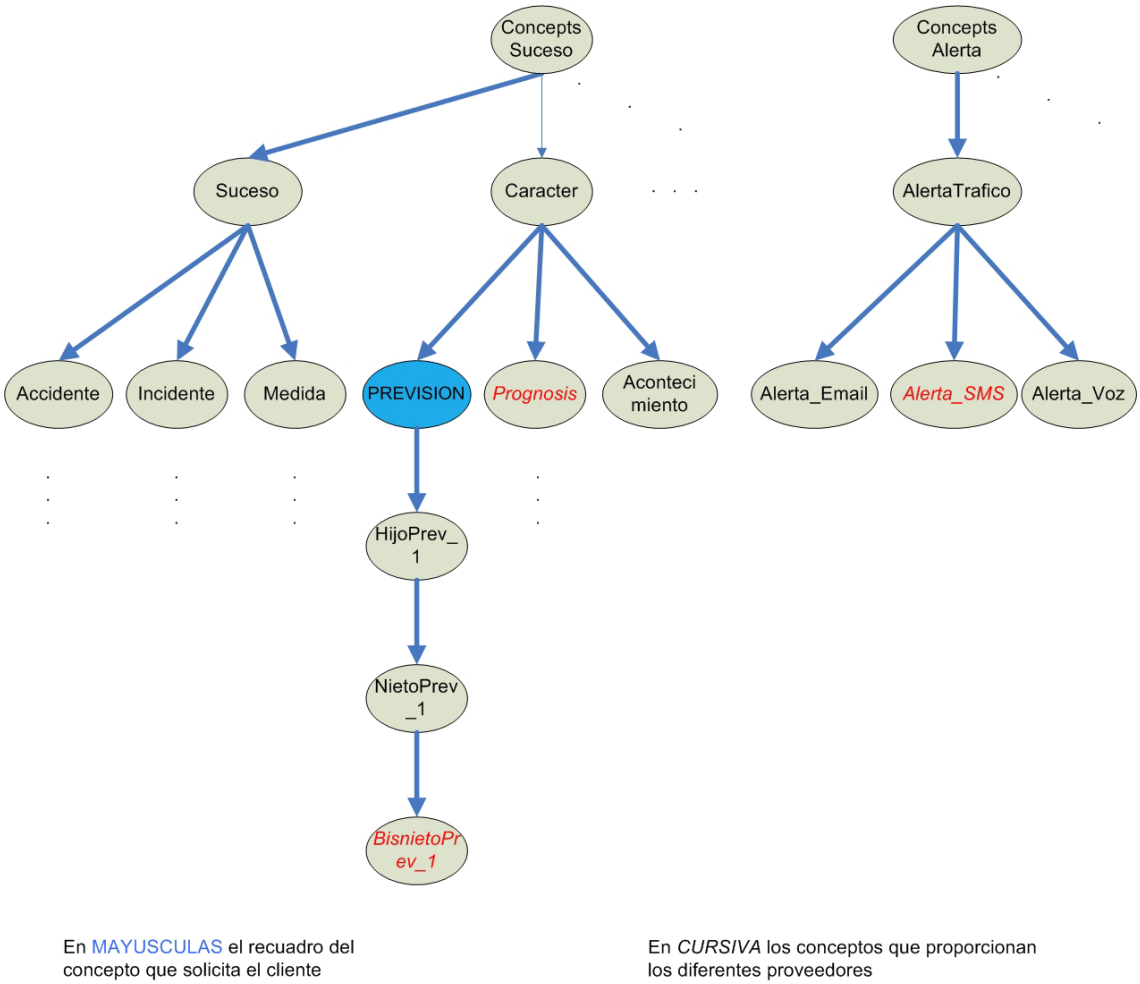


Figura 7.16 Jerarquía de conceptos. Escenario seis.

ESCENARIO 7

CLIENTE	PROVEEDORES	PUNTUACIÓN (O/I/Otros)	
		Propuesto	Paolucci
ClientePrevisionProfile **INPUTS:** Dia: *Time.daml#Date* **OUTPUTS:** Prevision: *Sucesos.daml#Prevision* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*	PrognosisProfile	1/6/8	0/6
	InfovozProfile	0/0/5	0/0
	PrevisionProfile	**6/6/8**	6/6
	PrevisionProfileMalo	6/6/5	**6/6**

Tabla 7.11: Resultados escenario 7

En este ejemplo se muestra como dado el caso de que se tengan anuncios de perfiles similares y que poseen un mismo radio geográfico, como un parámetro no funcional como es la calidad del servicio que proporciona el proveedor puede servir

para romper el empate entre estos proveedores en el algoritmo de emparejamiento propuesto, mientras que Paolucci et al. no tienen en cuenta estos parámetros en su algoritmo para emparejamiento.

En la figura 7.17, se puede observar dónde se encuentran localizados en la taxonomía los principales conceptos de los anuncios respecto a lo que solicita el cliente

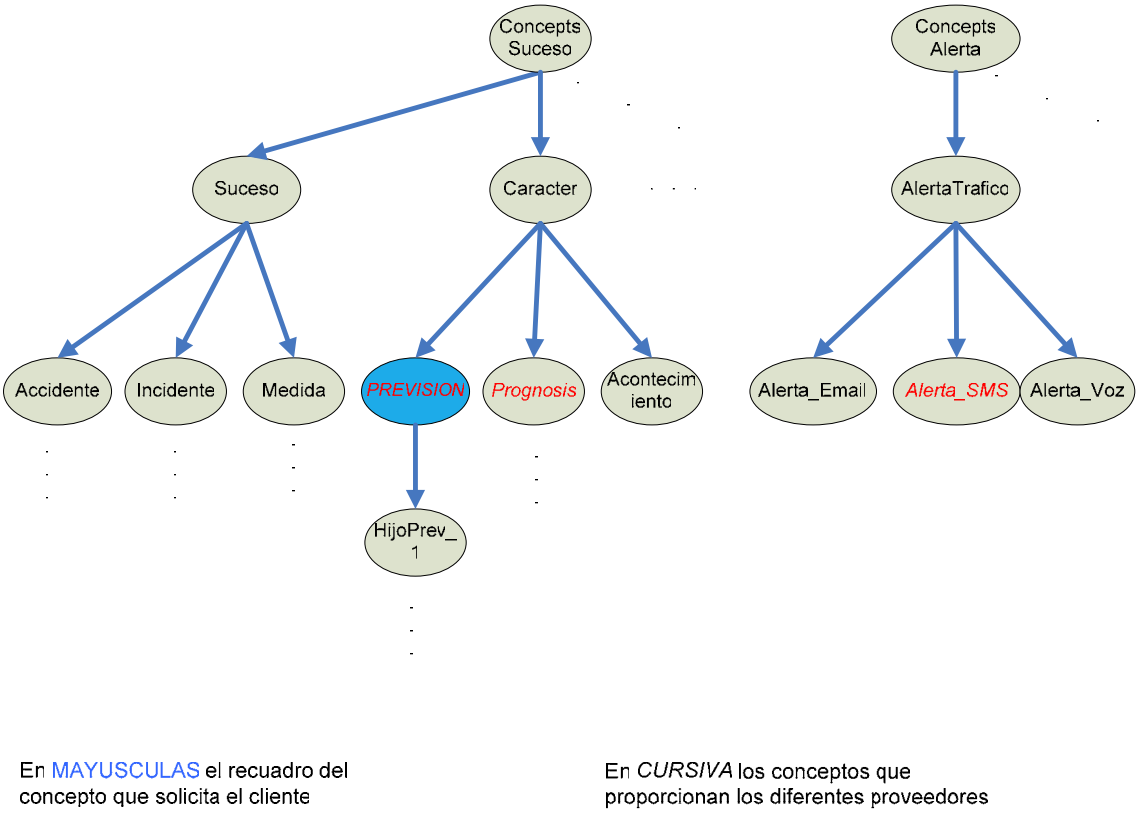


Figura 7.17 Jerarquía de conceptos. Escenario siete.

ESCENARIO 8

CLIENTE	PROVEEDORES	PUNTUACIÓN (O/I/Otros)		COMENTARIOS
CAProfile **INPUTS:** Fecha: *Time.daml#Date* **OUTPUTS:** Salida: *Concepts.daml#ConceptoA* **Radio Geográfico:** *Geografia.daml#Espanya* **Calidad de Servicio:** *Concepts.daml#Bueno*		Propuesto Versión 1.0	Propuesto Versión 1.5	Paolucci
	CBProfile	**1/6/5**	102/600/8	**0/6**
	CCProfile	1/6/5	103/600/8	0/6
	CDProfile	1/6/5	**104/600/8**	0/6

Tabla 7.12: Resultados escenario 8

Se decide probar cómo se incrementa el rendimiento y eficacia del algoritmo para emparejamiento con la primera de las mejoras del grado hermano.

Supongamos los perfiles de esta prueba, que proporcionan conceptos B, C, D hermanos del que solicita el cliente A (ver figura 7.18). Con la versión inicial de nuestro algoritmo, es decir la que solo comprobaba que tuvieran restricciones en común, los tres anuncios de proveedores puntúan igual, es decir, puntúan solo por ser hermanos, mientras que en la versión posterior, que tiene en cuenta el número de coincidencias en las restricciones, se premia a los conceptos hermanos que más coincidencias tienen con las restricciones del concepto que el cliente solicita.[37]

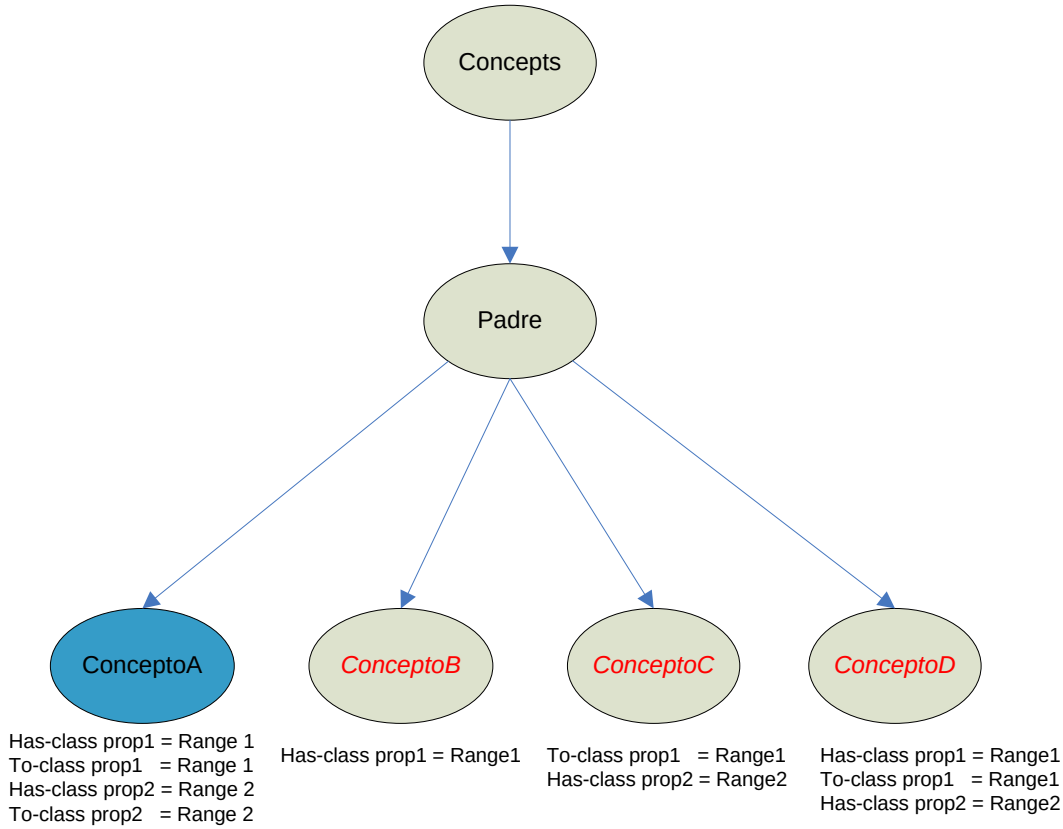


Figura 7.18 Conceptos hermanos. Escenario ocho.

En la figura 7.18 se puede observar como los cuatro conceptos hermanos tienen definidas diferentes restricciones para ciertas propiedades comunes. De esta forma, si el cliente solicitó un anuncio descrito mediante el *conceptoA*, el que más coincidencias posee en las restricciones es el anuncio del *conceptoD*, por lo que a éste se le premia respecto a los otros dos proveedores que tienen menos coincidencias.

7.6.4 Comparativa de los tiempos de respuesta.

Al igual que en las pruebas anteriores, hemos utilizado 2 prototipos, uno de ellos basado en nuestra propuesta de algoritmo de emparejamiento y el otro siguiendo el pseudocódigo propuesto en [Pao02a].

[37] En este caso hay que cambiar la escala de puntuación, de la inicialmente aportada.

Las pruebas se han realizado variando el número de perfiles de servicios anunciados (1, 3, 5, 8, 10, 15, 20, 25, 30 y 33). El tiempo obtenido está mostrado en milisegundos.

Caso 1: Todos los anuncios disponibles pertenecen a la misma categoría de servicio.

Éste es el peor caso para el algoritmo propuesto. Esto es debido a que todos los perfiles de proveedores de servicios insertados en el repositorio pertenecen a la misma categoría de servicio que el servicio buscado por el cliente. En este caso el filtro de categoría de servicio no descarta perfiles, por lo que el emparejamiento se realiza en ambos prototipos con el mismo número de perfiles. Como nuestro método para calcular el grado de emparejamiento es más costoso debido a que hace más comparaciones, el prototipo basado en el algoritmo de Paolucci obtiene el resultado del emparejamiento en menos tiempo. Véase el gráfico de la figura 7.19.

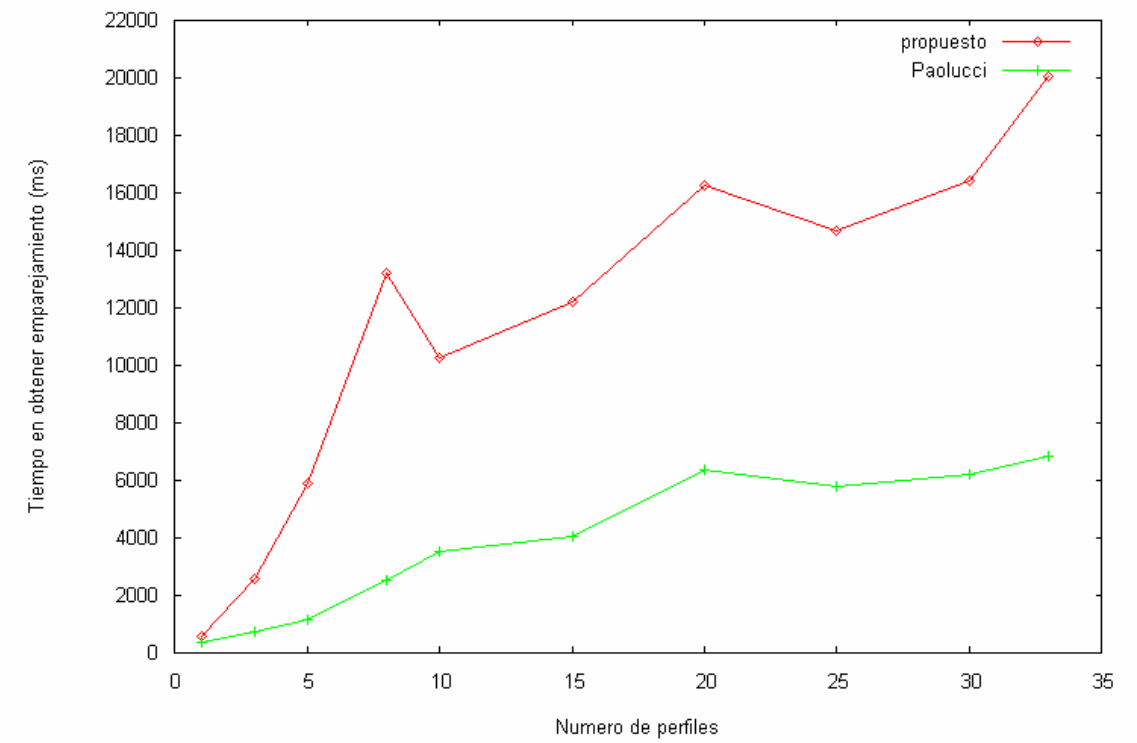


Figura 7.19 Caso 1: todos los anuncios pertenecen a la misma categoría que la petición.

Caso 2: El 20% de los anuncios pertenecen a la misma categoría de servicio buscada por el cliente.

Este caso muestra una situación que se aproxima más a la realidad. En ella no todos los perfiles del repositorio pertenecen a la categoría de servicio a la que pertenece el buscado por el cliente. En este caso hay un 20% de los perfiles que pertenecen a la misma categoría de servicio. Se puede observar mediante el gráfico de la figura 7.20 como nuestra propuesta obtiene tiempos muy similares a la de Paolucci et al.

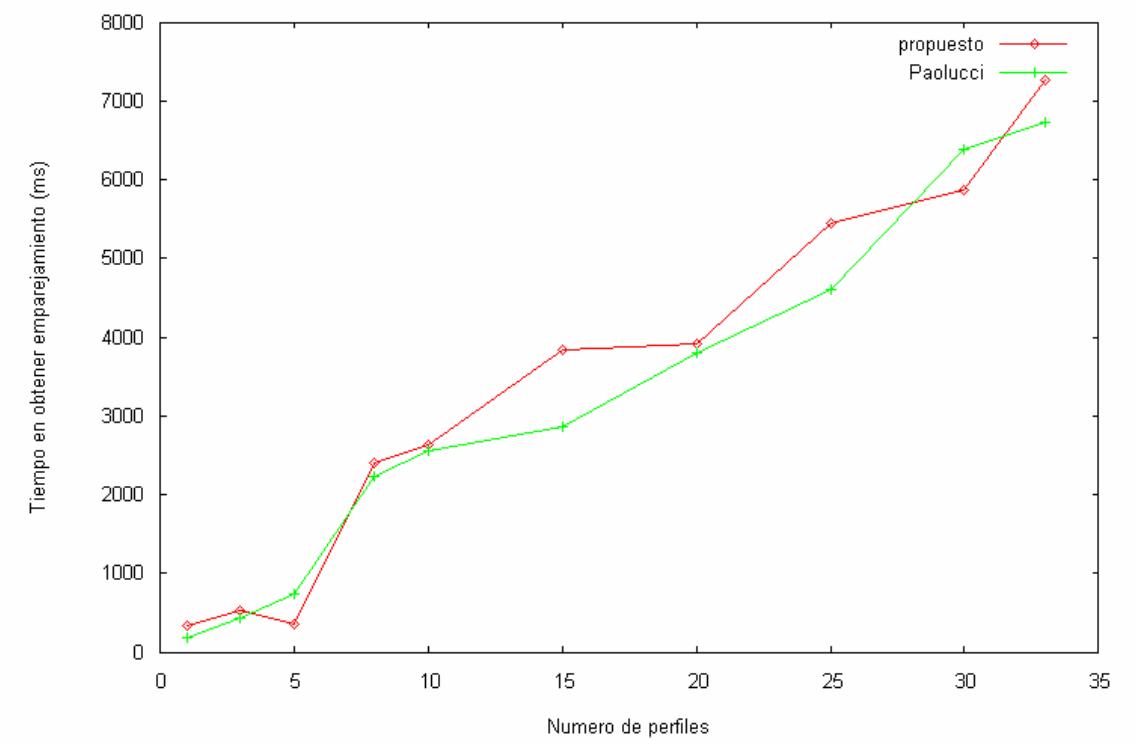


Figura 7.20 Caso 2: solo el 20% de los anuncios pertenecen a la misma categoría que la petición.

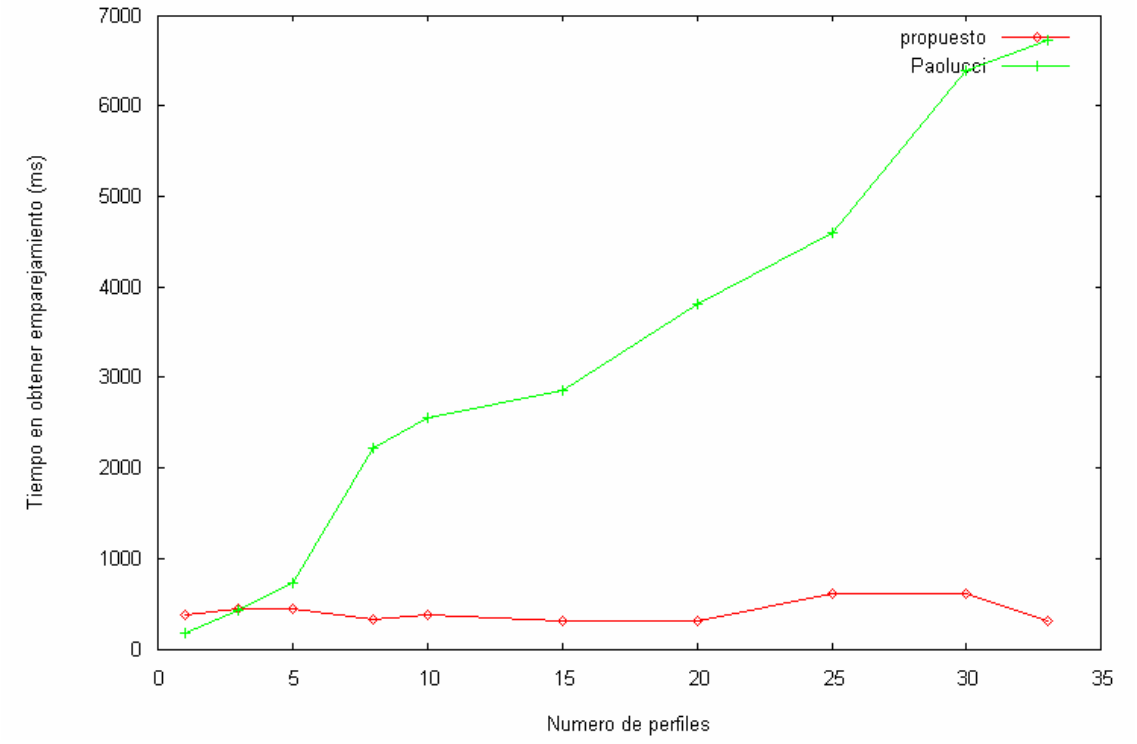


Figura 7.21 Caso 3: solo uno de los anuncios pertenece a la misma categoría.

Caso 3: Solo hay 1 servicio que pertenezca a la categoría de servicio buscada por el cliente.

Éste es el mejor caso para nuestra propuesta, ya que solo hay un perfil de servicio en el repositorio que coincida con la categoría buscada por el cliente. Mientras nuestra propuesta solo hace el emparejamiento con este perfil el otro algoritmo debe comparar con todos los disponibles, por lo que nuestra propuesta obtienes tiempos considerablemente mejores. Obsérvese el gráfico de la figura 7.21.

7.7 CONCLUSIONES

En este capítulo se ha presentado el diseño, implementación y evaluación de un algoritmo como extensión de los actuales algoritmos de emparejamiento de SWS. El principal punto de investigación han sido los diferentes grados de similitud que permitieran emparejamientos flexibles, uso de parámetros no funcionales así como los diversos filtros usados, algunos de los cuales no habían sido considerados en anteriores trabajos de otros autores. La principal contribución al mundo de los SW fue no solo reflejar el aspecto teórico, sino también el aspecto práctico mediante la construcción de un sistema capaz de hacer uso del algoritmo de emparejamiento propuesto.

En este algoritmo de emparejamiento se amplió el rango de grados de similitud de anteriores trabajos, considerando que alguno de los grados definidos por Paolucci et al. eran demasiado generales. Éste era el caso del grado de emparejamiento "*exact*" definido por Paolucci, que incluye los casos en los que el concepto definido por el cliente en su petición es exactamente el mismo o es subclase del concepto que definió el proveedor en su perfil. En el algoritmo propuesto, para una mayor precisión en la búsqueda, se distinguen dos subgrados, uno, el de mayor valor, que será aquél en el que tanto proveedor y cliente utilizan el mismo concepto para definir un parámetro funcional, y un segundo subgrado que será aquél en el que el concepto semántico descrito por el cliente es subclase directa del que define el proveedor. Adicionalmente se ha descrito un nuevo grado de emparejamiento no considerado por Paolucci et al., consistente en la relación de "hermanos" entre dos conceptos dados. Se consideran *hermanos* si ambos son subclase directa de otro concepto y poseen restricciones comunes sobre alguna de sus propiedades.

El algoritmo fue implementado como un componente dentro de un sistema multiagente para publicación y descubrimiento de servicios. Es independiente[38] del lenguaje de especificación de ontologías que se utilice para describir, tanto las descripciones de los SW como las diferentes ontologías utilizadas para calcular el grado de emparejamiento que se tenga. Por el momento, el sistema funciona únicamente con el lenguaje de especificación DAML + OIL para las ontologías de conceptos y DAML-S para la descripción de los SW. Se decidió no utilizar el lenguaje OWL como lenguaje de marcado semántico debido a que en el momento de iniciar la implementación del prototipo este lenguaje aún no era soportado por el razonador y repositorio seleccionados, no existiendo a su vez, ninguna alternativa que soportase el lenguaje. Sin embargo, se han creado descripciones de los servicios tanto en DAML-S 0.9 como en OWL-S 1.0 porque se prevé que próximamente el razonador / repositorio escogido dé

[38] Simplemente bastaría con realizar pequeñas modificaciones de ajuste en los parsers desarrollados.

soporte a OWL, por lo que este sistema podrá migrarse aplicando el mismo algoritmo de emparejamiento desarrollado.

A partir de las pruebas efectuadas se pudo observar como hecho más importante, que utilizando el algoritmo fue posible encontrar servicios basados en similitudes de tipo hermano, las cuales no habían sido estudiadas anteriormente. Aunque este tipo de similitud pudo no haberse tenido en cuenta debido a que en el caso de encontrarse similitudes entre nodos hermanos, posiblemente sería recomendable desde el punto de vista de diseño crear un nodo padre intermedio que agrupase las características comunes de los hijos, nuestro algoritmo las tiene en cuenta partiendo de la premisa de que es posible que este tipo de similitudes no hayan sido consideradas por el proveedor que definió semánticamente estas relaciones, bien sea por consideraciones de rendimiento (para evitar un grado de profundidad demasiado exhaustivo) o simplemente por no haber detectado este tipo de relaciones en la fase de diseño, con lo que el descubrimiento de servicios basado en algoritmos convencionales dejaría de producir resultados que están incluidos en el rango requerido de respuesta.
Por otra parte, la elección de ontologías en las que no hay conceptos con definiciones completas (condiciones necesarias y suficientes para ser miembro de una clase) puede dar lugar a este tipo de situaciones, ya que los razonadores serán incapaces de establecer clasificaciones de manera automática. En estos casos, el peso de la construcción de una buena y apropiada ontología recaerá en el desarrollador de ésta. Por el contrario, el diseño de ontologías con conceptos definidos permitirá solventar este tipo de problema aunque incrementará notablemente el coste computacional, por lo que la decisión de incorporar este grado de similitud se ve ampliamente respaldada.
Otras decisiones como el establecimiento de filtrados anteriores al algoritmo así como el uso de otros parámetros no funcionales, mejoraron considerablemente el coste y la elección del servicio más idóneo respectivamente.
Es importante destacar ciertas mejoras introducidas en el algoritmo que también supusieron un ahorro en el coste computacional como la finalización del proceso comparativo, con un determinado anuncio, cuando no hay emparejamientos de parámetros de salida, y la posibilidad de hacer uso del resto de parejas (petición-anuncio) almacenadas tras el resultado de la búsqueda, para establecer un sistema tolerante a fallos.

Como conclusión final tras las pruebas realizadas, podemos afirmar que el algoritmo propuesto, al contener muchas más consultas (comparaciones) debido a la inclusión de los nuevos grados y accesos al repositorio donde están las ontologías y los perfiles, tiene un mayor coste temporal, el cual es únicamente importante en casos extremos en los que exista un número muy elevado de anuncios de proveedores que pertenezcan a la misma categoría de servicio que el de la petición del cliente. Sin embargo, en situaciones próximas a la realidad, donde aproximadamente el 20 % de los anuncios pertenecerán a la misma categoría de servicio, el coste temporal es similar al del algoritmo de Paolucci et al. e incluso lo mejora en escenarios donde el número de anuncios pertenecientes a la categoría del de la petición es muy bajo. Por ello, podemos concluir que se ha conseguido aumentar la precisión en las búsquedas sin perjudicar el coste temporal en situaciones consideradas normales.

CAPÍTULO 8

MARCO DE TRABAJO PARA LA CREACIÓN DE UN SERVICIO WEB SEMÁNTICO DE TRÁFICO A PARTIR DE UN SITIO WEB CONVENCIONAL

8.1 RESUMEN

En este capítulo, se propone un marco de trabajo para la conversión de portales web convencionales de información de tráfico en SWS.

La información existente en los portales web es almacenada con la adición de significado, consiguiendo en algunos casos potenciar en los servicios creados nuevas capacidades que en un principio no existían.

Para describir la propuesta se resume el proceso planteado mediante la transformación del sitio de información sobre el estado de la red de carreteras de España, perteneciente al portal de información sobre tráfico vial de la DGT. A pesar de no ser un servicio web como tal, se creará éste a partir de un análisis previo del portal web, para su integración en el sistema y de esta manera poder proporcionar este servicio.

El uso de ontologías descritas en el capítulo 6 de la presente memoria, y de ontologías de alto nivel desarrolladas para tal fin en W3C, nos permite describir semánticamente los servicios. A su vez, se proponen diferentes mejoras en las ontologías de descripción, como la adición de nuevos parámetros, una escala Ad Hoc de QoS para este tipo de servicios, así como el desarrollo de una ontología de categorías de SW de tráfico, permitiendo establecer una jerarquía de servicios atendiendo a su tipo.

8.2 POTENCIACIÓN DE NUEVAS CAPACIDADES

En el estudio de los diferentes portales, podemos encontrarnos con dos situaciones diferentes: sitios donde el cliente se ve obligado a interactuar a través de un formulario, y por otra parte, sitios web donde el cliente o usuario de la aplicación recibe la información sin más. A juicio del autor, los primeros son mucho más elaborados y eficaces, ya que en ellos existe la posibilidad de limitar según los requerimientos del usuario, la información mostrada. Por el contrario, en la mayoría de los casos, en los segundos suele aparecer información sobrante sin ningún interés para el solicitante. Lo relevante de esta situación sea el hecho de que en estos casos, se pueden potenciar nuevas capacidades en los servicios mediante el uso de clases y propiedades especificadas en la ontología de conceptos. De esta manera, se pueden obtener nuevas funcionalidades que en un principio no habían sido definidas.

8.3 PROPUESTAS

A continuación se mostrarán una serie de propuestas que permiten crear y manejar SWS de información de tráfico vial.

8.3.1 Inclusión de nuevos parámetros en la ontología de descripción de servicios: Servicios de valor añadido

Desde la coalición de DAML-S indican que cualquier proveedor puede añadir características propias del servicio aunque la clase *Profile* no las haya definido. Para ello, el proveedor debe definir las nuevas propiedades antes de emplearlas como subclases de *ServiceParameter*. Esta clase es la clase raíz proporcionada para que cuelguen de ella los nuevos parámetros de servicio que se quieran añadir, ya que ésta es el rango que posee la propiedad *serviceParameter* definida para la clase *Profile*.
La clase *ServiceParameter* tiene una propiedad llamada *sParameter* que apunta a la clase que define el rango de valores de la nueva propiedad. Para poder definirla, solo tenemos que crear una clase que sea subclase de *ServiceParameter* y restringir la propiedad *sParameter* al tipo de la propiedad.

Para los *Profiles* descritos en esta tesis, se crearon dos nuevas propiedades relacionadas con un atributo nuevo consistente en tener un enlace a un servicio compatible con el que se provee o que el proveedor cree que le puede ser de utilidad al cliente. Por ejemplo, si tenemos un servicio de cálculo de rutas, puede ser que el mismo proveedor esté interesado en tener un enlace a un servicio de información de incidencias de tráfico, el cual podría considerarse útil para el cliente que solicitó el primero de los servicios. Por tanto, es un servicio añadido al que el proveedor anuncia. La idea de facilitar un servicio añadido surge de las investigaciones con SW de Dogac et al. [Dog02].

Para la definición de esta propiedad, se ha creado la clase *AddOn_To* (Añadido_a) como subclase de *ServiceParameter*, y se ha restringido la propiedad *sParameter* a la clase *Profile*, debido a que esta propiedad apunta a *Profiles* de otros servicios. El código está embebido en la ontología que define la clase *Profile*.

```
<daml:Class rdf:ID="AddOn_To">
    <rdfs:subClassOf rdf:resource="#ServiceParameter"/>
    <rdfs:subClassOf>
      <daml:Restriction>
       <daml:onProperty rdf:resource="#sParameter"/>
       <daml:toClass rdf:resource="#Profile"/>
      </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>
```

Una vez definida esta clase, es conveniente definir otro parámetro que sirve para establecer la relación inversa a la anterior, es decir, poder especificar si un servicio está

como valor añadido de otro servicio. Para ello, se ha creado una nueva clase, llamada *Added_Value* (Valor_añadido_a), siguiendo el mismo procedimiento que el realizado para la clase anterior.

```
<daml:Class rdf:ID="Added_Value">
    <rdfs:subClassOf rdf:resource="#ServiceParameter"/>
    <rdfs:subClassOf>
      <daml:Restriction>
       <daml:onProperty rdf:resource="#sParameter"/>
       <daml:toClass rdf:resource="#Profile"/>
      </daml:Restriction>
    </rdfs:subClassOf>
</daml:Class>
```

8.3.2 Propuesta de una ontología de categorización de servicios de tráfico. Jerarquía de perfiles.

El *Profile* es utilizado por los proveedores para anunciar sus servicios, y así permitir realizar de forma automática su descubrimiento por parte de los clientes. Estos *Profiles* son almacenados en diferentes tipos de registros o repositorios, y se le pueden aplicar diferentes técnicas de emparejamiento para la selección de un servicio, ya que desde la coalición no se indica que técnica aplicar, sino que se da libertad para la construcción del sistema que se desee según el contexto donde se encuentre.

Para ayudar a caracterizar un servicio dentro de algún dominio o en el mundo, facilitando un poco más el descubrimiento de un servicio, podemos posicionarlo en una jerarquía de clases de perfiles construida previamente. A partir de la clase *Profile* definido por DAML-S se pueden ir añadiendo subclases que vayan desglosando los diferentes dominios de servicios, diferenciándolos mediante la adición de propiedades características propias para distinguirlas del resto. Esta técnica de construcción de una jerarquía de subclases a partir de la clase *Profile*, es similar a una categorización de servicios o páginas amarillas, pero definido de una manera más formal y con el valor semántico añadido.

En la definición del *profile hierarchy* (o jerarquía de perfiles), suponemos que el nodo raíz de la jerarquía es la clase *Profile,* definida por DAML-S, con dos subclases que serían un primer nivel que diferencia entre ***servicios de información*** (con propiedades propias como el tipo de información que suministra) y ***servicios de e-commerce*** (con propiedades como el tipo de producto que venden etc.), y dentro de esta segunda categoría otro subnivel que diferencia entre los comercios que venden libros, discos de música y un tercero de venta de vehículos. Si tenemos un proveedor de venta de vehículos que quiere hacer un perfil sobre uno de sus servicios, éste será una instancia de la clase *Profile* de venta de vehículos dentro de la jerarquía, y así de esta forma se tendrá clasificado el servicio dentro de ella, por lo que se facilita su descubrimiento cuando alguien está buscando un servicio de este tipo.

Para nuestra aplicación se desarrolló una taxonomía de servicios de tráfico basada en ISO TC 204 [ISO04], la cual permite clasificar estos servicios bajo una categorización. Esta norma ISO se encuentra en fase de desarrollo a partir de diferentes clasificaciones ya realizadas por otras organizaciones o países como Estados Unidos, Europa, Japón, Finlandia, Taiwán, Canadá etc., como ya se comentó en el capítulo 3. Para construir esta taxonomía se construyó una clase raíz de estos servicios que se llamó "*TrafficServices*" y que es subclase de la clase Profile.

```
<daml:Class rdf:ID="TrafficServices">
       <rdfs:subClassOf rdf:resource="&profile;#Profile"/>
</daml:Class>
```

A partir de esta clase, el primer nivel de la jerarquía de los servicios que se indica desde la ISO serán hijos de ésta, y así sucesivamente con el resto de niveles. Por ejemplo, una categoría dentro del primer nivel de la jerarquía ISO la forman todos aquellos servicios que aportan información de viaje:

```
<daml:Class rdf:ID="Traveller_Information">
       <daml:subClassOf rdf:resource="#TrafficServices"/>
       <daml:comment>
       Class that represent all the TravellerInformation Services
       </daml:comment>
</daml:Class>
```

Y dentro de esta categoría, un segundo nivel sería aquellos servicios que aportan información sobre un viaje antes de iniciarse (cálculo de itinerario, servicio de incidencias etc.)

```
<daml:Class rdf:ID="Pre_trip_Information">
       <daml:subClassOf rdf:resource="#Traveller_Information"/>
       <daml:comment>
       Class that represent all the pre trip Services
       </daml:comment>
</daml:Class>
```

Se seguiría así con toda la clasificación de la ISO TC 204.

Esta clasificación, además de permitir catalogar los perfiles para que los proveedores de los servicios utilicen estas clases como plantillas para definir su *profile*, podemos usarla también como una categorización de servicios de tráfico para determinar a qué categoría de servicio de tráfico pertenece nuestro servicio. Para poder usarlo de esta manera, es decir, como un parámetro de la clase *Profile*, es necesario definir la clase *TrafficServices* como subclase de *ServiceCategory* (clase definida por DAML-S para este objetivo), y de esta manera será posible definir en todas nuestras clases de la taxonomía construida a partir de la ISO, las cuatro propiedades heredadas de la clase padre: *categoryName*, *taxonomy*, *value*, *code*. Estas propiedades se podrán emplear para diferenciar cada uno de los perfiles.

La primera de las propiedades, *categoryName*, fue definida por defecto para todas las clases al definirlas en el nodo raíz, dándole como valor" Servicios de Trafico ISO TC204":

```
<daml:Class rdf:ID="TrafficServices">
    <rdfs:subClassOf rdf:resource="&profile;#Profile" />
    <rdfs:subClassOf rdf:resource="&profile;#ServiceCategory" />
    <rdfs:subClassOf>
       <daml:Restriction>
          <daml:onProperty rdf:resource="&profile;#categoryName" />
          <daml:hasValue>
              Servicios de Trafico ISO TC204
         </daml:hasValue>

</daml:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
       <daml:Restriction>
              <daml:onProperty rdf:resource="&profile;#taxonomy" />
             <daml:hasValue>
                http://localhost/DAML_files/Concepts.daml
              </daml:hasValue>
       </daml:Restriction>
     </rdfs:subClassOf>
    <rdfs:subClassOf>
       <daml:Restriction>
              <daml:onProperty rdf:resource="#quality_Level" />
              <daml:hasValue>
   http://localhost/DAML_files/Concepts.daml#GradoCalidadServicioWebTrafico
              </daml:hasValue>
       </daml:Restriction>
    </rdfs:subClassOf>
  </daml:Class>
```

La propiedad *taxonomy* sirve para indicar cual es la URL de la ontología de conceptos asociada por defecto a esta categoría de servicios. En este caso, la clase raíz *TrafficServices* apunta a una ontología de conceptos más general, pero conforme se desciende en la taxonomía, los servicios asocian ontologías de conceptos más concretas al objetivo que persiguen.

Para esta clase, *TrafficServices*, se define una nueva propiedad denominada *qualityLevel* que apunta a la URI del concepto de calidad de servicio especificada para esta categoría de servicio (tal y como veremos en la siguiente sección). De esta manera, cuando se selecciona la categoría de servicio, la interfaz cliente conoce que concepto marca la calidad de servicio para esta categoría y permitir así al cliente definir la calidad que espera del servicio en sus búsquedas. Cada una de las clases que definen cada categoría de servicio en la taxonomía, define la restricción de valor a esta propiedad para cada concepto que defina la calidad de esa categoría.

Para poder hacer uso de la taxonomía de categorías de servicios de tráfico, en la subontología de servicios *Profile*, se deben añadir las siguientes líneas de código para que sea tenida en cuenta como un parámetro adicional para representar una categoría, ya que ha sido declarada como subclase de *ServiceCategory* que es el rango que posee la propiedad *serviceCategory* de la clase *Profile* definida para tal objetivo:

```
<daml:Class rdf:about="&micategoria;#TrafficServices">
    <daml:comment>
     Una categoria de servicios propia, cuando se haga uso se pondrá de la forma
correspondiente

       xmlns:micategoria = “http://localhost/DAML_files/CategoriaServicio.daml”
    </daml:comment>
</daml:Class>
```

De esta forma, se pueden definir perfiles de servicios en los que uno de sus parámetros sea su categoría de servicio dentro de tráfico:

```
<profileHierarchy:Pre_Trip_Information
        rdf:ID="Profile_Incidencias_Trafico_Service">

……….
   <profile:serviceCategory>
     <profile:TrafficServices rdf:about=
"http://localhost/DAML_files/CategoriaServicios.daml#Pre_trip_Information"/>
   </profile:serviceCategory>
……….
</profileHierarchy:Pre_Trip_Information>
```

Los parámetros *value* y *code* son propios de cada tipo de servicio o nodo dentro de la taxonomía, por lo que su valor se les asigna ya en la ontología *CategoriaServicios* al definir cada una de estas clases, siendo éste único y por lo tanto identificable unívocamente para esta categoría de servicio. Así, ya se dispone de dos propiedades más que diferencian cada *Profile* del resto, tal y como describimos al definir el significado de *profile hierarchy*. En el ejemplo de *Pre_trip_Information*, la clase quedaría definida de la siguiente manera en la ontología *CategoriaServicios*:

```
<daml:Class rdf:ID="Pre_trip_Information">
    <daml:subClassOf rdf:resource="#Traveller_Information"/>

    <daml:comment>
    Class that represent all the pre trip services
    </daml:comment>

    <rdfs:subClassOf>
       <daml:Restriction>
           <daml:onProperty rdf:resource="&profile;#value" />
           <daml:hasValue> pre trip information </daml:hasValue>
       </daml:Restriction>
     </rdfs:subClassOf>

    <rdfs:subClassOf>
       <daml:Restriction>
           <daml:onProperty rdf:resource="&profile;#code" />
           <daml:hasValue> 101 </daml:hasValue>
       </daml:Restriction>
     </rdfs:subClassOf>
</daml:Class>
```

De esta manera, cuando un proveedor seleccione la categoría de servicio a la que pertenece su servicio, se le asignará su *value* y su *code*. Así, cuando el cliente seleccione la categoría de servicio al construir su petición, también incluirá estos parámetros, y se podrá ya seleccionar de nuestro repositorio de servicios, aquéllos que tienen el mismo *value* y *code*, facilitándonos así el proceso de emparejamiento. Este filtro de categoría de servicio será el primero que se emplee en nuestro sistema prototipo, ya que nos permitirá seleccionar aquellos proveedores que pertenecen a la misma categoría de servicio que el cliente incluye en su descripción *Profile* de petición, y descartar así, muchos servicios que no satisfacen la petición del usuario. Al emplear este filtro liberamos de mucha carga de trabajo al resto de procesos, ahorrando mucho tiempo de procesamiento.

Un diagrama que representa parte del conocimiento descrito en esta ontología puede ser visto en la figura 8.1.

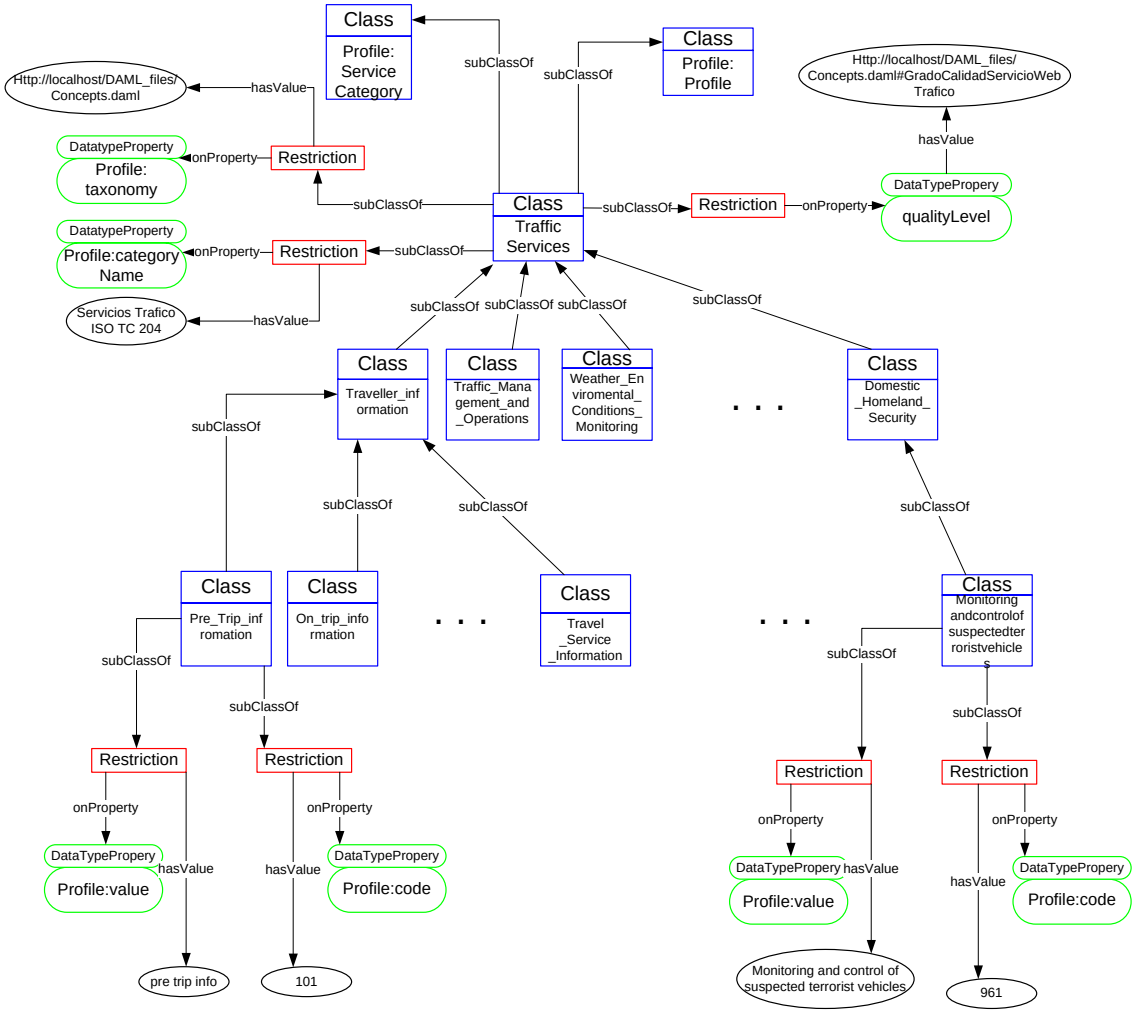


Figura 8.1 Parte de la ontología de categoría de servicios o Jerarquía de perfiles

Desde la coalición también se advierte de que estas jerarquías de perfiles de servicios, de productos o de regiones geográficas deberían ser definidas desde fuera de ella, y que deberían ser las instituciones las que se pusiesen de acuerdo para definirlas, ya que aunque tienen relación y son usadas por ellos, aconsejan que deben ser otros los que las definan en el lenguaje de marcado semántico correspondiente. Por este motivo, se utilizan las taxonomías ya definidas NAICS y UNSPSC para definir la jerarquía de *Profiles*, mientras que en la presente investigación, se hizo uso de la ISO TC 204 para construir la taxonomía específica de servicios de tráfico. También indican desde la coalición que no es obligatorio que todo perfil de servicio sea una instancia de un perfil clasificado en la jerarquía, ya que puede ser que el servicio no se pueda clasificar en un dominio concreto. En este caso, se decide instanciar directamente desde el nodo raíz (clase *Profile*) de la jerarquía.

8.3.3 Propuesta de una métrica para evaluar la calidad de servicio en servicios web de información de tráfico.

ISO define calidad como “La totalidad de características de un producto o servicio que se refieren a su habilidad para satisfacer necesidades indicadas” (ISO8402).

Existen numerosos enfoques y propuestas que pueden servir de referente para medir la calidad de servicio web como la política de seguridad, capacidad de soportar diferentes protocolos web, infraestructuras y alternativas ante fallos del servicio etc. [Man02].

En [Gu02a] se propone un modelo de agregación de calidad de servicio escalable que está compuesto por dos capas principales: composición de servicio bajo demanda y selección dinámica de punto (peer) donde los servicios son realmente instanciados, presentando a su vez un algoritmo de composición de servicio bajo demanda que permite satisfacer los requerimientos de calidad de un usuario. En [Gu02b] se describe un lenguaje de marcado basado en XML llamado HQML así como un entorno de programación visual denominado QoSTalk, los cuales han sido probados en el desarrollo de aplicaciones multimedia distribuidas. En [Gu05] se ha propuesto un framework de composición de servicio descentralizado llamado *SpiderNet* basados en requerimientos dinámicos de composición de servicio de un usuario.

En [Ali02] se propone un framework basado en QoS para administración en servicios orientados a Grid, donde se proveen extensiones a UDDI para incorporar definiciones de QoS en documentos basados en WSDL. En [Max04] se presenta un framework basado en agentes denominado *Web Services Agent Framework* (WSAF) y una ontología de calidad de servicio que incorpora varios aspectos de calidad encontrados en sistemas distribuidos (disponibilidad, capacidad, aspectos económicos, interoperabilidad etc.).

En la especificación de DAML-S/OWL-S se incluyen constructores para expresar parámetros relativos a la calidad de servicio (*QoS*), sin embargo, no proveen de un conjunto detallado de clases y propiedades que permitan representar esta métrica.

Cardoso et al. proponen en [Car02b] un modelo compuesto por las dimensiones de tiempo, coste, confiabilidad y fidelidad. Basándose en dicho modelo, desarrollaron una ontología para la especificación de la métrica. Otro trabajo relacionado es DAML-QoS Ontology de Chen at al. [Che04] en el que proponen el uso de una ontología de QoS como complemento a DAML-S. En este último trabajo, establecen tres niveles de capas en su ontología, entre las que destaca la QoS profile utilizada en las descripciones de servicios para anunciar y realizar peticiones haciendo uso de la calidad del servicio.

Desde el punto de vista de la dificultad de aplicar dichos modelos a los SW de tráfico, se ha creado en esta investigación, un modelo diferente que tuviera en cuenta las características propias de los portales de información de tráfico objeto de estudio.

La *QoS* completa parte del vacío existente en las técnicas de emparejamiento de servicios. La *QoS* viene definida por parámetros no-funcionales del servicio como son *performance* (representación, funcionamiento), *reliability* (confiabilidad), *availability* (disponibilidad) y *security* (seguridad). Estos parámetros no se pueden expresar en ninguno de los lenguajes de descripción de los SW como es WSDL y OWL-S. Aunque desde OWL-S, en su clase *Profile*, se ha descrito una propiedad (porque se permite añadir parámetros no funcionales) que representa el parámetro no-funcional indicador de la calidad del servicio.

Al ser un parámetro definido para los *Profile* de los SW, debe servir como otro parámetro más que facilite el emparejamiento de éstos. Siguiendo la recomendación de la coalición, la calidad de un servicio web no se medirá esta vez como disponibilidad, seguridad u otros parámetros que se sobreentienden que son los que deberían proporcionar el nivel de calidad de servicio, sino que a cada categoría de servicios se le asociará una escala específica.

Por otra parte, la QoS puede influir en la popularidad de un servicio, cuanto más tenga, más probabilidad tiene de ser escogido ante otros servicios con peor calidad. Por ejemplo, en la calidad SW de hoteles, el número de estrellas que el hotel posea puede resultar un aspecto fundamental, o en el caso de los restaurantes el número de tenedores que dispongan.

Como el tema de estudio en esta tesis han sido los SW de tráfico, inicialmente no se disponía de ninguna escala de calidad. Al describir un servicio Web semánticamente, y en especial cuando se completa el *Profile* del servicio, se indica el grado de QoS correspondiente.

Para la obtención de dicho grado, se ha propuesto una **métrica basada en factores de calidad**. Esta métrica tiene como variables que determinarán la calidad del servicio web los siguientes factores:

- Coste: si el servicio web es gratuito o no. Si es gratuito, el valor de esta variable será el máximo posible.
- Actualización: si el servicio web actualiza con bastante frecuencia su información o servicio, se obtendrán valores mayores en esta variable.

- Administraciones públicas: si el servicio web es proporcionado por las administraciones públicas tiene más prioridad que aquél que ha sido proporcionado por una entidad privada.
- Facilidad de manejo: si el cliente casi no tiene que intervenir en la ejecución de servicio web para obtener resultados, la calidad de este servicio es mayor que la de otro que necesita de mayor participación del cliente para que éste obtenga los mismos resultados.
- Confiabilidad e Integridad: el servicio debe dar garantías de que se mantendrá la corrección durante la ejecución del servicio web.
- Seguridad: si al servicio web de tráfico es necesario proporcionarle parámetros con información personal como el número de una tarjeta de crédito, de teléfono etc., para lo cual será necesario tener en cuenta las medidas adoptadas referentes a seguridad, como encriptación y firma digital.

Atendiendo a esta métrica se calcula el valor final correspondiente. Para su obtención se otorga a cada uno de los factores un valor entre 0 y 10 a excepción del factor de *Administraciones públicas* que acumula un valor de 4 puntos. Se ha premiado con esta decisión a aquellos servicios provistos por estas entidades, frente a los privados, debido principalmente al hecho de que como regla generalizada (en información de tráfico) los últimos suelen obtener la información de los primeros.

La evaluación de la calidad de un servicio, y más concretamente la concesión de un determinado valor a cada factor, deberá ser realizada por entidades ajenas al servicio, ejerciendo labores de auditoria.

Estas propuestas podrían incluir en un futuro otros factores de interés o incluso una ponderación relativa a cada factor considerado, según su importancia. Sin embargo, se ha optado por este tipo de métrica, en una fase inicial, por su sencillez.

$$\text{Calidad} = [\text{Coste}(0\text{-}10) + \text{Actualización}\ (0\text{-}10) + \text{Manejo}(0\text{-}10) + \text{Ad. Públicas}\ (4) + \text{Seguridad}\ (0\text{-}10) + \text{Confiabilidad-.Integridad}\ (0\text{-}10)] \ / \ 5.4$$

Finalmente, el valor de calidad obtenido se compara con la tabla 8.1 para indicar el grado de calidad que ofrece el servicio:

Valor de Calidad	Grado de Calidad
]8,10]	Excelente
]6,8]	Muy Buena
]4,6]	Buena
]2 ,4]	Regular
[0,2]	Mala

Tabla 8.1: Rango Calidad de servicio de tráfico

Los grados de calidad son instancias de la clase *GradoCalidadServiciosWebTrafico,* los cuales, tanto el concepto como las instancias, están definidos en la ontología de conceptos o *Concepts* (siguiente sección).

Esta escala de calidad ha sido pensada para su uso en la descripción de SW de tráfico, sin embargo, podría ser aplicada en otras áreas.

Una vez el proveedor conoce la calidad de su servicio web de tráfico, solo debe ser referenciada en la propiedad de *Qualityrating* en la instancia que define el *Profile* del servicio web:

```
<profile:qualityRating>
 <profile:QualityRating rdf:ID="#Trafico-Rating">
   <profile:ratingName>Grado Calidad</profile:ratingName>
   <profile:rating rdf:resource=
       "http://localhost/DAML_files/Concepts.daml#Excelente"/>
 </profile:QualityRating>
</profile:qualityRating>
```

Para que se puedan seleccionar los recursos que marcan los diferentes grados de nuestra calidad de servicio, se modifica la subontología *Profile* para que la propiedad *rating* apunte a instancias del concepto *GradoCalidadServiciosWebTrafico.*

Nuestro enfoque ha sido utilizado para evaluar la calidad de servicio en diversos prototipos que hemos desarrollado, y se ha orientado en informar al usuario teniendo en cuenta el elemento de QoS a partir de la métrica establecida. Este estudio trata de especificar este tipo de factores en los SW de información de tráfico vial, para que pueda ser utilizado en el descubrimiento de servicios de información, mediante el emparejamiento entre la calidad de servicio requerida por un usuario y la búsqueda que más se adapte a ella.

8.3.4 Integración con la ontología de conceptos de tráfico

La ontología de conceptos desarrollada podrá ser empleada dentro de un sistema multiagente, con tres finalidades principalmente:

- Especificar la base de conocimiento periódicamente instanciada y que contendrá todo el conocimiento extraído de la web.

- Confección de los perfiles de servicios requeridos por el cliente a través de un interfaz gráfica (ampliamente detallado en el capítulo 9), para que pueda ser visualizada y sea posible seleccionar las capacidades que se desea que tenga el servicio, así como por los proveedores para poder indicar cuales son las capacidades que poseen los SW que representan y,

- la utilización en el emparejamiento de estos servicios mediante la aplicación del algoritmo propuesto, el cual ha sido descrito en el capítulo 7.

En un sistema de emparejamiento de SW, se deben de tener ontologías de conceptos para cada una de las categorías de servicios. La ontología de conceptos de una categoría consiste en la definición de los términos y relaciones referentes al dominio de esa categoría.

En todas las ontologías de conceptos debe quedar expresado todo lo referente al dominio o categoría de servicio. Tanto términos como propiedades y relaciones deben aparecer de forma coherente y representan al máximo el conocimiento del dominio de conceptos, ya que la utilización de esta ontología es muy importante. Cuanto más conceptos y relaciones queden expresados en la ontología, más facilidades se les dan a los proveedores y clientes para describir los parámetros de sus perfiles.

Los proveedores y clientes de una misma categoría de servicio se basan en la misma ontología de conceptos para construir los respectivos perfiles, por lo que el motor del emparejador, el razonador, emplea esta ontología para hacer el emparejamiento semántico entre los parámetros del servicio porque ambos emplean las mismas clases y conceptos semánticos para describirlos.

En esta investigación al tratar con SW orientados a tráfico vial, se construyó una ontología de conceptos formada por un abanico amplio de términos y relaciones referidos a este dominio. Esta ontología permite definir y otorgar el valor semántico a los parámetros funcionales en la construcción de la descripción semántica de los servicios, tanto en el *Profile* como en el *Process*.

8.4 CREACIÓN DE UN SERVICIO WEB SEMÁNTICO

A partir de la búsqueda realizada no se encontraron SW de tráfico ni descripciones de los mismos. Sin embargo, páginas o portales web que ofrecen información sobre tráfico si que se han encontrado, y éstos serán descritos semánticamente empleando la ontología adecuada para tal fin, como si fuesen SW de tráfico reales.

Generalmente lo que ofrecen las administraciones de tráfico en sus portales web no son SW en sí, sino sitios web que en algunos casos incluyen formularios para poder interactuar con una base de datos y que dependiendo de los valores de entrada devolverán una información u otra. Estos portales no están capacitados para recibir peticiones de invocación mediante el protocolo SOAP, por lo que será necesario realizar un proceso de adaptación. También se necesita de un agente extractor (*Wrapper*) para poder tratar semánticamente la información que estos portales devuelven.

Para la creación de SWS se distinguen tres fases:

- Almacenamiento con contenido semántico de la información difundida por la web.
- Creación y despliegue de SW tradicionales que hagan uso de la información semántica almacenada.
- Descripción semántica de las capacidades de estos servicios.

8.4.1 Almacenamiento con contenido semántico: Instanciación.

No solamente deberemos de realizar correctamente el despliegue de los servicios y su posterior descripción semántica, sino que también se deberá convertir toda la información aportada por la web a contenido semántico según las ontologías creadas para ello. Este proceso puede ser realizado mediante *scripts* y dado que se debe realizar a intervalos regulares, para mantener la información actualizada, el uso de agentes de tipo *wrapper* es esencial para llevar a cabo nuestro objetivo. Para el desarrollo de los *scripts* extractores de contenido a partir de HTML se ha utilizado WebL [Mar99].

Los agentes de este tipo tienen como única tarea la extracción y la conversión a lenguaje formal semántico de información de web, y ésta la realizan periódicamente mediante el uso de temporizadores. Una vez el agente *wrapper* conoce las clases y propiedades de la ontología (*Concepts*), solo tiene que acceder a la página web, leer la información y generar las instancias de la clase que sirven para describir semánticamente la información.

A continuación, el *wrapper* debe almacenar las instancias generadas en algún repositorio determinado para ser mantenidas y consultadas. La inserción en el repositorio de los datos se realiza mediante los métodos de inserción desarrollados para el emparejador.

Uno de los servicios de tráfico que se han implementado es el servicio de información de incidencias de tráfico de la DGT. Este servicio, pertenece al tipo de servicios que poseen un formulario previo (figura 8.2) para que el cliente mediante la introducción de valores de entrada, restrinja la información de incidencias que quiere obtener del servicio.

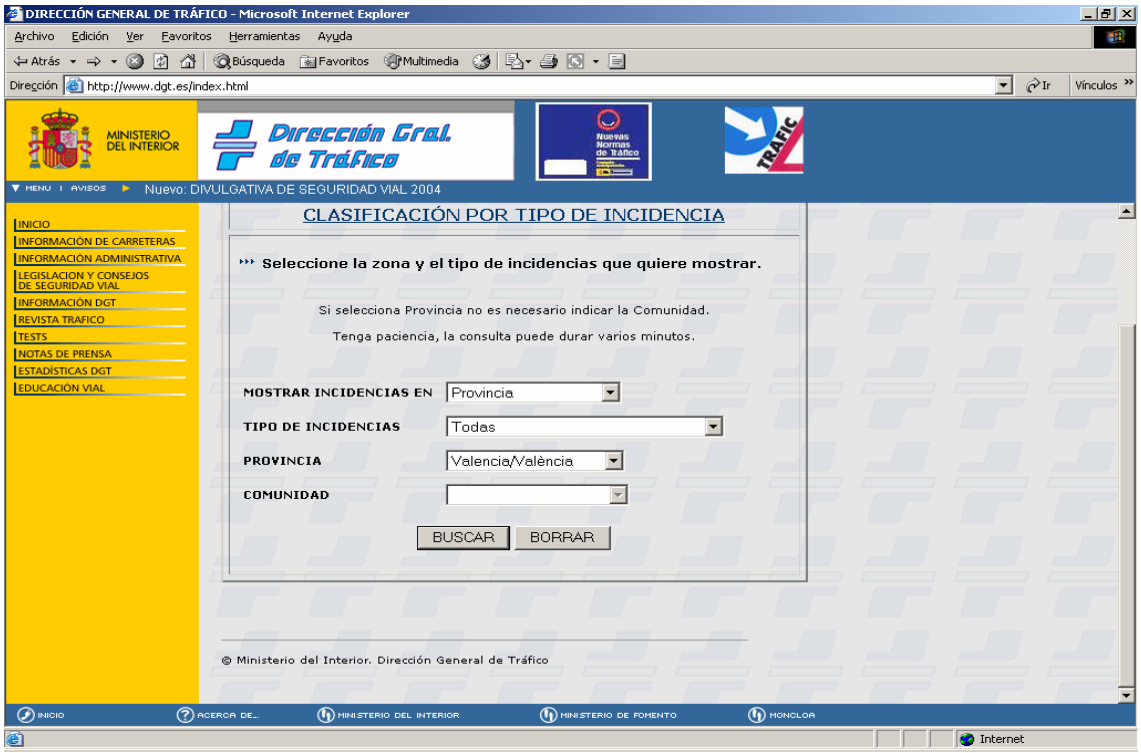


Figura 8.2 Portal web de información de incidencias de la DGT

La salida de este servicio es una página HTML que contiene una tabla con todas las incidencias de tráfico existentes en ese momento, contextualizada a los requerimientos del usuario (Ver figura 8.3). A partir de un análisis de este resultado, se debe adecuar la ontología para que sea capaz de describir semánticamente este tipo de información sin sufrir perdida de expresividad o funcionalidad.

CRITERIO: Toda España
TIPO DE INCIDENCIAS: Todas
Jueves, 1 de Julio del 2004 12:08:05

Tipo de Incidencia	Causas y observaciones	Provincia	Población	Fecha-Hora Inicial	Nivel	Carretera	Km. Inicial	Km. Final	Sentido	Hacia
METEOROLOGICA	VIENTO	CADIZ	CHICLANA DE LA FRONTERA	2004-06-26 15:37	VERDE	N-340	12.5	95.0	Ambos sentidos	Ambos
CONO	CARRIL EN SENTIDO CONTRARIO CONOS	GIRONA	LLAGOSTERA	2004-06-27 22:07	VERDE	C-65	2.0	5.0	Sur	BARCELONA
CONO	CARRIL INCORPORACION CONOS	VALENCIA	MASALFASAR	2004-07-01 09:15	VERDE	V-21	9.0	9.1	Creciente de la kilo	VALENCIA
OBRAS	TRABAJOS EN LA MEDIANA	ALICANTE/ALACANT	ALICANTE	2004-07-01 00:51	VERDE	A-70	5.0	7.0	Creciente de la kilo	MURCIA
OBRAS	OBRAS EN GENERAL	ALMERIA	LOS GALLARDOS	2004-06-30 12:06	VERDE	A-7	520.0	521.0	Creciente de la kilo	BARCELONA
OBRAS	OBRAS EN GENERAL	BARCELONA	MARTORELL	2004-04-19 07:07	VERDE	AP-7	170.0	168.0	Norte	BARCELONA
OBRAS	OBRAS EN GENERAL	BARCELONA	SANTA PERPETUA DE MOGODA	2003-12-23 00:53	VERDE	A-7	143.0	146.0	Norte	GIRONA
OBRAS	SEÑALIZACION DE LA CALZADA	ALBACETE	YESTE (MU)	2004-04-09 12:07	VERDE	CV-A-63	5.0	5.0	Creciente de la kilo	
OBRAS	TRABAJOS EN LA MEDIANA	ALICANTE/ALACANT	ALICANTE	2004-07-01 00:51	VERDE	A-70	5.0	7.0	Creciente de la kilo	MURCIA
OBRAS	MANTENIMIENTO DE PUENTES	CADIZ	BARBATE DE FRANCO	2004-05-26 12:12	VERDE	N-340	46.7	47.6	Ambos sentidos	CADIZ - BARCELONA
OBRAS	GRANDES OBRAS	CADIZ	VEJER DE LA FRONTERA	2004-06-07 19:44	VERDE	N-340	27.7	28.7	Decreciente de la ki	A
OBRAS	TRABAJOS DE MANTENIMIENTO	GIRONA	VILADASENS	2004-06-04 21:58	VERDE	AP-7	44.0	42.0	Norte	DESVIAMENT PROVISIONAL
OBRAS	TRABAJOS DE DEMOLICION	BARCELONA	CASTELLBISBAL	2004-05-06 10:57	NEGRO	B-225	0.0	1.0	Ambos sentidos	

Figura 8.3 Resultado tras completar el formulario sobre incidencias de la DGT

Los detalles del desarrollo de esta ontología así como de la instanciación, los cuales constituyen la base de conocimiento ya fueron expuestos en el capítulo 6. Esta base de conocimiento residirá en algún repositorio, el cual será actualizado periódicamente.

Por tanto, tras el almacenamiento de la información en el repositorio, un agente proveedor ya puede acceder a ella mediante cualquiera de los lenguajes de consulta permitidos por éste. Las consultas elaboradas formarán parte del código generado del nuevo servicio web creado, tal y como describiremos en el siguiente punto.

8.4.2 Construcción y despliegue de un servicio web.

Un servicio web es un componente al que se podrá acceder mediante protocolos web estándar:

- Los mensajes para invocar el servicio se codifican en XML y,
- estos mensajes se pueden transportar utilizando HTTP

Normalmente consta de una interfaz (conjunto de métodos) cuya invocación se puede realizar de forma remota desde cualquier lugar de la red, lo que permite crear aplicaciones distribuidas en Internet. Deben ser "interoperables", y para alcanzar esta característica se basan en el uso de protocolos estándar como SOAP. Por tanto, se debe proporcionar una interfaz que admita mensajes SOAP para recibir peticiones y devuelva los resultados, mediante este tipo de mensajes, al cliente que realice la invocación, después de haber finalizado la ejecución del servicio.

Para generar el esqueleto del servicio se pueden utilizar diversas herramientas como Apache Axis [Alm02]. El proceso básico para construir un servicio web con Apache Axis puede ser leido en el documento denominado **"Instanciación y despliegue de servicios**" que se encuentra en la URL http://robotica.uv.es/tesis/javi/document.html y en el CD-ROM que acompaña esta memoria.

Si no se utilizara ningún tipo de herramientas, se podrían cometer errores, no respetar el uso de estándares e incluso perder interoperabilidad. Sin embargo, la funcionalidad de los servicios ha de ser introducida manualmente, mediante el desarrollo de consultas (embebidas en los métodos de la interfaz) que accederán a la base de conocimiento. Gracias al desarrollo de estas consultas se pueden establecer las mismas capacidades ofrecidas por los portales web o por el contrario potenciar nuevas capacidades (referirse a la sección 8.2) como efectuar recuperación de información basada en requerimientos de tipo temporal, en servicios cuyos portales origen no disponían de esta capacidad.

La siguiente consulta devuelve las incidencias de la provincia y del tipo especificados por el cliente, en un servicio de incidencias, tal y como realiza la web:

```
Select Incidencia
From {Incidencia} <rdf:type> {<ns8:Incidente>},
       {Incidencia} <ns8:tieneProvincia> {A},
       {Incidencia} <ns8:tipo> {B}
Where A = "URIdelaprovinciaqueespecificaelcliente"
       and  B = "URIdeltipoincidenciaqueespecificaelcliente"
Using namespace
       rdfs = <!http://www.w3.org/2000/01/rdf-schema#>,
       service=<!http://www.DAML.org/services/DAML-s/0.9/Service.DAML#>,
       DAML = <!http://www.DAML.org/2001/03/DAML+oil#>,
       rdf = <!http://www.w3.org/1999/02/22-rdf-syntax-ns#>,
       ns8 = <!http://localhost/DAML_files/Sucesos.DAML#>,
            bor = <!http://www.ontotext.com/otk/2002/07/bor#>,
       xsd = <!http://www.w3.org/2000/10/XMLSchema#>
```

Mientras que esta otra, potenciará una nueva capacidad de tipo temporal en un servicio de previsiones de tráfico. La consulta devolverá aquellas previsiones cuya fecha sea la definida por el cliente:

```
Select Prevision, descripcion
From {Prevision} <rdf:type> {<ns8:Prevision>},
       {Prevision} <ns8:fechadePrevision> {A},
       {Prevision} <ns8:descripcionPrevision> {descripcion},
       {A} <time:year> {Y},
       {A} <time:month> {M},
       {A} <time:day> {D},
       {Y} <rdf:value> {y},
       {M} <rdf:value> {m},
       {D} <rdf:value> {d}
Where d ="dia"
       and m ="mes"
       and y ="anyo
Using namespace
       rdfs = <!http://www.w3.org/2000/01/rdf-schema#>,
       service=<!http://www.DAML.org/services/DAML-s/0.9/Service.DAML#>,
              DAML = <!http://www.DAML.org/2001/03/DAML+oil#>,
       rdf = <!http://www.w3.org/1999/02/22-rdf-syntax-ns#>,
       ns8 = <!http://localhost/DAML_files/Sucesos.DAML#>,
       time = <!http://localhost/DAML_files/Time.DAML#>,
       bor = <!http://www.ontotext.com/otk/2002/07/bor#>,
       xsd = <!http://www.w3.org/2000/10/XMLSchema#>
```

Una vez generada la interfaz SOAP del servicio así como su descripción mediante WSDL, se despliega la aplicación en un servidor web determinado, tras lo cual el servicio será accesible mediante su URI específica.

8.4.3 Descripción semántica de las capacidades del servicio.

Como ya vimos en la exposición del estado del arte sobre SW y más concretamente al tratar los SWS, la descripción de éstos mediante ontologías DAML-S (versión 0.9) / OWL-S (versión 1.0) pasa por la especificación de ésta a través de diferentes ficheros que componen la ontología.

8.4.3.1 Service

Al definir la descripción del servicio, el primer fichero a realizar es el *Service*, que servirá para rellenar los datos de una clase *Service* identificando el servicio e indicando en sus propiedades cuales son las instancias de *Profile*, *Process* y *Grounding* que presentan, describen y dan soporte al servicio.

El primer paso para escribir el fichero, será como en cualquier fichero de ontologías, la definición de los *namespaces* que se van a utilizar. Hay una serie de *namespaces* constantes en cualquier especificación, como son las direcciones de *XML Schema*, de la sintaxis de RDF, el modelo de *RDF schema*, el modelo de las ontologías DAML+OIL, al igual que las cuatro subontologías que forman la ontología DAML-S (*service*, *profile*, *process*, *grounding*) porque son éstas las que definen las clases y propiedades que se utilizan. También pueden ser útiles las direcciones de los ficheros donde el proveedor define cada una de las subontologías que describen el servicio web específico. A su vez serán importantes para nuestro fichero las ontologías de conceptos que se puedan emplear, la ontología geográfica y la ontología de jerarquía de perfiles.

El *URI* de la ontología *Profile* contiene la dirección del servidor en el que está alojado el Profile propuesto (MiProfile), el cual define las mismas clases y propiedades que el proporcionado por el comité de DAML-S, pero con la adición de las propiedades y restricciones que permiten emplear las nuevas propiedades añadidas, y que algunas de ellas tengan como rango instancias de conceptos definidos en las ontologías desarrolladas en esta tesis (p.e. la propiedad *GeographicRadius* tiene como rango instancias de las clases definidas en Geografia: *Pais*, *Comunidad_Autonoma*, *Provincia*, *Comarca*, *Poblacion*).

Después de esto, en todos los ficheros DAML-S que describen el servicio añadimos la cabecera típica de cualquier ontología DAML+OIL. Esta cabecera la podemos aprovechar para indicar la fecha y la versión de la ontología. Al final toda esta cabecera tendrá el siguiente aspecto:

```
 <?xml version="1.0" encoding="ISO-8859-1" ?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
xmlns:service="http://www.daml.org/services/daml-s/0.9/Service.daml#"
xmlns:profile= "http://localhost/DAML_files/MiProfile.daml#"
xmlns:process="http://www.daml.org/services/daml-s/0.9/Process.daml#"
xmlns:grounding="http://www.daml.org/services/daml-s/0.9/Grounding.daml#"

xmlns:profileHierarchy="http://localhost/DAML_files/CategoriaServicios.daml#"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema.xsd#"

xmlns:traficoService="http://localhost/DAML_files/TraficoService.daml#"
xmlns:traficoProfile="http://localhost/DAML_files/TraficoProfile.daml#"
xmlns:traficoProcess="http://localhost/DAML_files/TraficoProcess.daml#
xmlns:traficoGrounding="http://localhost/DAML_files/TraficoGrounding.daml#"

 xmlns:geografia="http://localhost/DAML_files/Geografia.daml#"
 xmlns:concepts="http://localhost/DAML_files/Concepts.daml#"
 xmlns:time="http://localhost/DAML_files/Time.daml#">
<daml:Ontology>
  <daml:versionInfo>
        $Id: TraficoProfile.daml,v 1.00 2003/05/15
  </daml:versionInfo>
  <rdfs:comment>
        Ejemplo de incidencias de trafico para DAML-S version 0.9
  </rdfs:comment>
  <daml:imports
        rdf:resource="http://www.daml.org/2001/03/daml+oil.daml" />
  <daml:imports
 rdf:resource="http://www.daml.org/services/daml-s/0.9/Service.daml" />
  <daml:imports rdf:resource=
       "http://localhost/DAML_files/TraficoProfile.daml" />
  <daml:imports rdf:resource=
       "http://localhost/DAML_files/TraficoProcess.daml" />
  <daml:imports rdf:resource=
       "http://localhost/DAML_files/TraficoGrounding.daml"/>
</daml:Ontology>
```

En este momento ya es posible instanciar la clase Service para definir el servicio de incidencias de la DGT, enlazando las propiedades *presents*, *describedBy* y *supports* con las URI's que vamos a dar a las otras tres instancias de las clases principales de la ontología de DAML-S, es decir a *Profile*, *Process* y *Grounding* respectivamente:

```
<service:Service rdf:ID="IncidenciasTraficoService">
 <!--  Reference to the Profile  -->
  <service:presents rdf:resource=
       "http://localhost/DAML_files/TraficoProfile.daml#Profile_Incidencias_
       Trafico_Service" />
 <!--  Reference to the Process Model   -->
  <service:describedBy rdf:resource=
       "http://localhost/DAML_files/TraficoProcess.daml#IncidenciasTrafico
       ProcessModel" />
<!--  Reference to the Grounding   -->
  <service:supports
rdf:resource="http://localhost/DAML_files/TraficoGrounding.daml#IncidenciasTrafico
       ServiceGrounding" />
</service:Service>
</rdf:RDF>
```

8.4.3.2 Profile

El siguiente paso en la descripción del servicio será la construcción de una instancia de la clase *Profile*. En la definición de la instancia, debemos de ser coherentes con lo que el servicio ofrece al cliente, ya que no podemos especificar un perfil mediante el fichero *profile* que no corresponda con lo que el proveedor realiza en su servicio. Este paso es importante, porque la ontología que contiene el *Profile* le servirá al emparejador y a los agentes web para descubrir servicios acordes con la petición de un usuario. Éste es un paso delicado y por tanto, lo que se describa o anuncie en el *Profile* de un proveedor debe ser exactamente lo que realice, no debe, anunciar capacidades que el servicio no disponga, ni quedarse corto cuando defina sus capacidades, porque puede perder competitividad respecto a otros SW.

Cuando se define un *Profile* de un servicio se deben de completar las propiedades que esta clase posee, porque son las que le sirven al servicio para autodefinirse y permitir así su publicidad. Por tanto, se analiza el servicio web para poder obtener la máxima información sobre las capacidades y parámetros, y de este modo, completar con la mejor información el fichero *profile* del servicio.

El portal web que ofrece información sobre incidencias de la DGT presenta un formulario para captar unos parámetros de entrada, los cuales podemos entender como los *Inputs* de la clase *Profile*. El formulario está formado por cuatro listas desplegables que nos sirven para indicar los siguientes parámetros: *Mostrar_Incidencias_en* (para especificar si queremos buscar por provincias o por Comunidad Autónoma), *Tipo_Incidencia* (podemos elegir un filtro de las incidencias para que solo nos muestre aquéllas que son provocadas por el tipo de incidencia que le indiquemos), *Provincia* (una lista desplegable con todas las provincias españolas) y *Comunidad* (otra lista desplegable con el nombre de todas las comunidades autónomas de España). Estos cuatro "*inputs*" deberemos de anotarlos como entradas de nuestro servicio, al igual que indicar el valor semántico que tienen, es decir, deberemos señalar por ejemplo, que el *input* de *Provincia* toma como valor semántico la clase que define a *Provincia* dentro de la ontología de conceptos referidos a términos de geografía, y así con el resto de *inputs*.

El portal web, tras la introducción de los parámetros de entrada, muestra tablas de información sobre incidencias que se deberán tener en cuenta en la especificación de la salida del servicio (*outputs*). Por lo que como en las entradas (*inputs)*, deberemos de definir todas estas salidas y el valor semántico que representa cada una. Para lograr este cometido, hacemos uso de la clase o concepto definido en una ontología de este dominio. En este caso, tenemos como salida el concepto **Incidente**, que deberá poseer propiedades para indicar cada una de las características que describen a una incidencia de tráfico como las aportadas por la DGT.

Pero veamos como construimos el fichero con la instanciación de la clase Profile, en el que paso a paso, las diferentes propiedades que un proveedor describe deberán ser completadas cuando quiera anunciarse mediante la semántica de DAML-S.

Primero se incluyen la cabecera de los *namespace* y de la definición de una ontología DAML+OIL. La siguiente línea ya será la identificación de la instancia de la clase *Profile* de nuestro servicio web.

```
<profileHierarchy:Pre_trip_Information
        rdf:ID="#Profile_Incidencias_Trafico_Service">
```

En este caso se ha instanciado de la subclase de Profile **Pre_trip_Information** definido en la jerarquía de perfiles (el *namespace profileHierarchy* enlaza con la ontología que define la jerarquía de perfiles para describir las clases Profile de los servicios de tráfico llamada **CategoriaServicios**) que nos sirve para identificar a perfiles que pertenecen a esta categoría de servicio. A continuación se completarán los valores de las propiedades de la clase Profile.

Las primeras propiedades a completar son las que permiten indicar cuales son las URI's de la instancia de *Service* y la instancia de *Process* que acompañan a este *Profile*. Las propiedades son *presentedBy* y *has_process* respectivamente.

```
<service:presentedBy
rdf:resource="http://localhost/DAML_files/TraficoService.daml#IncidenciasTraficoServ
ice"/>
<profile:has_process
rdf:resource="http://localhost/DAML_files/TraficoProcess.daml#IncidenciasTraficoProc
essModel"/>
```

Estas propiedades son importantes, porque cuando un agente cliente seleccione el perfil del servicio (su fichero *profile*), necesitará acceder a los ficheros *process* y *grounding* para leerlos y realizar la ejecución correcta del servicio. El acceso a las otras subontologías del servicio se realiza a través del fichero *Service*, el cual dispone de enlaces a todos los recursos que definen el servicio web.

El siguiente paso en la descripción del servicio será rellenar las propiedades de nombre y descripción del servicio.

```
<profile:serviceName>
       Incidencias_Trafico_Agent
</profile:serviceName>
<profile:textDescription>
       El agente proporciona un servicio de alerta de incidencias en
       la red de carreteras
</profile:textDescription>
```

El parámetro *textDescription* está orientado a que el proveedor describa su servicio mediante el lenguaje natural, pero se le podría dar otra funcionalidad. El campo de *textDescription* podría ser utilizado por los proveedores como un almacén de palabras claves (*keywords*) que tengan relación con el servicio, unas *keywords* que podrían servir para que el emparejador realice un nuevo filtro en la búsqueda de SW basado en palabras claves. Este nuevo filtro no sería semántico sino sintáctico, pero permitiría al sistema disponer de otros tipos para encontrar servicios que tengan relación con la petición de un usuario.

El siguiente paso consiste en completar las propiedades que sirven para especificar cual es el proveedor del servicio y la información de contacto. Cada proveedor lo puede hacer indicando los datos de la empresa, o los de un departamento de atención al cliente etc. Podemos poner tanta información de contacto como queramos. En nuestro caso hemos indicado datos ficticios sobre la DGT.

```
<profile:contactInformation>
<profile:Actor rdf:ID="#Trafico_contactos">
  <profile:name>IncidenciasTrafico</profile:name>
  <profile:title>Service Representative</profile:title>
  <profile:phone>96 333 4444</profile:phone>
  <profile:fax>96 333 555</profile:fax>
  <profile:email>dgt@dgt.es</profile:email>
  <profile:physicalAddress>
       somewhere 2, OnWeb, Valencia 22313, Spain
  </profile:physicalAddress>
  <profile:WebURL>
       http://www.dgt.es/IncidenciasTrafico.html
  </profile:WebURL>
</profile:Actor>
</profile:contactInformation>
```

En este momento, la clase *Profile* tiene definidos unas propiedades para especificar parámetros del servicio que especifican otras cualidades del servicio web que ofrece el proveedor. Estos parámetros no son obligatorios en todos los *Profile*, aunque es recomendable utilizarlos para una mayor expresividad en la descripción (radio geográfico, calidad del servicio etc.).

Para aprovechar todas las posibilidades de la clase *Profile*, se definen estos parámetros de servicio propuestos por el comité de DAML.

Analizando este servicio, se puede observar que actualiza varias veces al día su información, es gratuito y no necesita por parte del cliente, de demasiada especificación en los parámetros de entrada, dando como resultado información concreta. Además, está

proporcionado por las administraciones públicas (Ministerio de Interior), por lo que atendiendo a la métrica basada en factores de calidad propuesta en esta tesis para evaluar la calidad de SW de tráfico, el servicio de información de incidencias obtiene la siguiente calidad:

$$\text{Calidad} = [\text{Coste}(10) + \text{Actualización}(8) + \text{Manejo}(8) + \text{Ad. Públicas}(4) + \text{Seguridad}(9) + \text{Conf.Integridad}(8)\ ] \ / \ 5.4 = 8.70$$

Al obtener un valor comprendido entre el intervalo]8,10], el servicio web de incidencias de tráfico de la DGT adquiere el grado de *excelente* según la clasificación aportada en la tabla 8.1.

A su vez, se ha catalogado el servicio bajo dos clasificaciones distintas: UNSPSC de servicios, y la propuesta en esta tesis a partir de ISO-TC 204.

```
<profile:qualityRating>
   <profile:QualityRating rdf:ID="#Trafico-Rating">
     <profile:ratingName>Grado Calidad</profile:ratingName>
     <profile:rating
   rdf:resource="http://localhost/DAML_files/Concepts.daml#Excelente"/>
   </profile:QualityRating>
</profile:qualityRating>

<profile:serviceCategory>
   <profile:UNSPSC rdf:ID="UNSPSC-category">
     <profile:value> Traffic engineering </profile:value>
     <profile:code> 81102201 </profile:code>
   </profile:UNSPSC>
</profile:serviceCategory>

<profile:serviceCategory>
   <profile:TrafficServices
     rdf:about="http://localhost/DAML_files/CategoriaServicios.daml
                     #Pre_trip_Information" />
</profile:serviceCategory>
```

Otro parámetro utilizado es el del radio geográfico de actuación o validez del servicio web. Es un parámetro cuya restricción de rango de valores es una clase que representa un país o región y que está definida en una ontología específica. Como ya vimos en el capítulo 6, para este propósito y para otros ya especificados, construimos una ontología de países propia, para poder añadir además de países, las comunidades autónomas, provincias y ciudades de España.

La utilización de este parámetro en el *Profile* del servicio de incidencias de la DGT se realiza instanciando este parámetro mediante el país, que en el caso que nos ocupa tendrá el valor de “España” cuya definición fue dada en la ontología de “Geografia”:

```
<profile:serviceParameter>
   <profile:GeographicRadius rdf:ID="DGT-geographicRadius">
       <profile:serviceParameterName>
               DGT Geographicus Radius
       </profile:serviceParameterName>
       <profile:sParameter
rdf:resource="http://localhost/DAML_files/Geografia.daml#España"/>
   </profile:GeographicRadius>
</profile:serviceParameter>
```

Considerando una de las aportaciones descritas en el punto 8.3, especificamos valores añadidos mediante el uso de los nuevos parámetros. En nuestro ejemplo del servicio de información de incidencias de tráfico de la DGT, el proveedor puede decidir que el usuario que emplee este servicio necesite a su vez, de un servicio de alertas de tráfico como InfoVoz [Sam03]. Solo tenemos que rellenar una instancia de la propiedad *serviceParameter* y de la clase *AddOn_to*:

```
<profile:serviceParameter>
  <profile:AddOn_To rdf:ID="Enlace_otro_servicio">
    <profile:serviceParameterName>
        Enlace a Infovoz
    </profile:serviceParameterName>
    <profile:sParameter rdf:resource=
   "http://localhost/InfovozProfile.daml#Profile_Infovoz_Trafico_Service"/>
   </profile:AddOn_To>
</profile:serviceParameter>
```

La propiedad *sParameter* apunta al perfil del servicio de alertas de Tráfico Infovoz. De la misma manera y haciendo esta vez uso de la propiedad inversa, en el perfil de Infovoz, podemos indicar con *Added_Value* que este servicio es un valor añadido del servicio de incidencias de la DGT.

Una vez descritos estos parámetros del servicio, el siguiente paso es la descripción de los parámetros funcionales del servicio. Recordemos que éstos son los *inputs* y *outputs* que tiene un servicio, al igual que las precondiciones que se deben de cumplir antes de la ejecución del servicio, y los efectos que existen cuando el servicio es ejecutado.

En nuestro servicio de incidencias ya vimos los parámetros de entrada que poseía, y revisado el servicio, no es necesario que se cumpla ninguna precondición antes de la ejecución del servicio, por lo que se definen estos parámetros de la siguiente manera:

```
<profile:input>
  <profile:ParameterDescription rdf:ID="#tipoIncidencia">
  <profile:parameterName>incidencia</profile:parameterName>
  <profile:restrictedTo rdf:resource=
     "http://localhost/DAML_files/Sucesos.daml#Tipo_Incidencia"/>
  <profile:refersTo rdf:resource=
     "http://localhost/DAML_files/TraficoProcess.daml#Incidenciain"/>
  </profile:ParameterDescription>
</profile:input>

<profile:input>
  <profile:ParameterDescription rdf:ID="#Provincia">
  <profile:parameterName>provincia</profile:parameterName>
  <profile:restrictedTo
  rdf:resource="http://localhost/DAML_files/Geografia.daml#Provincia"/>
  <profile:refersTo
rdf:resource="http://localhost/DAML_files/TraficoProcess.daml#Provinciain"/>
  </profile:ParameterDescription>
</profile:input><profile:input>
  <profile:ParameterDescription rdf:ID="#Comunidad">
  <profile:parameterName>comunidad</profile:parameterName>
  <profile:restrictedTo rdf:resource=
       "http://localhost/DAML_files/Geografia.daml#Comunidad" />
  <profile:refersTo rdf:resource=
       "http:// localhost/DAML_files/TraficoProcess.daml#Comunidadin" />
  </profile:ParameterDescription>
</profile:input>
```

Como se puede apreciar, son los tres *inputs* que necesita el servicio web. A cada *input* se le ha asociado un nombre y se ha indicado la clase o valor que toma, es decir se le ha dado el valor semántico que tiene ese *input*, indicado a su vez, una referencia al *input* correspondiente en el *Process*.

Para el *output* del servicio se procede de forma análoga. Respecto a efectos de este servicio, como es un servicio de información, no se producen efectos. Los efectos son más comunes en servicios de e-commerce.

```
<profile:output>
  <profile:ParameterDescription rdf:ID="#Incidencia">
  <profile:parameterName>incidencia</profile:parameterName>
  <profile:restrictedTo rdf:resource=
       "http://localhost/DAML_files/Sucesos.daml#Incidente"/>
  <profile:refersTo rdf:resource=
       "http://localhost/DAML_files/TraficoProcess.daml#Incidenciaout" />
  </profile:ParameterDescription>
</profile:output>
```

Como ya hemos terminado de definir todos los parámetros funcionales del servicio ya podemos finalizar la instancia del *Profile* del servicio y la ontología.

```
</profileHierarchy:Pre_trip_Information >
</rdf:RDF>
```

Una vez concluido el *Profile*, éste ya sirve al proveedor para anunciarse y registrarse en cualquier repositorio o bases de datos de *Profiles* de SW. Si en algún

momento el servicio cambiase, el *Profile* debe modificarse para que sea coherente con lo que el servicio realiza.

8.4.3.3 Process y Grounding

Veamos ahora como se describe el *Process* y el *Grounding* de un "servicio web". Continuando con el servicio de incidencias de la DGT, inicialmente hay que diferenciar si el servicio está compuesto por varios subprogramas y que secuencia sigue, o si se trata de un servicio atómico en el que solo existe un único programa que ejecutar.

Una distinción importante cuando describimos un servicio, es la diferencia existente entre la descripción de un programa que se está haciendo accesible por web, y la descripción que se realiza para describir el servicio que éste provee. El *Process* y el *Grounding* de un servicio sirven para el primer tipo de descripción y el *Profile* para el segundo tipo de descripción.

Por lo tanto, para definir un servicio web semántico, debemos describir el servicio que realiza el programa. Como hemos indicado, la ontología DAML-S provee unas clases para la descripción física del programa, como son la clase *Process Model* y la clase *WSDLgrounding*, las cuales permiten indicar la composición del servicio y la interoperabilidad del mismo.

El primero de los pasos para la descripción física del programa con la ontología de *Process* consiste en definir el programa o servicio web como un proceso. En este caso se trata del servicio de información de estado de la red viaria de la DGT, un servicio que tiene un único subproceso, que es la ejecución de una consulta dados unos parámetros de entrada. Podría darse el caso de que se tuviera un servicio web compuesto, en el que se tendrían que distinguir los diferentes subprocesos individuales que lo componen, describiéndolos de forma individual.

Continuando con el ejemplo, al ser un único proceso se define como atómico, es decir, únicamente necesita una llamada o invocación para la ejecución del servicio, no existe una conversación extensa entre éste y el cliente, solamente hay una petición de ejecución de servicio y una repuesta a esta petición [Ank02].

8.4.3.3.1 Process

Antes de describir el *Process*, se deben de completar los atributos de la instancia de *Process Model* que sirven para indicar de que clase de tipo *Process* es el servicio, y a que instancia *Service*.DAML que identifica a nuestro servicio de la DGT se refiere para describir el servicio:

```
<process:ProcessModel rdf:ID="IncidenciasTraficoProcessModel" >
  <process:hasProcess rdf:resource="#IncidenciasTrafico" />
  <service:describes rdf:resource=
       "http://localhost/TraficoService#IncidenciasTraficoService"/>
</process:ProcessModel>
```

En este momento, es cuando se puede definir el proceso de nuestro servicio, indicando de que tipo de servicio se trata (*atomic* en nuestro caso). Cuando definimos nuestro proceso podemos añadirle restricciones a cualquiera de las propiedades que después se le definen. Por ejemplo, podemos añadir una restricción de cardinalidad en la que solo se permite introducir un único tipo de incidencia como entrada del servicio:

```
<daml:Class rdf:ID="IncidenciasTrafico">
  <rdfs:subClassOf rdf:resource="&process;#AtomicProcess"/>
  <rdfs:subClassOf>
     <daml:Restriction daml:cardinality="1">
        <daml:onProperty rdf:resource=”#Incidenciain”/>
      </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

Teniendo en cuenta que la ontología de *Process* describe unas propiedades asociadas a la clase *process* para definir los *inputs, outputs*, precondiciones y efectos del servicio y en este caso ***IncidenciasTrafico*** es una subclase de *AtomicProcess*, lo que debemos de hacer son subpropiedades de los IOPE's. En el caso que nos ocupa tenemos como entradas del servicio, el tipo de incidencia, la comunidad autónoma y la provincia, los cuales se definen en DAML-S 0.9 como subpropiedades de la propiedad *input* definida en la ontología de *Process*, solo que en este caso definiremos como dominio de la propiedad el proceso que lo utiliza (***IncidenciasTrafico***) como *input* y como rango la clase semántica que emplea para dar valores a esa entrada.

El aspecto de los *inputs* de nuestro servicio queda de la siguiente manera:

```
<rdf:Property rdf:ID="Incidenciain">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#IncidenciasTrafico"/>
  <rdfs:range rdf:resource=
     "http://localhost/DAML_files/Sucesos.daml#Tipo_Incidencia"/>
</rdf:Property>

<rdf:Property rdf:ID="Proviniciain">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#IncidenciasTrafico"/>
  <rdfs:range rdf:resource=
     "http://localhost/DAML_files/Geografia.daml#Provincia"/>
</rdf:Property>

<rdf:Property rdf:ID="Comunidadin">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#IncidenciasTrafico"/>
  <rdfs:range rdf:resource=
       "http://localhost/DAML_files/Geografia.daml#Comunidad_Autonoma"/>
</rdf:Property>
```

Para los *outputs* ocurre de la misma manera, es decir, las salidas son subpropiedades de la propiedad *output* definida para la clase *Process*. En el caso de nuestro servicio tenemos una salida que representa cada una de las incidencias que nos devuelve el mismo.

```
<rdf:Property rdf:ID="Incidenciaout">
  <rdfs:subPropertyOf rdf:resource="&process;#output"/>
  <rdfs:domain rdf:resource="#IncidenciasTrafico"/>
  <rdfs:range rdf:resource=
       "http://localhost/DAML_files/Sucesos.daml#Incidente"/>
</rdf:Property>
```

Los *inputs* de un servicio pueden ser obligatorios o pueden ser opcionales y los *outputs* obligatorios o condicionales. Este tipo de *outputs* condicionales al resultado de una ejecución del servicio, también han sido definidos por la ontología de *process* para que sean utilizados cuando se necesiten.

Con la definición de las propiedades de *inputs* y *outputs* de nuestro proceso, éste ya puede tanto ser utilizado de forma automática para ser invocado y ejecutado, o ser utilizado por otros para la composición de nuevos servicios. Aunque para la composición de servicios se le deben de definir otro tipo de propiedades del proceso que son importantes para una composición de servicios. Estas propiedades, si existen en el proceso, son las precondiciones que deben cumplirse antes de poder invocar al servicio y los posibles efectos que se tengan tras la ejecución del servicio.

Las precondiciones y efectos de un proceso se describen de la misma manera, es decir, como subpropiedades de las ya definidas para la clase *Process*, aunque para los efectos, al igual que los *outputs*, pueden ser declarados como efectos condicionales. Como en este servicio no tenemos ni precondiciones ni efectos al tratarse de un servicio de información no han sido definidas.

Una vez ya especificadas las propiedades de nuestro proceso, es decir, los *inputs*, *outputs*, solo hay que cerrar convenientemente la ontología del proceso para que ésta esté bien formada.

```
</rdf:RDF>
```

8.4.3.3.2 Grounding

Para completar la descripción semántica del servicio web de incidencias de la DGT debemos de completar la descripción física del servicio, es decir, completar las instancias correspondientes de las clases definidas para el *grounding*, además de definir el servicio web en un fichero WSDL asociado a nuestro Grounding.

Como para el resto de las ontologías, se definió una clase con varias propiedades que sirve para representar toda la información indicada en el *Process* del servicio. Ésta posee dos clases principales: *WSDLGrounding* para identificar el *Grounding* del

servicio y tener enlaces a todas las instancias que representan el emparejamiento entre *operation* de WSDL con procesos atómicos del servicio, y otra clase *WSDLAtomicGrounding* que sirve para hacer instancias de la relación anterior, es decir, relacionar todas las partes de un mensaje WSDL dada una descripción de un proceso atómico en DAML-S.

Por lo tanto, el fichero DAML que representa el *grounding* de un servicio está compuesto por las instancias necesarias de las clases definidas. En nuestro caso, tendremos una instancia de la clase *WSDLGrounding* con una propiedad para enlazar con el fichero *Service* que describe al servicio, al igual que otra propiedad con la instancia de la clase *WSDLAtomicGrounding* para representar el único proceso atómico que posee el servicio de información de estado de la red viaria de la DGT.

```
<grounding:WsdlGrounding rdf:ID="IncidenciasTraficoServiceGrounding">
   <service:supportedBy
        rdf:resource="&trafico_service;#IncidenciasTraficoService"/>
   <comment>
        El servicio es solo un proceso, por eso se hará solo una
        instancia de la propiedad hasAtomicProcessGrounding.
   </comment>
   <grounding:hasAtomicProcessGrounding
        rdf:resource="#IncidenciasTraficoGrounding"/>
</grounding:WsdlGrounding>
```

A continuación se muestra la instancia de la clase *WSDLAtomicProcessGrounding* del servicio, que enlaza con el fichero WSDL que sirve para especificar cómo debe ser el *grounding* del servicio empleando este lenguaje. Esta instancia posee propiedades para hacer las referencias a cada una de las partes del mensaje WSDL, indicando que valores y correspondencias con las propiedades del proceso atómico descrito en DAML-S debe realizar.

```
<grounding:WsdlAtomicProcessGrounding
       rdf:ID="IncidenciasTraficoGrounding">

  <grounding:damlsProcess
       rdf:resource="&trafico_process;#IncidenciasTrafico"/>

  <grounding:wsdlOperation>
    <grounding:WsdlOperationRef>
       <grounding:portType>
        <xsd:uriReference
 rdf:value="&trafico_wsdl_grounding;#IncidenciasTrafico_PortType"/>
      </grounding:portType>
        <grounding:operation>
        <xsd:uriReference
rdf:value="&trafico_wsdl_grounding;#IncidenciasTrafico_operation"/>
        </grounding:operation>
     </grounding:WsdlOperationRef>
  </grounding:wsdlOperation>

  <grounding:wsdlInputMessage>
     <xsd:uriReference
    rdf:value="&trafico_wsdl_grounding;#IncidenciasTrafico_Input"/>
  </grounding:wsdlInputMessage>
  <grounding:wsdlInputs rdf:parseType="daml:collection">
       <grounding:WsdlInputMessageMap>
         <grounding:damlsParameter
              rdf:resource="&trafico_process;#Incidenciain"/>
         <grounding:wsdlMessagePart>
            <xsd:uriReference
                       rdf:value="&trafico_wsdl_grounding;#incidencia"/>
         </grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
       <grounding:WsdlInputMessageMap>
         <grounding:damlsParameter
              rdf:resource="&trafico_process;#Provinciain"/>
         <grounding:wsdlMessagePart>
            <xsd:uriReference
                       rdf:value="&trafico_wsdl_grounding;#provincia"/>
         </grounding:wsdlMessagePart>
       </grounding:WsdlInputMessageMap>

       <grounding:WsdlInputMessageMap>
         <grounding:damlsParameter
              rdf:resource="&trafico_process;#Comunidadin"/>
         <grounding:wsdlMessagePart>
            <xsd:uriReference
                       rdf:value="&trafico_wsdl_grounding;#comunidad"/>
         </grounding:wsdlMessagePart>
       </grounding:WsdlInputMessageMap>
    </grounding:wsdlInputs>

    <grounding:wsdlOutputMessage>
   rdf:value="&trafico_wsdl_grounding;#IncidenciasTrafico_Output”/>
    </grounding:wsdlOutputMessage>

    <grounding:wsdlOutputs rdf:parseType="daml:collection">
       <grounding:WsdlOutputMessageMap>
         <grounding:damlsParameter
              rdf:resource="&trafico_process;#Incidenciaout"/>
         <grounding:wsdlMessagePart>
            <xsd:uriReference
                       rdf:value="&trafico_wsdl_grounding;#incidenciao"/>
         </grounding:wsdlMessagePart>
       </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutputs>
```

```
    <grounding:wsdlReference>
       <xsd:uriReference
         rdf:value="http://www.w3.org/TR/2001/NOTE-wsdl-20010315"/>
    </grounding:wsdlReference>
    <grounding:otherReference>
       <xsd:uriReference rdf:value=
       "http://www.w3.org/TR/2001/NOTE-wsdl-20010315"/>
    </grounding:otherReference>

    <grounding:otherReference>
        <xsd:uriReference
              rdf:value="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </grounding:otherReference>
    <grounding:otherReference>
         <xsd:uriReference
              rdf:value="http://schemas.xmlsoap.org/soap/http/"/>
    </grounding:otherReference>
    <grounding:wsdlDocument>
        <xsd:uriReference rdf:value="&trafico_wsdl_grounding;"/>
    </grounding:wsdlDocument>
    </grounding:WsdlAtomicProcessGrounding>
</rdf:RDF>
```

Una vez concluida la descripción del *Grounding* de todos los procesos atómicos del servicio, se debe construir el fichero WSDL que describa todos los mensajes, la descripción del protocolo que se utilizará (SOAP) así como el mecanismo de transporte de mensajes (HTTP) que se empleará para el intercambio de mensaje y ejecución del servicio.

El fichero *Grounding* de WSDL consiste en la definición de cada una de las partes de la descripción de un servicio web: los mensajes de entrada y salida, la elección del protocolo de transporte, la descripción del mensaje SOAP etc.

En la primera parte de un fichero WSDL se definen los *namespace* y los mensajes de entrada y salida que se emplearan en el servicio. En los mensajes se incluye la definición de cada uno de los *inputs* y *outputs* y cual es el parámetro correspondiente asociado al *process* definido.

```
<?xml version="1.0"?>
  <definitions name="TraficoGrounding"
    targetNamespace="http://localhost/TraficoGrounding.wsdl"
    xmlns:tns="http://localhost/TraficoGrounding.wsdl"
    xmlns:Trafico="http://localhost/TraficoProcess.daml"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:daml-s-wsdl="http://www.daml.org/services/daml-s/wsdl/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">

    <!-- ..................................... -->
    <!-- Messages definition for IncidenciasTrafico -->
    <message name="IncidenciasTrafico_Input">
      <part name="incidencia"
              daml-s-wsdl:daml-s-parameter="Trafico:#Incidenciain"/>
      <part name="provincia"
              daml-s-wsdl:daml-s-parameter="Trafico:#Provinciain"/>
      <part name="comunidad"
              daml-s-wsdl:daml-s-parameter="Trafico:#Comunidadin"/>
    </message>
    <message name="IncidenciasTrafico_Output">
      <part name="incidenciao"
              daml-s-wsdl:daml-s-parameter="Trafico:#Incidenciaout"/>
    </message>
```

Las siguientes partes de la descripción WSDL del servicio consisten en la definición del *operation* que corresponde al proceso atómico y la asociación de los mensajes anteriormente definidos a este servicio:

```
<portType name="IncidenciasTrafico_PortType">
    <operation name="IncidenciasTrafico_operation"
       daml-s-wsdl:daml-s-process="Trafico:#IncidenciasTrafico">
         <input message="IncidenciasTrafico_Input"/>
         <output message="IncidenciasTrafico_Output"/>
    </operation>
</portType>
```

Después se debe enlazar esta *operation* de WSDL con la descripción SOAP de intercambio de mensajes, el protocolo de transporte en el que se deben de encapsular y la *operation* definida para la que se realiza esta descripción SOAP.

```
<binding name="IncidenciasTrafico_SoapBinding"
type="tns:#IncidenciasTrafico_PortType">
      <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="IncidenciasTrafico_operation">
        <soap:operation soapAction="tns:#IncidenciasTrafico "/>
        <input>
          <soap:body
           parts="incidencia provincia comunidad"
           use="encoded"
           namespace="http://localhost/TraficoService.daml"/>
        </input>
        <output>
          <soap:body
           parts="incidenciao"
           use="encoded"
           namespace="http://localhost/TraficoService.daml"/>
        </output>
      </operation>
</binding>
```

Por último se define la descripción WSDL a la que se le añaden los enlaces a las definiciones de la *operation* y la descripción SOAP, así como el enlace de dirección SOAP utilizado para asignar una dirección a un puerto (un URI). Con estos tres parámetros ya se dispone de la descripción del servicio mediante WSDL, y puede ser utilizada por agentes web para la ejecución de los servicios.

```
<service name="IncidenciasTrafico_Service">
      <documentation> WSDL descripcion para obtener las incidencias
                        de trafico </documentation>
      <port name="IncidenciasTrafico_Port"
            binding="tns:#IncidenciasTrafico_SoapBinding">
        <soap:address
location="http://localhost:8080/axis/services/servicioincidencias"/>
      </port> </service>
</definitions>
```

8.5 SERVICIO WEB COMPUESTO

Otro de los objetivos de la *Web Semántica*, y en especial, dentro de la investigación de un lenguaje que sirva para describir SW, es permitir la interoperabilidad y composición de servicios.

Para el sistema emparejador, se ha desarrollado un servicio web compuesto llamado *Prevision_Multilenguaje*. Este servicio web consiste en un servicio de previsiones y recomendaciones del tráfico para los próximos días, que sirve la información en el idioma que indique el cliente. Para implementar este servicio, se tuvo que realizar como la composición de dos SW: el servicio de previsiones de tráfico de la DGT y el servicio web de traducción de www.elmundo.es. La estructura de control que sigue este servicio es una secuencia, debido a que la salida del servicio de previsiones de la DGT posee la descripción de la misma escrita en castellano y ésta nos servirá como entrada en el traductor como el texto a traducir al idioma seleccionado por el cliente. Al final, la salida del servicio es una instancia de *Previsión* de tráfico cuya descripción está traducida al lenguaje indicado por el cliente. La figura 8.4 describe este servicio compuesto.

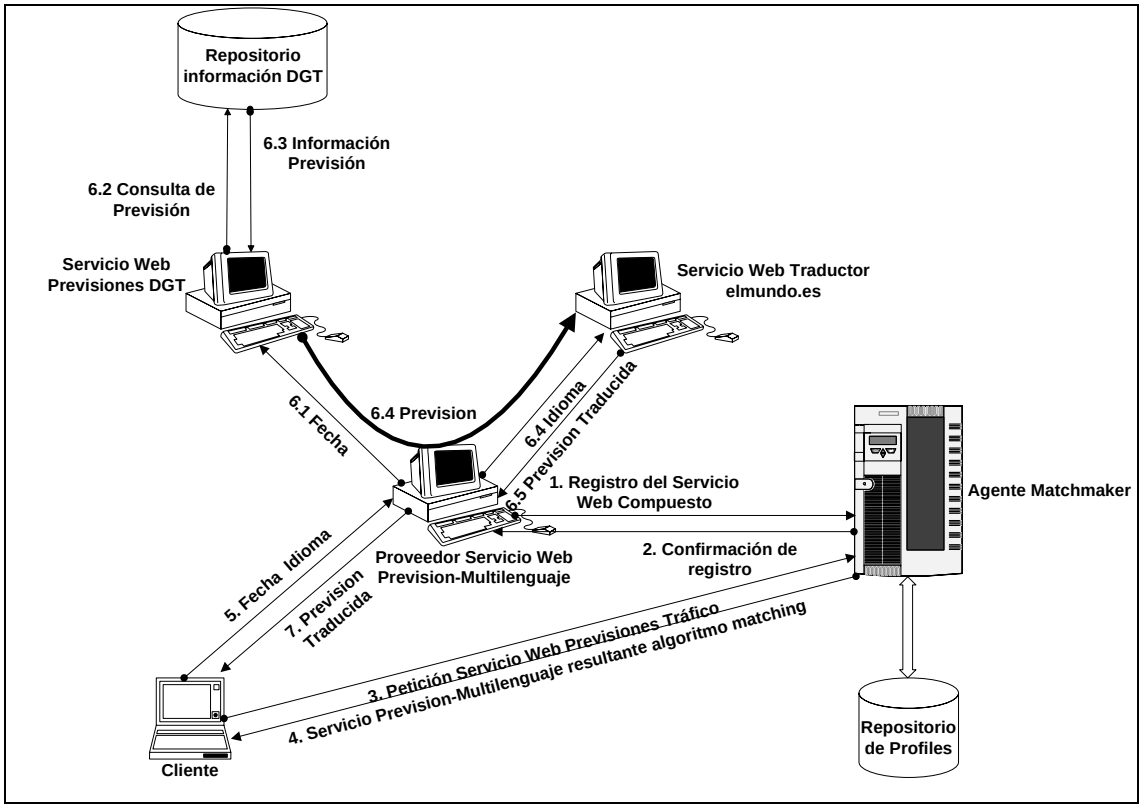


Figura 8.4 Arquitectura del servicio web compuesto Previsiones Multilenguaje

La figura 8.5 muestra el diagrama de secuencia de la ejecución del servicio, incluido la necesidad del Wrapper para que rellene el repositorio de información de la DGT con las previsiones correspondientes, como también la necesidad del *Wrapper* para capturar la traducción de la página web de elmundo.es con la traducción de la previsión.

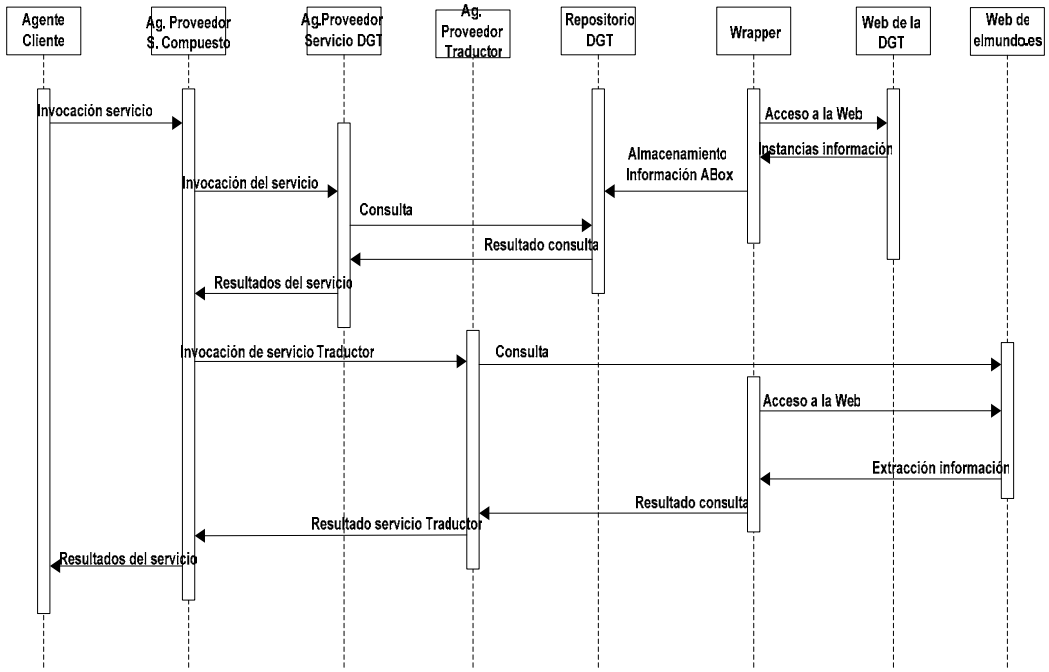


Figura 8.5: Secuencia ejecución servicio web previsiones multilingüe

El servicio web se anunciará en un *Profile* como un servicio más, y en él se definen como parámetros funcionales dos entradas: la fecha para la que se quiere la previsión y el idioma en el que se espera la información, y como salida se define la previsión.

```
<profile:hasInput>
   <profile:Input rdf:ID="Dia">
      <profile:parameterType
       rdf:resource="http://localhost/OWL_files/Time.owl#Date"/>
    </profile:Input>
</profile:hasInput>

<profile:hasInput>
   <profile:Input rdf:ID="Idioma">
      <profile:parameterType
       rdf:resource="http://localhost/OWL_files/Concepts.owl#Idioma"/>
    </profile:Input>
</profile:hasInput>

<profile:hasOutput>
    <profile:Output rdf:ID="Salida">
      <profile:parameterType
       rdf:resource="http://localhost/OWL_files/Sucesos.owl#Prevision"/>
    </profile:Output>
</profile:hasOutput>
```

En el fichero *Process* se especifica la composición del servicio, el modelo que sigue, y la clase que representa el proceso que es el servicio. Esta clase, será subclase de *CompositeProcess* para de esta forma utilizar sus propiedades y definir la composición del servicio.

Una de las propiedades definida para este tipo de procesos (*composedOf*) será útil para indicar una restricción sobre el proceso que identifica nuestro servicio web y así especificar que tipo de estructura de control forma el servicio web, y cuales son los procesos que forman este servicio.

```
<daml:Class rdf:ID="Compuesto">
  <rdfs:subClassOf rdf:resource="&Process;#CompositeProcess" />
  <rdfs:subClassOf>
     <daml:Restriction>
       <daml:onProperty rdf:resource="&Process;#composedOf" />
         <daml:toClass>
           <daml:Class>
             <daml:intersectionOf rdf:parseType="daml:collection">
               <daml:Class rdf:about="&Process;#Sequence" />
                 <daml:Restriction>
                   <daml:onProperty rdf:resource="&Process;#components" />
                     <daml:toClass>
                       <daml:Class>
                         <process:listOfInstancesOf
                           rdf:parseType="daml:collection">
                           <daml:Class rdf:about="#Prevision_Process"/>
                              <daml:Class rdf:about="#Traductor_Process" />
                          </process:listOfInstancesOf>
                       </daml:Class>
                      </daml:toClass>
                   </daml:Restriction>
             </daml:intersectionOf>
           </daml:Class>
       </daml:toClass>
     </daml:Restriction>
  </rdfs:subClassOf>
….
</daml:Class>
```

Como en el resto de servicios, se han definido propiedades para representar los parámetros de entrada y salida, análogos a los descritos en el *Profile* del servicio:

```
<rdf:Property rdf:ID="CompuestoFecha">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#Compuesto"/>
  <rdfs:range rdf:resource="http://localhost/Time.daml#Date"/>
</rdf:Property>

<rdf:Property rdf:ID="CompuestoIdioma">
  <rdfs:subPropertyOf rdf:resource="&process;#input"/>
  <rdfs:domain rdf:resource="#Compuesto"/>
  <rdfs:range rdf:resource="http://localhost/Concepts.daml#Idioma"/>
</rdf:Property>

<rdf:Property rdf:ID="CompuestoPrevision">
  <rdfs:subPropertyOf rdf:resource="&process;#output"/>
  <rdfs:domain rdf:resource="#Compuesto"/>
  <rdfs:range rdf:resource="http://localhost/Conceptsdgt.daml#Prevision"/>
</rdf:Property>
```

El siguiente paso es definir las clases que representen los dos procesos atómicos y las correspondientes propiedades que describen las entradas y salidas del servicio web. De esta manera, se definió un proceso *Prevision_Process* que nos identifica el servicio web de Previsiones de tráfico de la DGT, al que se le ha definido una propiedad de entrada *FechaPP* y una propiedad que represente la salida *PrevisionPP* (de tipo *Prevision*, o de tipo S*tring*), y como segundo proceso *Traductor_Process*, que identifica al servicio web de traducción que ofrece elmundo.es, con dos propiedades de entrada: *EntradaTP* (de tipo *Prevision* o S*tring*) y *IdiomaTP* (de tipo *Idioma*) y una propiedad de salida *PrevisionTP* (de tipo *Previsión*).

Una característica especial de este tipo de servicios es la posibilidad de indicar que valores de los parámetros son comunes entre sí dentro de las diferentes entradas y salidas que hay en los procesos atómicos que componen el servicio global. Por ejemplo, se puede indicar que la salida de un proceso atómico es entrada de otro proceso dentro del mismo servicio web, o ya es salida del servicio web. Para estos emparejamientos de valores, DAML-S proporciona las propiedades adecuadas, y solo hace falta definirlas dentro de las descripciones de los procesos en los que se realice algún emparejamiento de alguna de sus propiedades con otras propiedades de otro proceso dentro del mismo servicio Web.

En nuestro servicio web, se tienen varios emparejamientos de propiedades de procesos:

1. Propiedad de entrada *CompuestoFecha* del proceso Compuesto debe tener el mismo valor que la propiedad de entrada *FechaPP* del proceso atómico *Prevision_Process*.
2. Propiedad de entrada *CompuestoIdioma* del proceso Compuesto debe tener el mismo valor que la propiedad de entrada *IdiomaTP* del proceso atómico *Traductor_Process*.
3. Propiedad de salida *CompuestoPrevision* del proceso Compuesto debe tener el mismo valor que la propiedad de salida *PrevisionTP* del proceso atómico *Traductor_Process*.
4. Propiedad de salida *PrevisionPP* del proceso *Prevision_Process* debe tener el mismo valor que la propiedad de entrada *EntradaTP* del proceso atómico *Traductor_Process*.

```
<rdf:Description rdf:about="#Compuesto">
  <process:sameValues rdf:parseType="daml:collection">
    <process:ValueOf process:atClass="#Compuesto"
        process:theProperty="#CompuestoFecha"/>
    <process:ValueOf process:atClass="#Prevision_Process"
        process:theProperty="#FechaPP"/>
  </process:sameValues>
  <process:sameValues rdf:parseType="daml:collection">
    <process:ValueOf process:atClass="#Compuesto"
        process:theProperty="#CompuestoIdioma"/>
    <process:ValueOf process:atClass="#Prevision_Process"
        process:theProperty="#IdiomaTP"/>
  </process:sameValues>
  <process:sameValues rdf:parseType="daml:collection">
    <process:ValueOf process:atClass="#Compuesto"
        process:theProperty="#CompuestoPrevision"/>
    <process:ValueOf process:atClass="#Traductor_Process"
        process:theProperty="#PrevisionTP"/>
  </process:sameValues>
</rdf:Description>
```

```
<rdf:Description rdf:about="#Traductor_Process">
  <process:sameValues rdf:parseType="daml:collection">
    <process:ValueOf process:atClass="#Prevision_Process"
        process:theProperty="#PrevisionPP"/>
    <process:ValueOf process:atClass="#Traductor_Process"
        process:theProperty="#EntradaTP"/>
  </process:sameValues>
</rdf:Description>
```

Una vez emparejados los parámetros funcionales de todos los *Process*, el proveedor solo tiene que completar los ficheros de *grounding*, tanto .daml como .wsdl, del servicio web para poder ser invocado y accedido por clientes. Para un servicio compuesto no existe ninguna característica especial para definir el *grounding*, ya que en este tipo de SW, el *Grounding.daml* consiste en definir las instancias de *WsdlAtomicProcessGrounding* de cada uno de los procesos atómicos que componen el servicio web, es decir, como si se especificase el *grounding* de cada uno de los SW atómicos que forman el servicio web completo. Para el fichero *Grounding.wsdl* se realiza de la misma manera, es decir, se define de manera independiente todas las descripciones WSDL, con todos sus componentes (mensajes, *operation*, *port type*), de cada uno de los procesos atómicos que lo componen, y solo se define la conjunción de las dos descripciones atómicas al rellenar los parámetros más específicos de nombre del servicio y los *port* (conjunto de operaciones que lo definen) que componen el servicio web.

En nuestro caso, se ha definido el grounding de los procesos atómicos *Prevision_Process* y *Traductor_Process* que forman el proceso *Compuesto_Process* que representa el servicio, tanto en el fichero .daml como en el .wsdl

```
<grounding:WsdlGrounding rdf:ID="CompuestoGrounding">
   <service:supportedBy
        rdf:resource="&compuesto_service;#CompuestoService"/>
   <grounding:hasAtomicProcessGrounding
        rdf:resource="#PrevisionGrounding"/>
   <grounding:hasAtomicProcessGrounding
        rdf:resource="#TraductorGrounding"/>
</grounding:WsdlGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="PrevisionGrounding">
   <grounding:damlsProcess
       rdf:resource="&compuesto_process;#Prevision_Process"/>
   ...
</grounding:WsdlAtomicProcessGrounding>

<grounding:WsdlAtomicProcessGrounding rdf:ID="TraductorGrounding">
   <grounding:damlsProcess
       rdf:resource="&compuesto_process;#Traductor_Process"/>
   ...
</grounding:WsdlAtomicProcessGrounding>
```

La descripción WSDL del servicio es la definición de los dos *port* que definen los procesos que lo componen, es decir los dos procesos atómicos definidos tanto en el fichero *Grounding.daml*, como en el fichero *Proces*s*.daml* del servicio web:

```
<service name="Compuesto_service">
      <documentation>
           WSDL description of request de previsiones de tráfico en ingles
      </documentation>
      <port name="Prevision_Port" binding="tns:#Prevision_SoapBinding">
            <soap:address location="http://localhost:8080/axis/services/servicioPrevision"/>
      </port>
       <port name="Traductor_Port" binding="tns:#Traductor_SoapBinding">
            <soap:address location="http://localhost:8080/axis/services/servicioTraductor"/>
      </port>
</service>
```

Terminada la descripción del servicio web, el proveedor solo ha de registrarse en el sistema *emparejador* para permitir ser descubierto y esperar peticiones de ejecución del servicio por parte de clientes, de tal forma que el cliente desconoce si se trata de un servicio compuesto o atómico.

8.6 OTROS SERVICIOS DESCRITOS SEMÁNTICAMENTE

Para complementar el sistema se implementaron otros servicios de información de tráfico cuyo origen fueros sitios web heterogéneos:

- servicio de previsiones de tráfico vial (DGT),
- servicio de restricciones de vehículos pesados (DGT),
- servicio de restricciones de mercancías peligrosas (DGT),
- servicio de incidencias del Servei Català de Trànsit (SCT),
- servicio de cálculo de rutas interurbanas (CAMPSA), y
- servicio de alertas de incidencias de tráfico, InfoVoz (Instituto de Robótica).

Estos servicios han sido detallados en la URL http://robotica.uv.es/tesis/javi/sws.html y en el CD-ROM que acompaña esta memoria.

8.7 CONCLUSIONES

En este capítulo se ha propuesto un marco de trabajo para la conversión de páginas web tradicionales de información de tráfico en SWS, de manera que la información aportada por ellas pueda ser almacenada con la adición de significado y a su vez se consiga potenciar nuevas capacidades que en un principio no existían.

Se ha introducido y validado la utilidad de un nuevo parámetro no funcional en las ontologías DAML-S y OWL-S, correspondiente al **valor añadido** en determinados servicios. La adición de este nuevo parámetro supone un nivel de descubrimiento de servicio más completo, en cuanto a que el cliente no solamente obtiene aquel servicio que está buscando, sino de forma suplementaria, consigue otra serie de servicios que complementan al primero, y los cuales pudieran ser necesitados en búsquedas posteriores. De igual manera, desde el punto de vista de proveedor, se facilita el anuncio de servicios que complementan o pudieran interesar al servicio anunciado. La adición de este nuevo parámetro se ha realizado mediante la inclusión de dos nuevas propiedades (*AddOn_To* y *Added_Value*), una inversa de la otra, que aportan una mayor expresividad en cuanto a la caracterización tanto en el servicio complementado como en el servicio complemento.

Se ha puesto de manifiesto la necesidad de crear una jerarquía de perfiles para este tipo específico de servicios, basándonos en la categorización de servicios de tráfico, marco de trabajo de la ISO. Las categorizaciones recomendadas hasta la fecha por la coalición se fundamentan en las taxonomías UNSPSC y NAICS, pero éstas son demasiado generales para el tipo específico de servicios de tráfico, y por tanto se ha tenido en cuenta el trabajo de normalización llevado a cabo por el grupo ISO TC204 WG1 *(Architecture, Taxonomy* & *Terminology* de la *International Organization for Standardization*) en la norma "*Transport Information and Control Systems (TICS) – Reference Model. Architecture(s) for the TICS Sector – Part 1: TICS Fundamental Services*" la cual representa la primera clasificación normalizada de servicios ITS.

Se ha realizado un aporte basado en una propuesta para medir la calidad de servicio en servicios de tráfico. Se han estudiado otras propuestas, poniéndose de manifiesto la dificultad de aplicar dichos modelos a los SW de tráfico debido a las características intrínsecas de estos servicios. Hay que considerar que éstos ofrecen información de tráfico y que al contrario que la mayoría de servicios de la web actual no poseen una orientación de negocio. Para el desarrollo de esta propuesta se han tenido en cuenta diversos factores que pueden incidir en la calidad de un servicio, y los cuales fueron ya analizados por otros autores, así como también se han incluido otros nuevos factores que reflejan la peculiaridad de este nuevo tipo de servicios, como por ejemplo, la pertenencia a una administración pública etc. Por último, se ha propuesto una métrica que recoge todos estos factores de calidad a través de variables y la cual ha permitido confeccionar una escala de valores apropiada.

Tanto la clasificación bajo la jerarquía de perfiles creada como el valor de QoS asociado a un servicio específico de tráfico, son fundamentales en el descubrimiento de éste, ya que suponen mecanismos de filtrado en el proceso de búsqueda, permitiendo su optimización. Tanto una descripción más detallada como su evaluación mediante diferentes casos de prueba en el algoritmo propuesto en esta tesis, han sido expuestas en el capítulo 7.

Por último, se ha especificado una arquitectura de integración de varios servicios que ponen de manifiesto la utilidad de este tipo de sistemas, para por ejemplo, obtener información en un idioma diferente al originalmente ofertado. Los servicios especificados han hecho uso de estas propuestas y se han utilizado en determinadas pruebas de validez realizadas que buscaban la comprobación de la integración de todo el sistema y las cuales serán descritas en el capítulo 10.

SECCIÓN IV: OTROS RESULTADOS.

CAPÍTULO 9

ONTOSERVICE: ENTORNO DE EDICIÓN DE PERFILES DE SERVICIO Y VISUALIZACIÓN / VERIFICACIÓN DE ONTOLOGÍAS.

9.1 RESUMEN

En este capítulo se expone el desarrollo de un entorno de edición de perfiles de servicio de tráfico que adicionalmente integra la visualización y verificación de consistencia de ontologías que definan los conceptos con los cuales interactúa un SW.

Primeramente se analizan los diferentes entornos que existen en la actualidad para justificar el motivo de la decisión de construir un entorno de edición propio en nuestro sistema. A continuación se detallan las características de éste, atendiendo a los diferentes elementos que lo integran, para posteriormente especificar el aspecto de generación de gráficos y el uso de diferentes tecnologías emergentes como SVG, que permitan alcanzar nuestro objetivo mediante la aplicación de un algoritmo de visualización. Se muestra la correspondencia entre la sintaxis de los lenguajes de representación de ontologías y una simbología mediante iconos, y el uso de la herramienta para no solo visualizar la ontología sino proceder a su verificación.
Posteriormente, se especifica la función principal de la herramienta como la búsqueda y la llamada a servicios determinados a partir de la generación de un perfil.
Para describir esta función de búsqueda se detalla en que consiste la generación de un perfil y las posibilidades que brinda la herramienta para poder elegir adecuadamente los parámetros necesarios, mediante la interacción del usuario con la ontología visualizada.
Para optimizar el proceso de búsqueda, se expone en que ha consistido el uso de perfiles de usuario (proxy) así como su implementación. Mediante estos perfiles un usuario registrado podrá beneficiarse del uso de caches que almacenen sus últimas consultas (perfil de usuario) incrementando de esta manera la velocidad del sistema, debido a que los usuarios probablemente buscarán un número reducido de tipos de servicio y por tanto su búsqueda no siempre será necesaria.

9.2 INTRODUCCIÓN

Al iniciar el desarrollo de esta herramienta se partió de una revisión bibliográfica relacionada con herramientas existentes para la edición de ontologías que permitieran la generación de código en diferentes lenguajes de marcado semántico (como DAML+OIL y OWL) y se comparó la funcionalidad de algunas de ellas, dentro de las que se cuentan

DUET [DUET03], OntoSaurus [OntoSaurus04], OntoEdit [OntoEdit04], SWOOP [SWOOP04], WebODE [WebODE04], WebOnto [WebOnto04], Protégé [PROTEGE04], OilEd [OILEd04]. Dos trabajos importantes relacionados con la comparación de algunas de estas herramientas han sido desarrollados por Corcho et al. [Cor02] ,[Ont02].

La mayoría de herramientas de edición que soportan OWL-S, toman como base la descripción WSDL del servicio Web proporcionada, para después emplear una herramienta de traducción de WSDL a OWL-S la cual provee de esqueleto a cada uno de los componentes de este último lenguaje, a partir de la información aportada por el WSDL. En una fase posterior, los proveedores pueden completar esta descripción generada automáticamente.

En relación con herramientas para descripción de SWS las principales propuestas que se han encontrado son A-Match [A-Match04] y OWL-S Editor [OWLSE05] de Carnegie-Mellon University que permiten especificar servicios compuestos y establecer flujos de datos, OWL-S *Editor for Semantically Enabled Web-Services* del Departamento de Ciencias de la Computación e I.A. de la Universidad de Malta [OWLSEdit04] con tratamiento de tres tipos diferentes de composición como son *Sequence*, *If-Then-Else* y *Split*, y varias herramientas de Mindswap (*Maryland Information and Network Dynamics Lab Semantic Web Agents Project*) de la Universidad de Maryland, para validación, traducción, así como invocación de servicios atómicos y algunos tipos de servicios compuestos [OWLSML04] [Sir04]. A partir del estudio realizado a las herramientas antes mencionadas, se pudo observar la integración en cada una de ellas de un razonador aunque difieren en su elección (Jena, Pellet, JTP etc) y aunque proveen facilidades para la edición de perfiles, no hay una integración directa con la visualización de ontologías de descripción de conceptos, que nos permita abordar de una forma eficiente la construcción de perfiles basados en dichas ontologías. Es necesario destacar, que en línea con nuestros requerimientos, en el momento de escribir estas líneas está siendo desarrollado un plugin para Protégé [OWLSRI04] el cual toma como base herramientas de traducción aportadas por el grupo de la Universidad de Maryland. Como crítica a este último editor, decir que no aporta los elementos indispensables de independencia y portabilidad para su uso en otro tipo de editores básicos de OWL o DAML+OIL como los citados anteriormente o su posible integración en plataformas de agentes como es nuestro caso de estudio.

Por tanto, la primera necesidad que surgió fue la falta de una herramienta que combinase la visualización de ontologías con la creación de perfiles de búsqueda y que fuera independiente de cualquier editor básico. A partir de este análisis se inició la realización de una herramienta basada en la integración de capacidades para definición de perfiles de SWS con visualización y verificación de consistencia de los conceptos sobre los cuales interactúa un determinado servicio.

La herramienta implementada tiene por tanto dos funcionalidades independientes, a la vez que complementarias. Por un lado es un visualizador/verificador de ontologías, las cuales pueden estar descritas indiferentemente en el lenguaje DAML+OIL o bien en OWL. Por otro, es una herramienta de búsqueda de servicios a partir de un perfil de búsqueda generado automáticamente.

9.3 CARACTERÍSTICAS DE LA HERRAMIENTA *ONTOSERVICE*

La definición de perfiles semánticos está encaminada a la búsqueda de servicios y por lo tanto, orientado al usuario, no al proveedor de servicios. Por este motivo, únicamente se genera el fichero de perfil de servicio (*Profile*) a partir de una plantilla siendo innecesarios los ficheros de *process* y *grounding*. El usuario puede almacenar esta plantilla en el disco duro o bien, dado que la herramienta puede integrarse en una plataforma de agentes, realizar el envío del perfil generado a un agente encargado de llevar a cabo la búsqueda del servicio web e incluso lanzar la ejecución de este agente.

9.3.1 Visualizador/Verificador de ontologías

9.3.1.1 Descripción de los elementos

En relación con el manejo de las ontologías de concepto, la herramienta implementada incluye un visualizador de ontologías descritas en DAML u OWL, que permite la carga de ficheros locales o remotos a partir de una URL. Un aspecto importante relacionado con la visualización gráfica de ontologías es la necesidad de presentar la información de manera escalonada, adaptada a las necesidades del usuario. Una información escasa puede resultar insuficiente, mientras que un exceso puede restar claridad a los conceptos. Por este motivo, se han definido tres capas, las cuales permiten activar o desactivar la visualización de las instancias (*Instances layer*), las dependencias entre clases (*Class Dependencies layer*) y las relaciones entre clases (*Object Properties layer*). Ver figura 9.1.

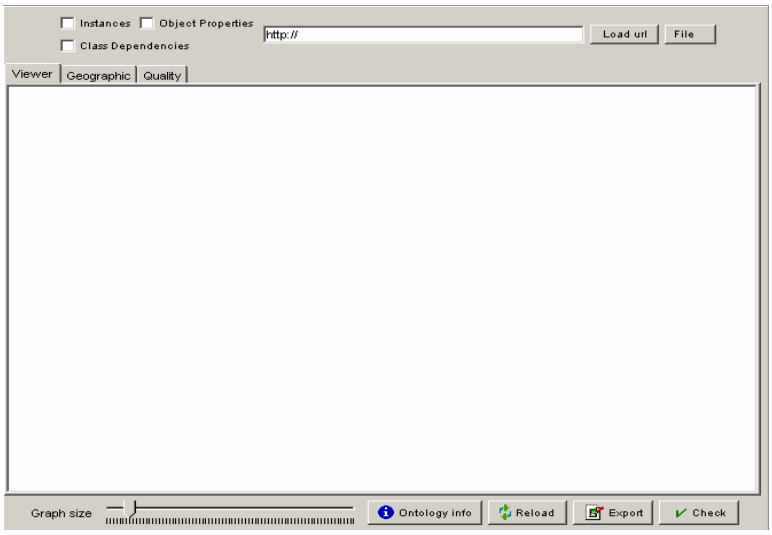


Figura 9.1 Area de visualización

Por defecto, con todas las capas desactivadas, lo que se muestra es la jerarquía de clases, expresada con la etiqueta *subClassOf*.
En la parte inferior de la interfaz, existen cuatro botones con funcionalidades diferentes. El primero de ellos denominado "*Ontology info*", abre un cuadro de diálogo donde se

muestra información general asociada con la ontología (figura 9.2). Esta información incluye:

- Número total de clases cargadas.
- Propiedades cargadas.
- Información contenida en la etiqueta '*Ontology*', como la versión, comentarios asociados y lista de *imports* utilizados.
- Lista de espacios de nombre y clases pertenecientes a cada espacio de nombres.

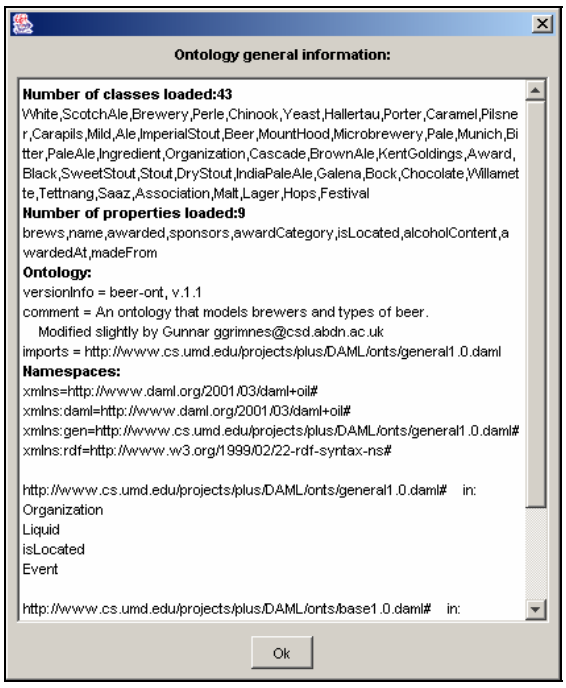


Figura 9.2 Información ontológica, detalles

El segundo botón, etiquetado como "Reload" se encarga de actualizar la ontología visualizada. Esto permite aplicar los posibles cambios, tanto a nivel de código fuente, como a nivel de visualización (activación o desactivación de capas, tamaño del gráfico etc.). El botón llamado "Export" permite almacenar el gráfico visualizado en un formato de salida determinado. La lista de formatos disponibles es GIF, JPG, PNG, SVG [SVG03], PostScript, PostScript con anotaciones PDF, CMAP, PIC y VRML.Ver figura 9.3.

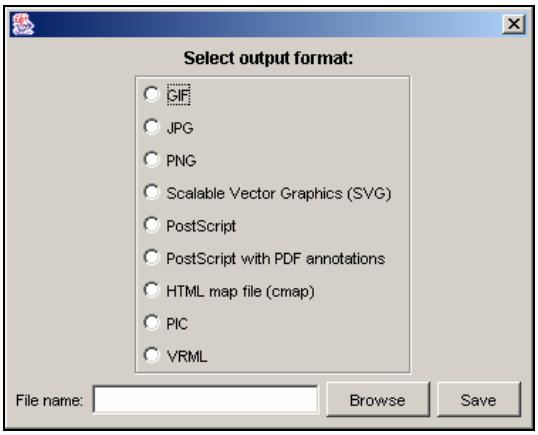


Figura 9.3 Elección del formato gráfico de salida

El último botón, etiquetado con el texto "Check" realiza una comprobación de la ontología utilizando para ello el razonador RACER. Esta comprobación nos permite probar si todos los conceptos son satisfechos y todas las instancias son consistentes.

9.3.1.2 Generación de gráficos

Para visualizar gráficamente una ontología descrita en DAML+ OIL u OWL, se dispone de un área de dibujo que se ha implementado haciendo uso de las librerías *Batik* [BATIK04]. Estas librerías permiten la generación, visualización y edición de gráficos SVG desde un programa Java. En este caso concreto, únicamente se hace uso del módulo encargado de la visualización de gráficos. Existe una clase llamada JSVGCanvas que representa el área de dibujo, así como una serie de manejadores de eventos que se pueden asociar a esta clase y que permiten realizar acciones al cargar los datos del documento SVG, al crear el árbol DOM del fichero fuente o al realizar el dibujado de la imagen. También se pueden asociar los manejadores de eventos comunes como eventos de ratón o de teclado. En este caso, se ha asociado una clase que gestiona el manejo del ratón para detectar el pulsado sobre elementos del gráfico.

Para generar el código en SVG a partir del código DAML u OWL, se ha utilizado una librería de gráficos llamada *Graphviz* [Graphviz04]. Este programa genera código en diferentes formatos, como SVG, JPG, PNG o VRML entre otros, a partir de un lenguaje propio de descripción de grafos. Se ha utilizado este programa en lugar de realizar un parser de DAML + OIL a SVG directamente principalmente por dos motivos:

- La herramienta Graphviz permite generar múltiples formatos a partir de un lenguaje muy sencillo y básico de descripción de grafos, lo que hace posible exportar fácilmente el gráfico.
- Graphviz se encarga de realizar ciertas tareas en el dibujado que no son nada triviales, como es la disposición espacial de los distintos elementos de que consta la escena. En su forma más sencilla, el usuario tan solo se encarga de describir los elementos de los que consta el grafo y sus conexiones. Graphviz se encarga de realizar los demás cálculos geométricos.

El esquema dado en la figura 9.4 muestra los pasos que sigue la aplicación en la generación de gráficos:

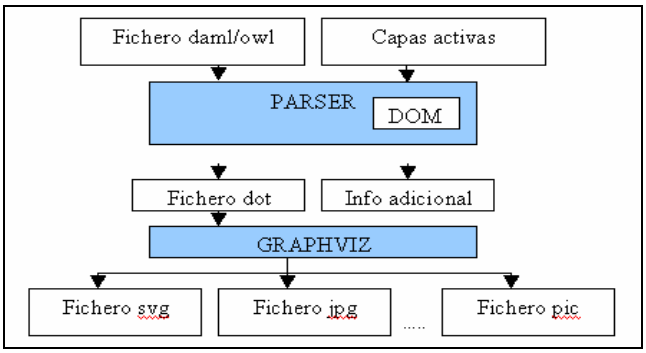


Figura 9.4 Pasos en la generación de gráficos

Tras un proceso de traducción, el fichero OWL/ DAML +OIL se convierte mediante un intérprete en DOT. En este proceso, se elimina información innecesaria que el usuario no desea visualizar (capas), a la vez que se genera información adicional acerca de los elementos del gráfico. Esta información se almacena en memoria para su uso posterior, cuando el usuario pulse sobre dichos elementos. El fichero en formato DOT es utilizado por Graphviz para generar el código en SVG, el cual hace posible su visualización en el área de dibujado SVG de BATIK. Por otra parte el código SVG nos permite exportar el fichero a otros formatos gráficos.

El algoritmo que visualiza una ontología a partir de una URL es el siguiente:

Abrir socket con URL
Descargar archivo y almacenarlo localmente como Temp.daml o Temp.owl
Leer fichero Temp y cargarlo en una estructura de árbol DOM
Recorrer el árbol DOM para generar la estructura de información adicional y el fichero dot con partes que se van a visualizar
Generar fichero SVG a partir del fichero DOT utilizando la utilidad DOT.EXE del paquete Graphviz
Convertir el fichero SVG en un árbol DOM
Cargar en el JSVGCanvas el árbol DOM del gráfico SVG
Para cada clase dibujada, registrar los listeners para los eventos onMouseOver, onMouseOut y onMouseClick

A continuación podemos ver un ejemplo de generación dinámica de gráficos SVG a partir del código de ejemplo dado por la coalición DARPA en [DAML01].

Tras pasar por el parser, el código más sencillo que se puede generar es el siguiente:

```
digraph G{
      size="7,7";
      rankdir=LR;
      node[fontsize=11,shape=box,style=filled,fillcolor=lightyellow];
      "Animal" -> "Masculino";
      "Animal" -> "Femenino";
      "Persona" -> "Hombre";
      "Masculino" -> "Hombre";
      "Persona" -> "Mujer";
      "Femenino" -> "Mujer";
      "Animal" -> "Persona";
      "Animal" [shape=octagon];
      "Persona" [shape=octagon];
}
```

En el código fuente generado lo primero que se hace (líneas 2 y 3) es indicar el tamaño total del gráfico y el sentido del dibujo, en este caso, de izquierda a derecha. En la línea 4 se establecen las características de los nodos, como la forma, el color de relleno y el tamaño de las fuentes. A partir de la línea 5, se describen los nodos y las relaciones entre ellos. En las líneas 12 y 13 se indica que los nodos *animal* y *persona* tienen una forma distinta a la genérica, en este caso, adoptan una forma de octógono. Ver figura 9.5.

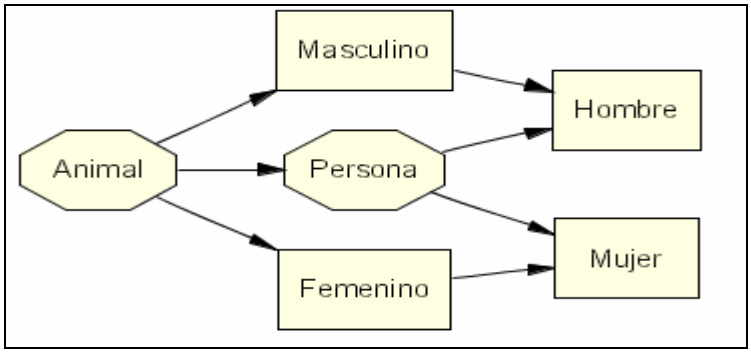


Figura 9.5 Visualización gráfica de los nodos

Si activásemos, por ejemplo, la visualización de las propiedades objeto (*objectProperty)*, el código DOT generado sería más complejo y al final obtendríamos el gráfico mostrado en la figura 9.6:

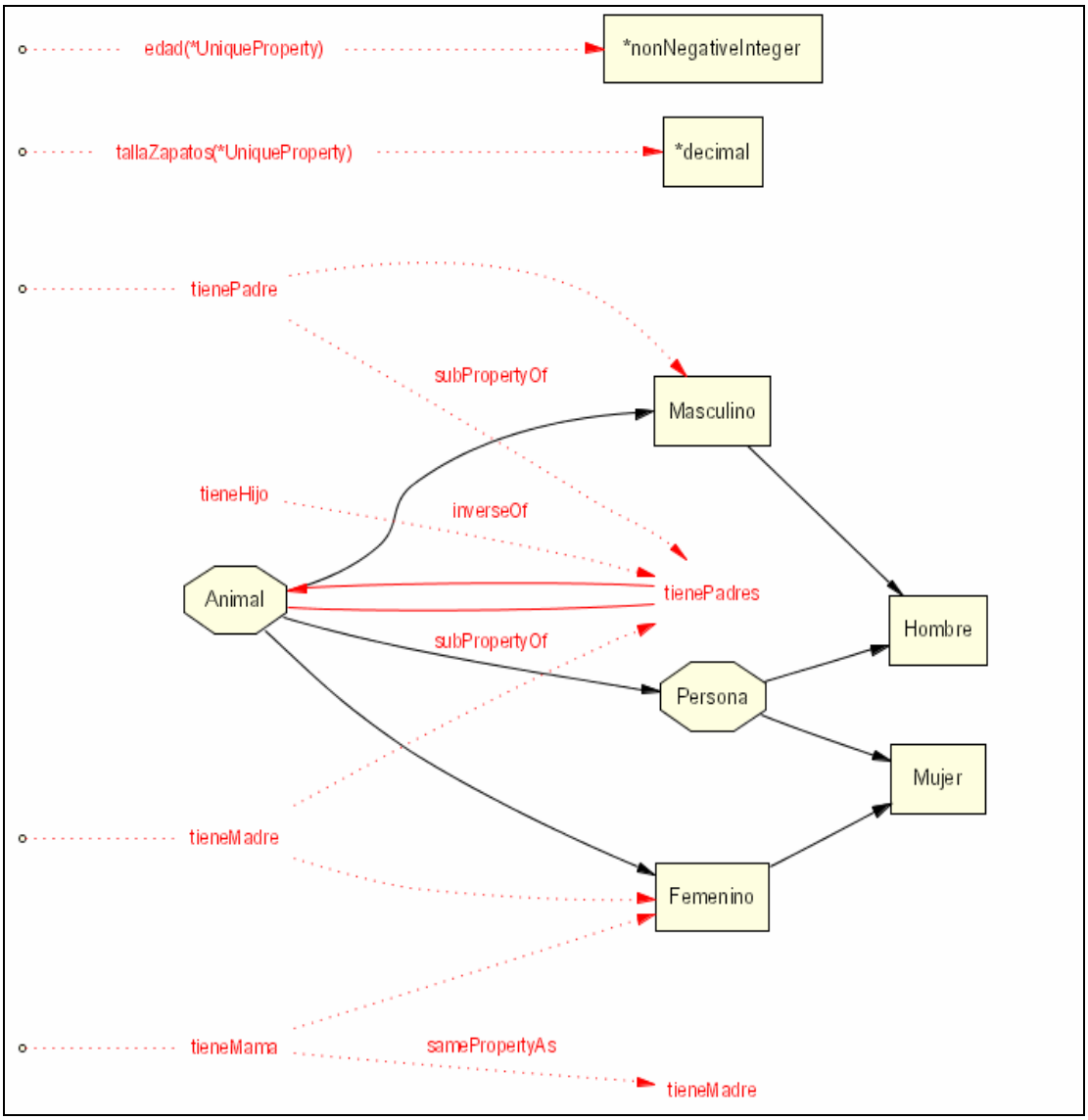


Figura 9.6 Visualización de la ontología ejemplo, con propiedades objeto activadas

9.3.1.2.1 Simbología utilizada en la generación de gráficos e información contextual.

Tal y como se puede apreciar en la figura 9.6 se han utilizado diferentes iconos para representar los elementos.

Clases

- Una clase está representada por un cuadrado de color amarillo: 

- En el caso que la clase tenga restricciones asociadas, tendrá una forma octogonal: 

- Si la clase tiene asociada una condición necesaria y suficiente, tendrá forma octogonal y color rojo claro: 

Instancias

Se representan como una clase rodeada de un doble borde rojo: 

Relaciones

Podemos observar en la tabla 9.1 la representación de cada tipo de relación.

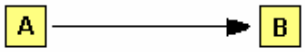
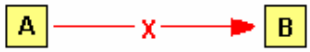
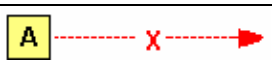
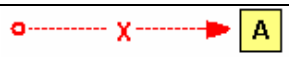
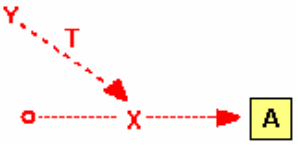
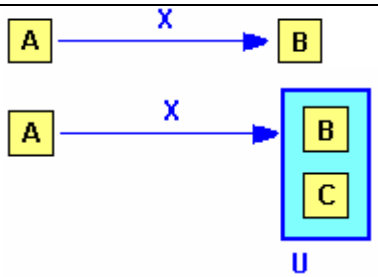
B Subclase de A Se representa con una flecha negra desde A hacia B		subclassOf
X es una propiedad con dominio A y rango B Se representa con una flecha roja desde A hasta B		ObjectProperty, domain, range
X es una propiedad con solo dominio en A Flecha roja punteada que sale de A		ObjectProperty, domain
X es una propiedad con solo rango en A Flecha roja punteada que llega a A		ObjectProperty, range
La propiedad Y tiene una relación T con la propiedad X Flecha roja punteada de Y a X		inverseOf
Combinaciones de igualdad y boleanos. • Flecha azul. • Si implica a varias clases, estas están agrupadas dentro de un cuadrado azul.		sameClassAs, equivalentClass, equivalentTo, sameAs, disjointWith, complementOf, intersectionOf, disjointUnionOf, unionOf

Tabla 9.1: Representación de las distintas relaciones de la ontología.

Hay que tener en cuenta que se ha asociado a cada una de las clases representadas, eventos de ratón de tipo *onMouseOver* que permiten visualizar la información que aparece comentada mediante las etiquetas <comment> en el código de la ontología. Además, si las clases están representadas con forma octogonal están también asociadas a eventos de tipo *onMouseClick,* para obtener toda aquella información relacionada con la clase, que no se está dando explícitamente en el gráfico.

Figura 9.7 Visualización gráfica de una ontología

En la figura 9.7 y centrándonos en la clase "Traffic_Roadway" se puede obtener diferente información contextual dependiendo del tipo de evento:

- Al pasar el ratón por encima de la clase, aparecerá el texto *"Class that represents all the Traffic_Road Services"* en el cuadro de información.
- Al pulsar sobre la clase se abre un cuadro de información con sus restricciones (figura 9.8).

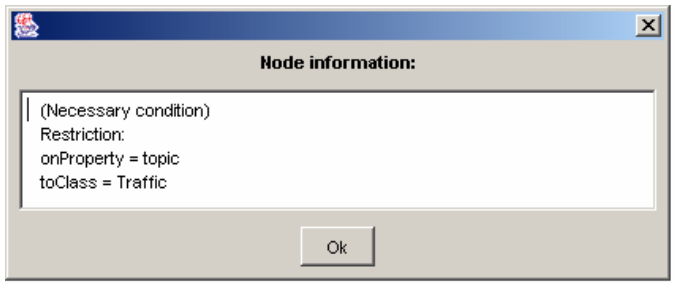


Figura 9.8 Información de un nodo cualquiera

9.3.1.3 Comprobación de la ontología

La aplicación permite chequear la ontología en busca de incorrecciones o incoherencias. Para ello, hace uso de RACER como motor de inferencia incorporado a la interfaz. Al pulsar el botón de check con una ontología cargada, aparece una ventana de diálogo con varias opciones.Ver figura 9.9.

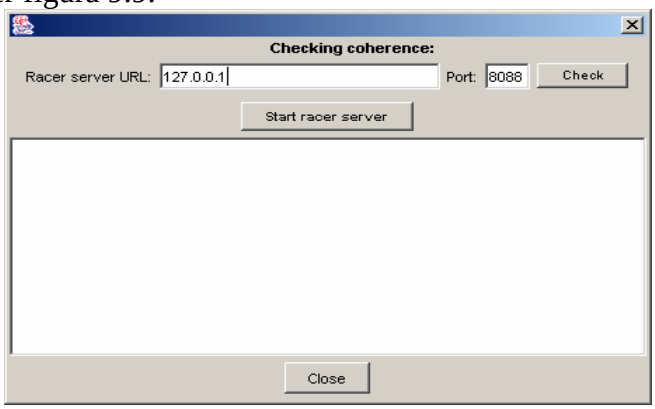


Figura 9.9 Localización del motor de inferencia Racer

RACER funciona como una aplicación cliente-servidor, donde el cliente envía una consulta mediante HTTP al servidor, que le devuelve la respuesta. Para ello, el usuario debe introducir la dirección del servidor y el número de puerto. En la versión actual de la interfaz, solo está permitido que el servidor RACER esté lanzado localmente, ya que necesita acceder al archivo OWL o DAML + OIL almacenado del mismo modo. En el caso en que la ontología se haya cargado mediante una URL, también es necesario que RACER esté lanzado en la misma máquina, ya que donde realmente está accediendo es a una copia local.

Al pulsar el botón de *check*, se comprueba la corrección de la ontología (sistema terminológico e instancias) mediante consultas propias de RACER, y si todo es correcto, aparecerán los siguientes mensajes:

Ok - Concepts are satisfiable
Ok - Individuals are consistent

En caso contrario, aparecerá una descripción del error encontrado.

9.3.2 Buscador de servicios

La función principal de la herramienta descrita es la búsqueda de servicios determinados a partir de la generación de un perfil. Para ello, se puede utilizar la herramienta gráfica descrita en el punto anterior, aunque no es imprescindible. En el capítulo 10, veremos como ésta se convierte en una herramienta útil, para interaccionar con el resto del sistema a través del agente cliente. Este agente cliente recibe el URI de la descripción de servicio generada por OntoService, pudiendo ser éste último quien lance la ejecución del agente. Lo peculiar del interfaz, además del hecho de poder integrarse como un elemento más en el sistema de búsqueda de servicios, es el hecho de hacer uso de perfiles de usuario, los cuales permiten optimizar el proceso de búsqueda. A continuación, describiremos en que consiste este procedimiento y cuales son los elementos que hacen posible la implementación de los proxies necesarios.

9.3.2.1 Optimización de la búsqueda mediante el uso de un proxy

En el proceso de búsqueda de un servicio, una de las fases más costosas es el proceso de "match machine", en el cual se trata de localizar los servicios realizando un emparejamiento del perfil generado con los perfiles suministrados por los proveedores de servicios. El uso de una caché que almacene las últimas consultas realizadas permite devolver con rapidez los servicios buscados sin necesidad de realizar el proceso de búsqueda, incrementando la velocidad del sistema, ya que probablemente los usuarios buscarán un reducido número de tipos de servicios. Para ello, se ha diseñado una arquitectura de almacenamiento remoto de perfiles (figura 9.10) que permite que un usuario se pueda conectar desde cualquier lugar al servidor de perfiles y cargar las últimas búsquedas realizadas.

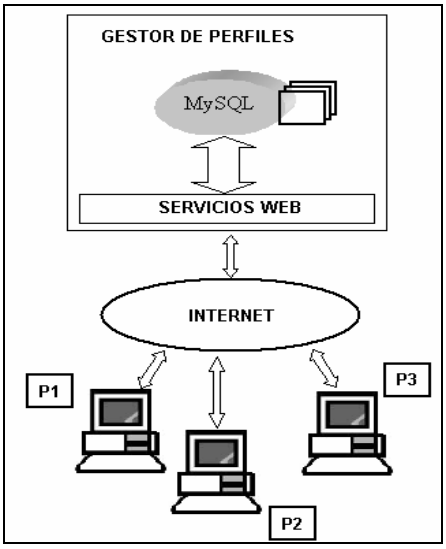


Figura 9.10 Arquitectura del gestor de perfiles

Cuando un usuario inicia la aplicación realiza tres tipos de acciones:

1. Puede darse de alta en el servidor abriendo una cuenta nueva. Para ello, introducirá el nombre de usuario y la contraseña deseada y pulsará en botón de "create new account". El usuario será informado del resultado de la operación en una ventana. Los posibles mensajes se pueden observar en la tabla 9.2:

User exists	Ya existe un usuario con el identificador introducido. Es necesario, por lo tanto, cambiar de nombre de usuario.
Server error. Bad URL?	No se ha encontrado un servidor en la dirección indicada. O bien la dirección es errónea o el servidor está caído.
Trusted server error. Please contact server administrator	Se ha producido un error en el servidor y no se ha podido completar la operación. Este tipo de errores suele aparecer por ejemplo si no se ha podido acceder a los datos porque no se ha lanzado el demonio del gestor de bases de datos.
New user created	La operación ha sido realizada con éxito.

Tabla 9.2: Mensajes posibles en el alta de un usuario.

2. Acceso a la aplicación como usuario registrado. Para ello, la opción seleccionada debe ser "Login". La aplicación accederá al servidor de perfiles y cargará localmente el perfil devuelto por el servidor. A partir de ese momento, la búsqueda de perfiles se realizará en una primera instancia sobre la caché local y en caso de no encontrarse, se lanzará la petición al agente encargado de realizar el "match machine". Los posibles mensajes que pueden aparecer son descritos en la tabla 9.3:

Server error. Bad URL?	No se ha encontrado un servidor en la dirección indicada. O bien la dirección es errónea o el servidor está caído.
Trusted server error. Please contact server administrator	Se ha producido un error en el servidor y no se ha podido completar la operación. Este tipo de errores suele aparecer por ejemplo si no se ha podido acceder a los datos porque no se ha lanzado el demonio del gestor de bases de datos.
Bad user/password	El nombre de usuario y/o la contraseña no son correctos

Tabla 9.3: Mensajes posibles en el acceso como usuario registrado.

Una vez confirmado el usuario y la contraseña (figura 9.11), el nombre de usuario aparece en la parte superior izquierda de la interfaz.

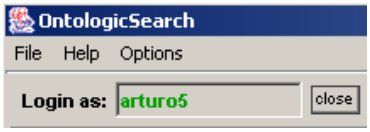


Figura 9.11 Usuario que accede

3. Acceso a la aplicación como usuario anónimo. Escogiendo esta opción, la función de caché de perfiles queda deshabilitada y por tanto, no se pueden utilizar sus funciones. Cualquier consulta que se realice tiene que ser gestionada por el agente encargado de la búsqueda de perfiles.

9.3.2.1.1 Implementación del servidor de perfiles

Para implementar el servidor de perfiles, se definieron previamente cuáles eran las funciones que debía realizar:

- Se debían poder crear nuevas cuentas de usuario.

- Tenía que existir un mecanismo para recuperar los datos de un usuario determinado, así como también actualizarlos.
- La comunicación entre los clientes y el servidor debía seguir protocolos estándar.

Para implementar estas necesidades, se han implementado tres servlets, cada uno de los cuales realiza una tarea. El intercambio de información entre clientes y servidor se realiza mediante el protocolo SOAP.

- Conseguir perfil: Devuelve los perfiles del usuario en la base de datos
- Insertar perfil: Inserta la nueva información de perfiles en la base de datos.
- Nuevo usuario: Crea un nuevo usuario en la base de datos del servidor.

En la figura 9.12 se puede observar el intercambio de mensajes que se realiza en el proceso de conseguir un perfil.

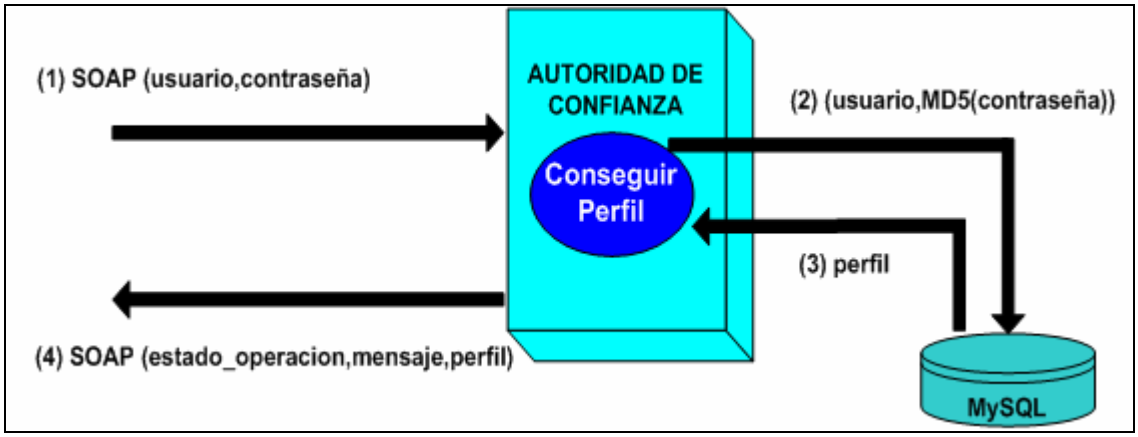


Figura 9.12 Implementación del servidor de perfiles: Conseguir un perfil

El servlet *conseguir perfil* devuelve el perfil almacenado enviando el usuario y la contraseña correspondiente.

Mensajes codificados devueltos:

- (0) Éxito en la operación.
- (1) El usuario/password no son válidos.
- (2) Problemas accediendo a la base de datos.
- (3) Problemas de encriptación del password.
- (4) El usuario no tiene ningún perfil asociado

Mensaje SOAP enviado por el cliente con su identificación:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <SOAP-ENV:Header/>
        <SOAP-ENV:Body>
                <parameters>
                        <username xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xsi:type="xsd:string">javier</username>
                        <password xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
                        xsi:type="xsd:string">art_pwd</password>
                </parameters>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Mensaje SOAP recibido por el cliente con perfil embebido:

```
------=_Part_0_17103608.1083144426484

Content-Type: text/xml

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><SOAP-ENV:Header/>
<SOAP-ENV:Body><return><state xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xsd:string">0</state><message xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:type="xsd:string">Success!</message></return></SOAP-ENV:Body>
</SOAP-ENV:Envelope>

------=_Part_0_17103608.1083144426484
Content-Type: text/plain
Content-Id: profile

<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"

 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

…

<daml:Ontology><daml:versionInfo>24-11-2004</daml:versionInfo><rdfs:comment>Profile
description realizado en DAML-S por OntoService</rdfs:comment></daml:Ontology>

<profileHierarchy:Pre_trip_Information rdf:ID="Profile_Incidencias_Trafico_Service">


…

</profileHierarchy:Pre_trip_Information >
</rdf:RDF>
```

Veamos ahora el intercambio de información en el proceso de insertar un perfil. Figura 9.13.

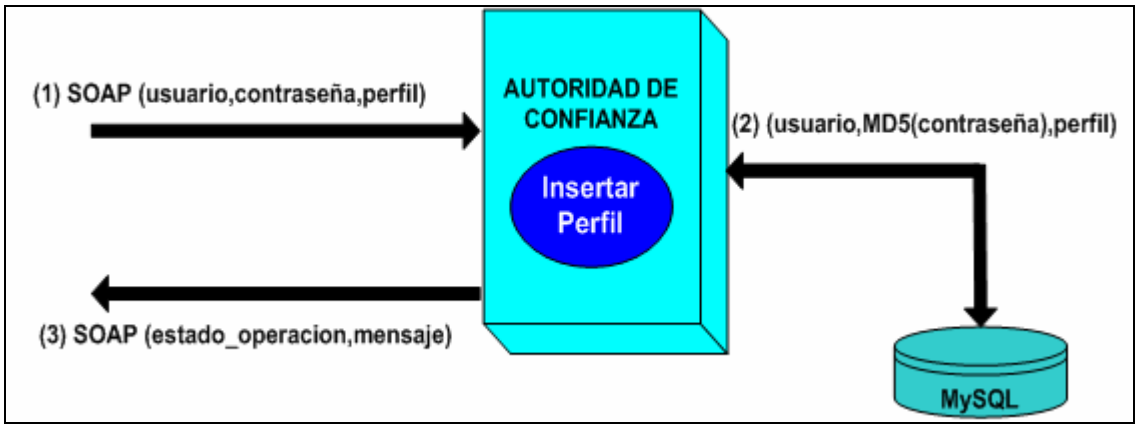


Figura 9.13 Implementación del servidor de perfiles: Insertar un perfil

El servlet *insertar perfil* se encarga de actualizar o insertar un nuevo perfil en la base de datos MySQL. Si todavía no existe un perfil para el usuario dado, introduce uno nuevo. Si ya existía un usuario con perfil, lo actualiza. Acepta ficheros de hasta 16MB (tipo *mediumblob*). No se permite que un usuario tenga varias filas en la tabla con diferentes perfiles, sino que un solo fichero debe contener los elementos *<rdf>* que se estimen necesarios para almacenar más de un perfil.
El mensaje consta de dos partes. La primera contiene el nombre del usuario y la contraseña. La segunda parte, contiene el archivo XML con el perfil. De esta forma, el perfil se envía como fichero adjunto.

Los mensajes codificados devueltos se corresponden con los numerales de 0 a 3 de *conseguir perfil*.

Por último, en la figura 9.14 observaremos el proceso de crear una cuenta de usuario.

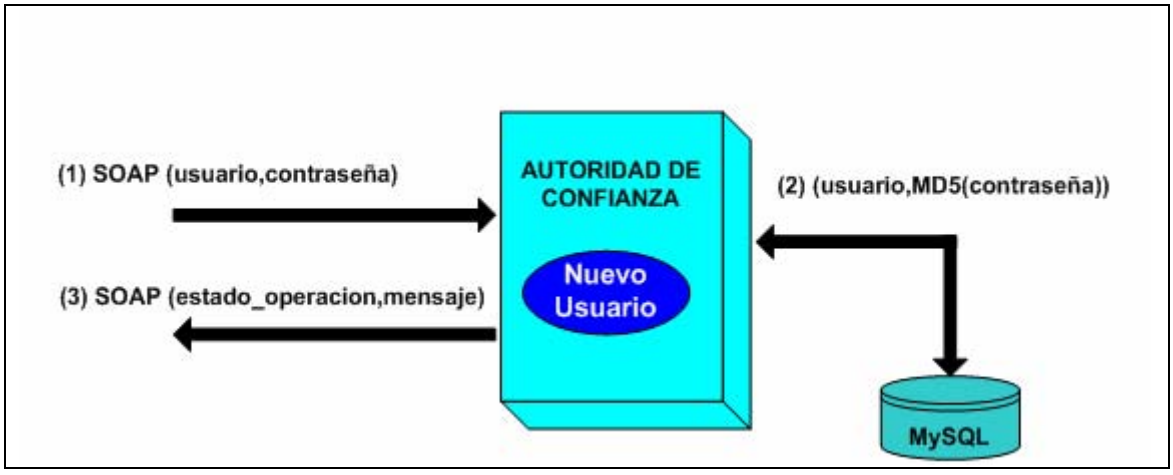


Figura 9.14 Implementación del servidor de perfiles: Crear una cuenta

Mensajes SOAP devueltos por el servicio *Nuevo Usuario*

- (1) El proceso se ha llevado con éxito.

- (0) El usuario ya existe.
- (3) Errores con la base de datos: "No se encuentra la base de datos" y "no es posible conectar con ella".
- (4) Problemas de encriptación del password.

9.3.2.2 Generación del perfil de búsqueda

La búsqueda de servicios se puede realizar con el apoyo de una herramienta de visualización o sin ella, dependiendo si el usuario quiere rellenar los campos introduciendo el texto o seleccionándolo del gráfico directamente.

El número de parámetros que se pueden configurar en la búsqueda de perfiles se puede apreciar en la tabla 9.4:

Campo	**Obligatorio**	**Seleccionable gráficamente**	**Descripción**
Keywords	NO	NO	Palabras claves que deseamos introducir en la búsqueda (inclusión en <profile:textdescription>)
Service Name	NO	NO	Nombre del servicio, en caso de conocerlo (inclusión en <profile:serviceName>)
Inputs	NO	SI	Parámetros de entrada del servicio
Outputs	SI	SI	Parámetros devueltos por el servicio
Geographic	NO	SI	Parámetro relativo a la localización del servicio
Quality	NO	SI	Parámetro relativo a la calidad del servicio
Actor Name	NO	NO	Proveedor del servicio buscado

Tabla 9.4: Características de los parámetros en la búsqueda.

El aspecto de la aplicación lo podemos apreciar en la figura 9.15.
El menú de la parte superior permite realizar las acciones expuestas en la tabla 9.5:

File -> Exit	Sale de la aplicación
Help -> About	Muestra información acerca de los autores de la aplicación
Options -> Logout	Desconecta el usuario actual y permite entrar con otro usuario
Options -> View profiles in cache	Muestra la información de los perfiles que tiene almacenados en la caché de perfiles, incluyendo los servicios encontrados en anteriores búsquedas. Esta opción es solo válida para usuarios registrados, ya que el usuario anónimo no contempla la gestión de caché de perfiles.

Tabla 9.5: Acciones posibles mediante el menú de OntoService

En la parte izquierda podemos encontrar una interfaz con estructura de árbol donde se muestran tres tipos de clasificaciones de jerarquías de conceptos: NAICS, UNSPSC y

TRAFFIC. Navegando por la estructura de árbol en cualquiera de las tres clasificaciones podemos llegar a un nodo hoja del árbol, donde tras pulsar el botón etiquetado con una flecha a la derecha, se visualizará la ontología de conceptos asociada al nodo del árbol (en caso de existir).

Como se observa en la tabla 9.4, algunos campos como 'inputs' y 'outputs' permiten ser completados mediante la interfaz gráfica. Para ello, existen dos cuadros que permiten indicar que se desea rellenar el campo correspondiente. De esta forma, al pulsar sobre una clase en el visualizador de ontologías, es posible distinguir si lo que se desea es ver información de una clase o seleccionarla como input u output.

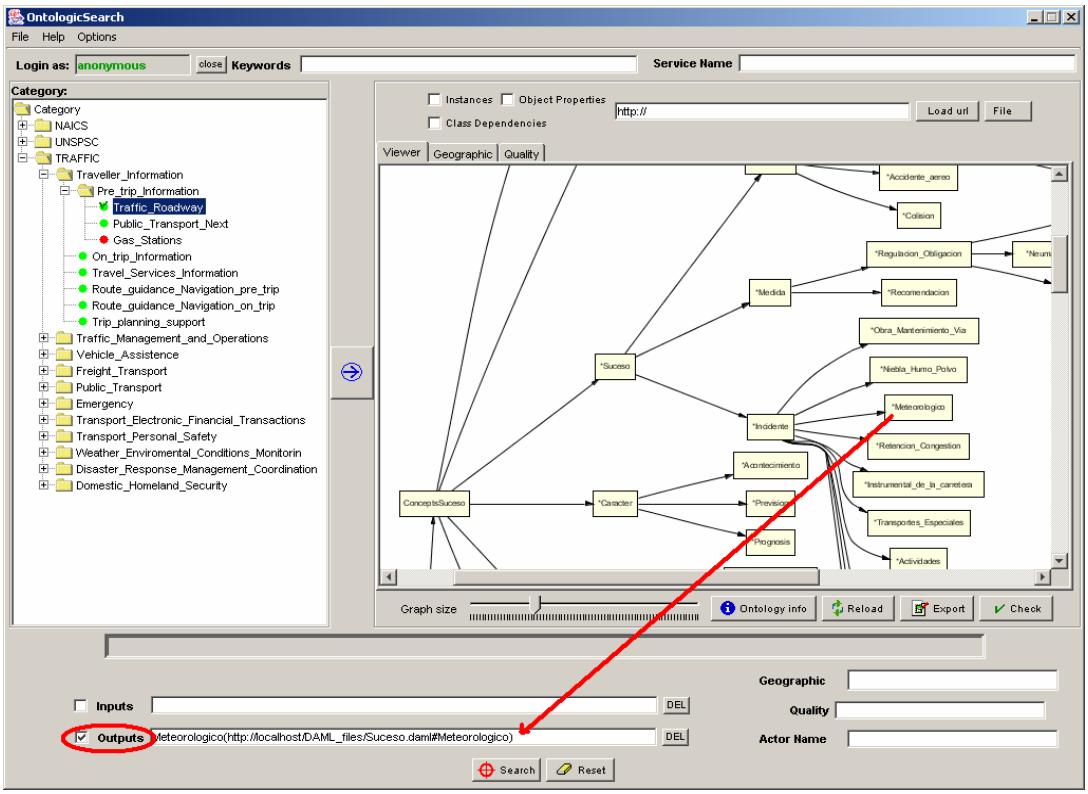


Figura 9.15 Elección de parámetros funcionales de salida

Tras pulsar en la clase que representa un concepto determinado, se copia en el campo seleccionado el nombre de la clase, incluyendo su espacio de nombres. Los campos que se rellenan en el perfil son *<profile:parameterName>* con el nombre de la clase seleccionada y *<profile:restrictedTo rdf:resource=X >* siendo X la URI del concepto seleccionado. Los botones etiquetados como 'DEL' permiten borrar el campo.

Los campos *Geographic* y *Quality* permiten indicar el alcance geográfico y la calidad del servicio respectivamente. Para ello, de la misma forma que los parámetros de tipo funcional, se pueden seleccionar gráficamente pulsando sobre las pestañas correspondientes. Tras este primer paso, se visualizarán las ontologías de geografía y la subontología de conceptos relativa a QoS, tras lo cual, pulsando sobre las clases, los conceptos seleccionados se cargan en los campos requeridos.

Para comenzar el proceso de búsqueda, se debe pulsar sobre el botón de '*Search*', el cual realiza dos funciones:

1. Busca en la caché si existe algún perfil similar al actual. En caso que sea así, informa al usuario que anteriormente se realizó una búsqueda similar y se encontraron determinados servicios. Si el usuario lo desea, puede utilizar alguno de los servicios encontrados anteriormente y de esta manera no tener que relanzar una búsqueda similar. El criterio que se ha seguido para comparar los perfiles en memoria con el buscado es comparar los *inputs* y *outputs*. Si coinciden todos, se considera que el servicio anteriormente encontrado es de interés para el usuario.

2. Envía el perfil generado al agente encargado de realizar la búsqueda del servicio.

Perfil básico de servicio generado con OntoService

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:rdf= "http://www.w3.org/1999/02/22-rdf-syntax-ns#"

 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
 xmlns:service="http://www.daml.org/services/daml-s/0.9/Service.daml#"
 xmlns:process="http://www.daml.org/services/daml-s/0.9/Process.daml#"
 xmlns:grounding="http://www.daml.org/services/daml-s/0.9/Grounding.daml#"
 xmlns:profileHierarchy="http://localhost/DAML_files/CategoriaServicios.daml#"
 xmlns:xsd="http://www.w3.org/2000/10/XMLSchema.xsd#"
 xmlns:traficoService="http://localhost/DAML_files/TraficoService.daml#"
 xmlns:traficoProcess=http://localhost/DAML_files/TraficoProcess.daml#
 xmlns:traficoGrounding=http://localhost/DAML_files/TraficoGrounding.daml#
 xmlns:country="http://localhost/DAML_files/Geografia.daml#"
 xmlns:concepts="http://localhost/DAML_files/Concepts.daml#"
 xmlns:time="http://localhost/DAML_files/Time.daml#"
 xmlns:profile= "http://localhost/DAML_files/MiProfile.daml#"
 xmlns="http://http://localhost/DAML_files/TraficoProfile.daml#">

<daml:Ontology>
<daml:versionInfo>      24-11-2004    </daml:versionInfo>
   <rdfs:comment>
         Profile description realizado en DAML-S por el Visualizador de Ontologías
   </rdfs:comment>
</daml:Ontology>

<profileHierarchy:Pre_trip_Information rdf:ID="Profile_Incidencias_Trafico_Service">
<profile:serviceName></profile:serviceName>
<profile:textDescription></profile:textDescription>

<profile:serviceParameter>
   <profile:GeographicRadius rdf:ID="GeographicRadius">
      <profile:serviceParameterName>Geographic Radius</profile:serviceParameterName>
      <profile:sParameter
rdf:resource="http://localhost/DAML_files/Geografia.daml#Comunidad_Valenciana"/>
   </profile:GeographicRadius>
</profile:serviceParameter>
<profile:output>
   <profile:parameterDescription rdf:ID="Meteorologico">
      <profile:parameterName>Meteorologico</profile:parameterName>
      <profile:restrictedTo
rdf:resource="http://localhost/DAML_files/Sucesos.daml#Meteorologico"/>
   </profile:parameterDescription>
</profile:output>
</profileHierarchy:Pre_trip_Information>
</rdf:RDF>
```

9.4 CONCLUSIONES

En este capítulo se ha presentado una herramienta que permite combinar la visualización/verificación de ontologías de conceptos con la creación de perfiles de SWS necesarios para la búsqueda éstos, lo cual representa una ventaja frente a las herramientas concebidas hasta la fecha, basadas generalmente en la traducción de ficheros WSDL. Nuestra herramienta permite crear desde cero, un perfil de búsqueda a través de la interacción del cliente con el entorno gráfico de la base de conocimiento.

Otra diferencia con respecto al resto de entornos, es el hecho de que la edición de perfiles semánticos está encaminada a la búsqueda de servicios y por lo tanto, orientada al usuario, no al proveedor. Por este motivo, únicamente se genera el fichero de *profile* a partir de una plantilla, siendo innecesarios los ficheros de *process* y *grounding*.

La doble funcionalidad de la herramienta logra resultados positivos, principalmente por dos razones de peso: la primera de ellas es que es posible ajustar, modificar, actualizar y verificar cualquiera de las ontologías usada como soporte en la construcción de los perfiles de servicio, lo que permitirá realizar búsquedas más exactas. La segunda de las razones es la posibilidad de buscar servicios mediante la interacción con el sistema prototipo, lo que permite simplificar el proceso de búsqueda y asistir al cliente en la edición de tediosos perfiles que describan las capacidades del servicio requerido.

Los dos requisitos exigidos en cualquier interfaz gráfica se han cumplido: Ser intuitiva y sencilla. Los clientes no tienen porque saber como describir un servicio mediante DAMLS/OWLS Profile, de que parámetros consta, y cuales de ellos se consideran imprescindibles. Tampoco debe conocer cómo están formadas las ontologías y cuales son las URI de cada uno de los recursos que desea emplear. Por otra parte, la interfaz muestra únicamente aquello que se considera necesario para la realización del cometido, que no es otro que la construcción de un perfil o la simple visualización y/o comprobación de una ontología. Se deja muy poco margen de error en la interacción con el usuario, mediante información clara y concisa, por lo que un usuario será capaz de encontrar un servicio web a través de la descripción semántica de sus capacidades, sin tener un ápice de conocimiento sobre el lenguaje usado para realizar está descripción de manera formal.

Se han descrito las principales características y componentes de OntoService, tomando como referencia su dicotomía funcional: Visualizador/verificador de ontologías y buscador de servicios.

Atendiendo a su primera funcionalidad, destaca el hecho de su integración con RACER, para comprobar la consistencia de las ontologías, y en la función de buscador, uno de los aspectos que distingue la herramienta presentada de las existentes es la posibilidad de utilizar un gestor de cachés de perfiles de búsqueda. La utilización de la caché de perfiles, la cual almacena su información de forma remota, permite a los usuarios acceder a su información personal desde cualquier lugar desde el que se conecten, permitiendo acelerar el proceso de búsqueda de perfiles para consultas similares, minimizando la carga en la red y el procesamiento en el agente que realiza la búsqueda.

En lo que concierne a la exportación, esta herramienta tiene la capacidad de generar información en diferentes formatos, dentro de los que se encuentra el formato SVG, propuesto por el W3C para representación de información gráfica basada en XML.

La herramienta ha sido desarrollada en lenguaje Java, lo cual posibilita su uso en diversas plataformas. Ha sido utilizada para la visualización de ontologías y descripción semántica de servicios en sistemas inteligentes de información de tráfico. El resultado generado por la herramienta ha sido usado dentro del sistema multiagentes de emparejamiento de SWS desarrollado en este trabajo de investigación. Aunque no forma parte de la plataforma de agentes implementada tal y como veremos en el capítulo 10, sin embargo se convierte en una herramienta útil, para interaccionar con el resto del sistema a través del agente cliente, mediante el paso de la URI que identifica la descripción de servicio generada por OntoService.

CAPÍTULO 10

VALIDACIÓN DE PROPUESTAS: ARQUITECTURA DE INTEGRACIÓN

10.1 RESUMEN

En este capítulo se expone en qué ha consistido la creación de una **arquitectura de integración de SWS de información sobre tráfico vial**. El sistema en general es entendido como un sistema multiagente orientado a facilitar, asistir, y optimizar los procesos de anuncio, descubrimiento e invocación de SW relacionados con el área de tráfico vial.

Primeramente se describe en que radica el emparejador de SW denominado ESWIT, atendiendo a sus características, así como los diferentes actores que lo componen. Se exponen diversos trabajos relacionados y su relación más estrecha con alguno de ellos. A continuación se detallan las distintas fases en las que consiste el funcionamiento global del sistema, a través de las interacciones entre los agentes que lo forman, bajo diferentes situaciones o escenarios.

Posteriormente, se detalla el sistema como una arquitectura en capas y se especifica cada una de ellas, haciendo hincapié en aquellos elementos que no han sido tratados ampliamente en los capítulos previos. Una vez analizadas cada una de estas capas, se muestra en qué radica el despliegue de todos los elementos del sistema, inclusive aquéllos que no forman parte de la plataforma de agentes.

Finalmente, se describen en que han consistido las pruebas realizadas que buscaban la comprobación de la integración del emparejador en la plataforma de agentes y su uso en un caso real, desde la realización de la petición de búsqueda (perfil del servicio) hasta la invocación y ejecución del servicio encontrado.

10.2 EMPAREJADOR DE SERVICIOS WEB DE INFORMACIÓN DE TRÁFICO (ESWIT)

En el ámbito de recuperación de información o búsqueda podemos encontrar diferentes tipos de herramientas como buscadores genéricos (directorios, motores de búsqueda), buscadores especializados en algún dominio, buscadores inteligentes (sitios interesantes para personas con preguntas similares realizadas en lenguaje natural etc.), metabuscadores (integración de resultados de diferentes motores de búsqueda) y por último los agentes inteligentes.

Dentro de este ámbito de búsqueda y haciendo uso de agentes inteligentes como herramienta, están aquellos sistemas como ESWIT, cuya tarea es buscar información a través de SW, mediante el emparejamiento de éstos, haciendo uso de sus capacidades descritas semánticamente.

ESWIT puede ser clasificado como un sistema de recuperación de información constituido por agentes cooperativos como lo son RETSINA (*Reusable Task Structure-based Intelligent Network Agents*) [Syc99] [RET01], InfoSleuth [Bay97], [INF03] e IMPACT (*Interactive Maryland Platform for Agents Collaborating Together*) [Ari00],[IMP03].
De entre los dos tipos de agentes mediadores citados en [Dec96], ESWIT se basa en el uso de agentes *matchmaker* y por tanto el sistema está basado en el modelo de intermediación entre agentes denominado "*matchmaking*", de esta forma, está más en la línea de RETSINA e IMPACT que de InfoSleuth, ya que éste último se caracteriza por el uso de agentes de tipo *broker*. El motivo de la elección de esta aproximación fue liberar a los agentes mediadores de tareas como controlar las transacciones de servicios y obtención de resultados finales, de tal forma que la invocación y ejecución de los servicios quedara fuera de su alcance, con lo cual se mejora considerablemente la función de encontrar los servicios adecuados. A su vez, este tipo de modelo nos permite controlar este proceso de invocación, al ser una tarea realizada no por el agente mediador, sino por el agente cliente, el cual está en contacto a través de la interfaz con el usuario de la aplicación.
Como entorno de desarrollo se utilizó JADE por cumplir lo propuesto por FIPA en su modelo de referencia de administración de agentes [FIPA04], donde se establecen los elementos básicos de los que debe constar un sistema multiagente. La plataforma implementada permite el desarrollo del ciclo completo de anuncio, petición, descubrimiento, invocación y baja de SW dentro de ella. Actualmente existen otros trabajos relacionados que hacen uso de JADE como plataforma [Lei03].

El sistema en general cumple una serie de características que a continuación se exponen:

- Los SW de tráfico han sido descritos semánticamente empleando los lenguajes de ontologías diseñados para este objetivo (remitirse al capítulo 8).
- Han sido necesarias diferentes ontologías para describir los diferentes dominios de conceptos que se emplean en el sistema (capítulo 6).
- El sistema tiene un componente encargado de realizar el emparejamiento de la petición de servicio realizada por un posible cliente con todos los servicios disponibles. Las operaciones de búsqueda basadas en las descripciones semánticas de los servicios ofertados y los requeridos estarían centralizadas en
- él. Este elemento integra el algoritmo expuesto en el capítulo 7 el cual como ya se comentó explota las posibilidades de los valores semánticos de las descripciones de SW.
- El sistema permite que tanto clientes como proveedores aparezcan y desaparezcan de él en función de si ya han recibido el servicio buscado o dejan de estar operativos respectivamente.
- Para facilitar a los clientes la construcción de los perfiles de SW requeridos, se hace uso de la herramienta o interfaz OntoService, expuesta en el capítulo número 9 de la presente memoria.

Siguiendo el modelo de intermediación *matchmaking* se identificaron durante la fase de análisis los siguientes actores, que son los componentes básicos de nuestro sistema:

- ***Cliente o Usuario***: es la persona que usa el sistema.
- ***Agente cliente***: agente con el rol de cliente. Representa al usuario dentro de la plataforma, y proporciona al agente la descripción del servicio que busca.
- ***Proveedor***: empresa, organización o persona que anuncia un servicio en la plataforma mediante un agente proveedor.
- ***Agente proveedor***: agente con el rol de proveedor de servicio web, representa al proveedor dentro de la plataforma.
- ***Agente emparejador***: agente que tiene el rol de *matchmaker*, es el encargado de emparejar en función de su proximidad semántica, descripciones de servicios recibidas de clientes con descripciones de servicios anunciados por los proveedores.
- ***Facilitador de directorio (DF)***: agente incluido en las plataformas que cumplen el modelo FIPA, que aunque en esta especificación es opcional, se utiliza para facilitar las tareas de administración de los agentes. Cumple la función de *páginas amarillas.*
- ***Servicio(s) Web***: servicios web externos a la plataforma y que son representados dentro de ella por agentes proveedores.
- ***Emparejador de servicios***: emparejador de servicios basado en la descripción semántica de sus capacidades, y que será utilizado por el agente emparejador.

Los casos de uso identificados pueden ser observados de forma gráfica en la figura 10.1:

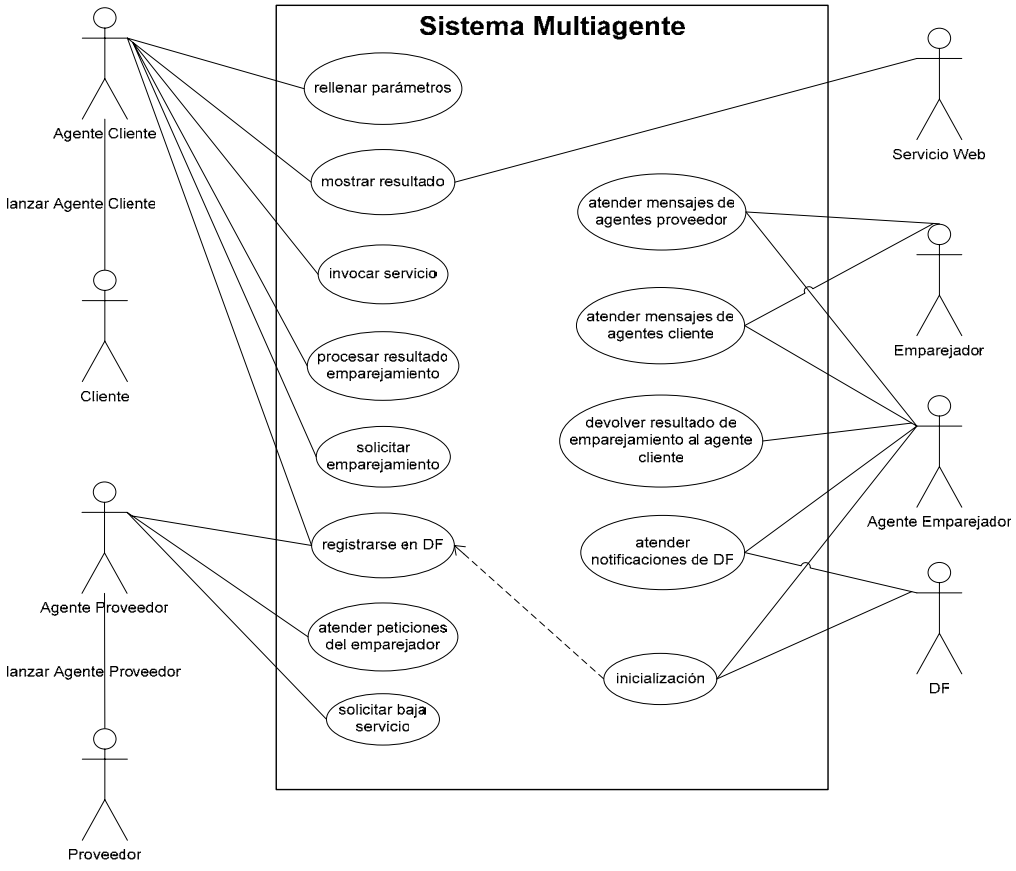


Figura 10.1 Diagrama de casos de uso del sistema multiagente

Teniendo en cuenta los requisitos de nuestro sistema y los casos de uso especificados, se identificaron las principales clases, relaciones y atributos del sistema representado mediante el diagrama AUML de la figura 10.2, en el que por simplificación aparece el paquete sma.Matchmaker como la implementación del algoritmo de emparejamiento propuesto, cuyo diagrama de clases será presentado en un punto posterior de este capítulo.
Las principales clases identificadas fueron las correspondientes al agente cliente, agente emparejador y agente proveedor.

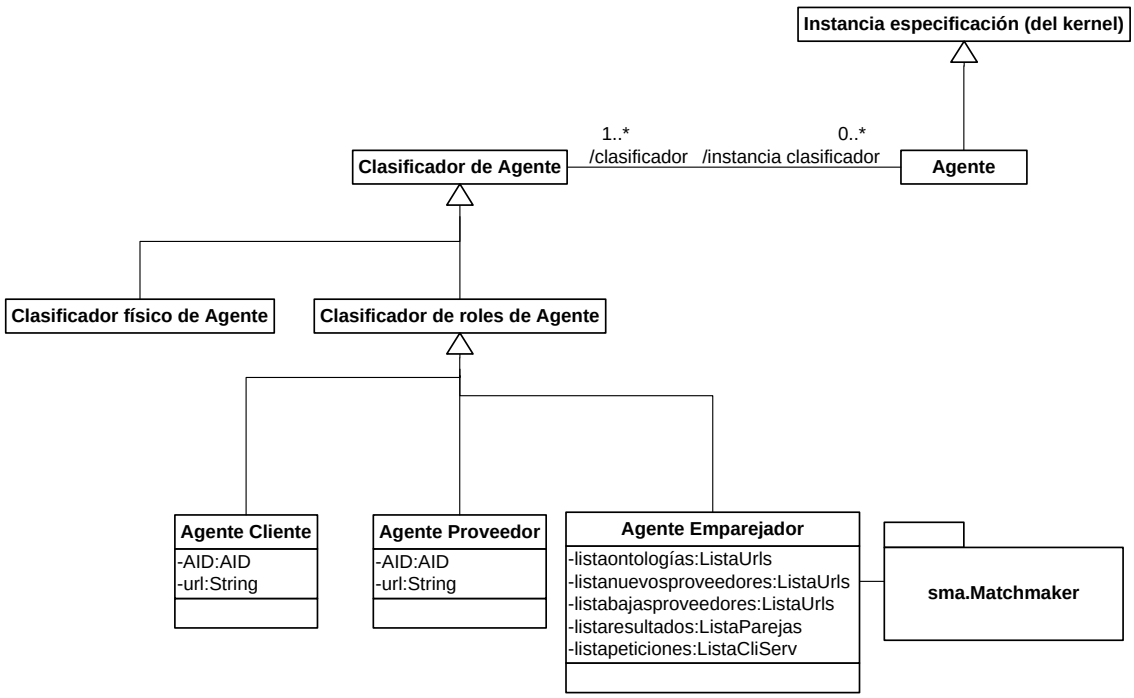


Figura 10.2 Diagrama AUML del sistema multiagente

Los diferentes atributos correspondientes a las clases principales son:

- las URLs de los profiles tanto del agente cliente como del agente proveedor,
- los atributos necesarios para que el agente proveedor pudiese hacer uso del emparejador de servicios,
- la lista en la que el agente emparejador almacena las URLs de las descripciones de los servicios representados por agentes del tipo proveedor que solicitan la baja,
- la lista en la que el agente emparejador almacena las URLs de las descripciones de servicios representados por agentes proveedores que solicitan el alta,
- la lista en la que se almacenan las URLs de las peticiones de emparejamiento recibidas de agentes cliente,
- la lista de las URLs de las ontologías a utilizar,

– y por último, la lista de donde el agente emparejador extrae los resultados obtenidos por el emparejador semántico.

10.2.1 Interacción entre agentes que componen el sistema.

Las fases que resumen el funcionamiento del sistema son las siguientes:

1. Cuando se anuncia un nuevo servicio, su agente proveedor, representante de este servicio dentro de la plataforma, en primer lugar se registra en el DF. A continuación el DF avisa al agente emparejador y éste solicita al nuevo proveedor la URI de la descripción semántica del servicio que ofrece.

2. Al lanzar un usuario un agente cliente, éste le pasa como parámetro la URI de la descripción semántica del servicio que busca en la plataforma. Este agente cliente se pone en contacto con el agente emparejador y le proporciona la URI de la descripción de este servicio. El agente emparejador interroga al razonador y contrasta la descripción del servicio buscado con todas las de los servicios disponibles, decidiendo qué descripción es la que más se ajusta al servicio buscado, e informando posteriormente al agente cliente de la URI de la descripción de este servicio.

3. Una vez el agente cliente conoce la descripción que más se parece semánticamente al servicio que busca, genera a partir de ella, y de forma dinámica, la interfaz gráfica para recoger los datos de entrada necesarios para poder hacer la correcta invocación del servicio que describe.

4. Cuando el usuario rellena el formulario que le pide los datos de entrada del servicio, el agente cliente lo invoca directamente con esos datos y recoge el resultado que devuelve.

5. El agente cliente vuelve a ponerse en contacto con el usuario y le muestra el resultado de la invocación del servicio.

Para hacer posible estas interacciones es necesario que se establezcan comunicaciones entre los distintos agentes que forman parte del sistema. Las principales comunicaciones establecidas entre éstos se detallan en los siguientes diagramas AUML de protocolo, todos ellos utilizando actos comunicativos estandarizados por FIPA.

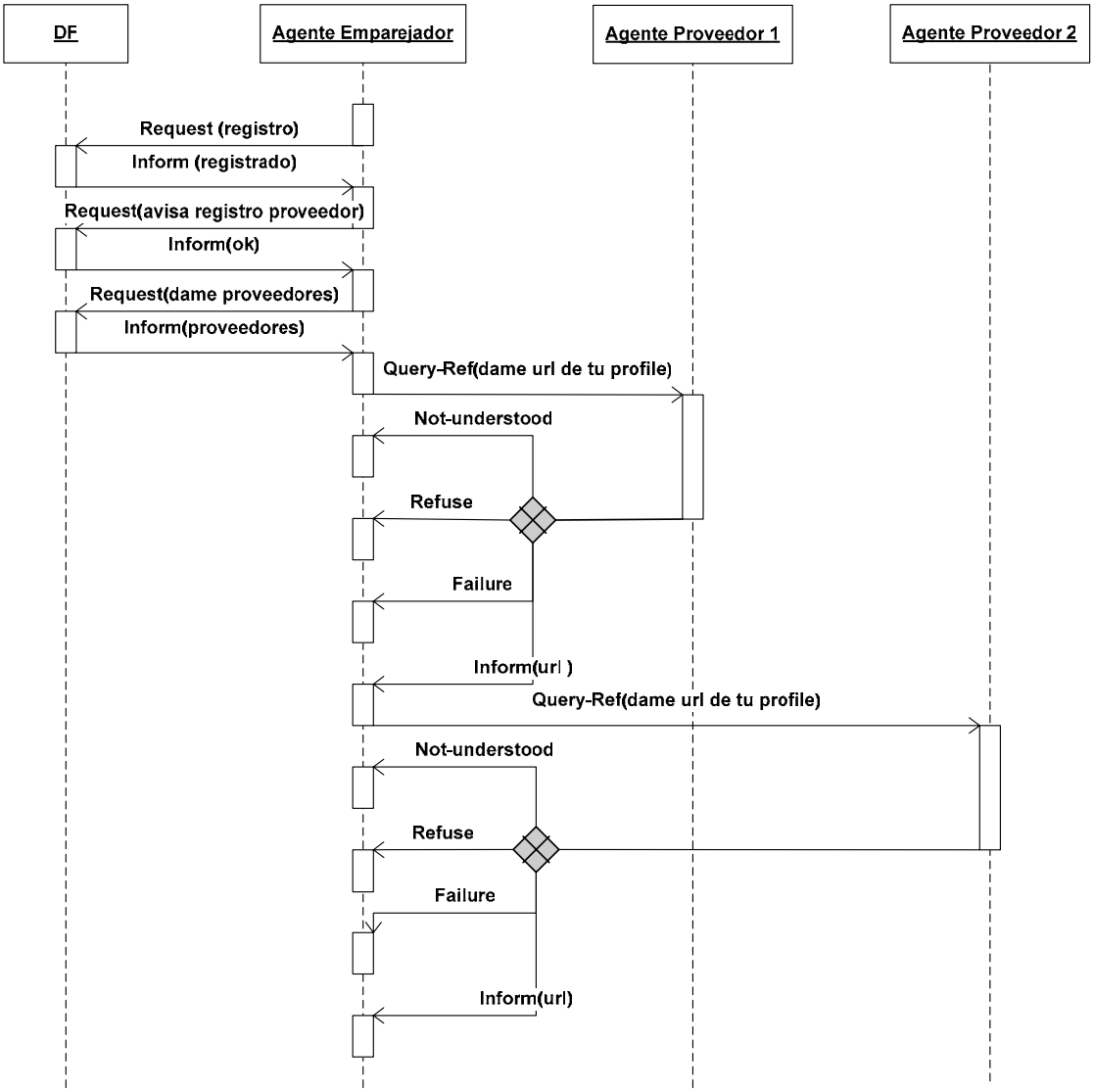


Figura 10.3 Diagrama de protocolo. Agentes proveedores se lanzan antes de ser iniciado el agente emparejador

La figura 10.3 se corresponde con el caso en el que el agente emparejador es lanzado cuando ya hay agentes proveedores activos en la plataforma.

1. En primer lugar el agente emparejador se registra en el facilitador de directorio.
2. Después de recibir la notificación de que el registro se ha realizado con éxito, solicita al facilitador de directorio ser avisado cada vez que un agente del tipo proveedor obtenga el alta en el sistema.
3. Una vez solicitado el aviso pregunta al facilitador de directorio por todos los agentes de tipo proveedor que ya están registrados.

4. Cuando recibe la lista de agentes de tipo proveedor ya registrados, el agente emparejador solicita a cada uno de ellos la URI del profile que describe el servicio al que representa.

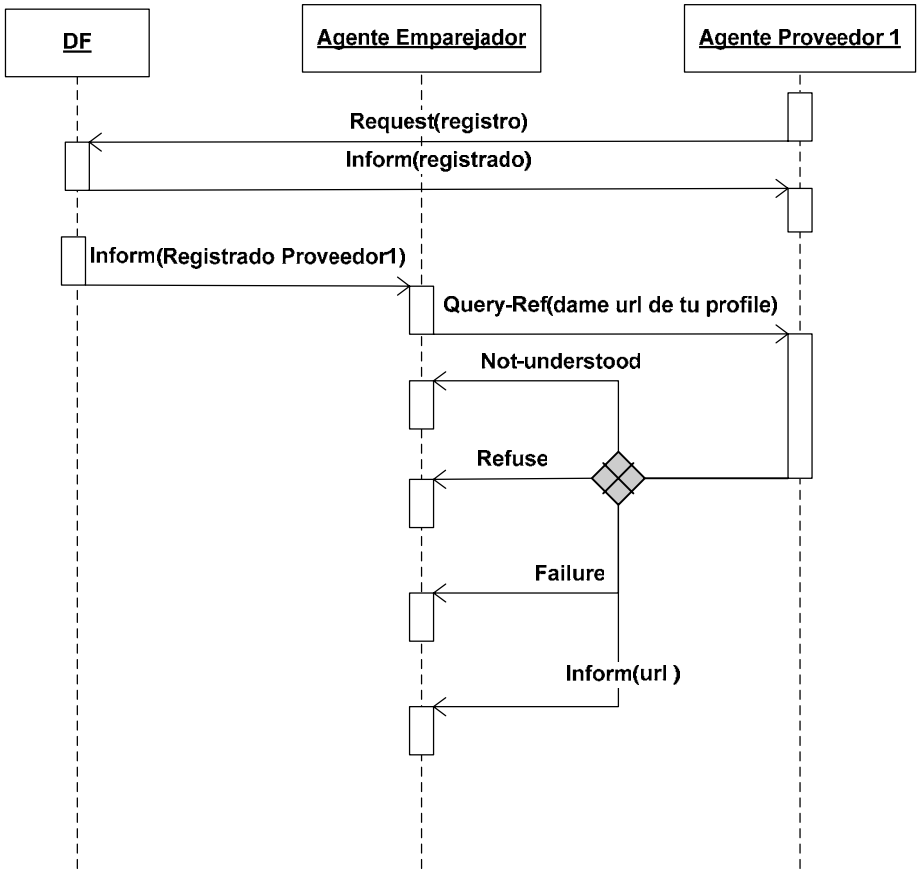


Figura 10.4 Diagrama de protocolo. El agente proveedor es lanzado cuando el agente emparejador ya está activo

En este otro caso representado en la figura 10.4, los agentes proveedores se lanzan después de ser iniciado el agente emparejador.

1. Cuando un agente de tipo proveedor aparece en la plataforma lo primero que hace es registrarse en el facilitador de directorio.
2. Como vimos en el diagrama anterior (figura 10.3) el agente emparejador había solicitado ser avisado cuando agentes de tipo proveedor se registren, por lo que el facilitador de directorio notifica al agente emparejador este nuevo registro.
3. El agente emparejador pide entonces al nuevo agente proveedor que le proporcione la URI del profile que describe el servicio al que representa.
4. El agente proveedor proporciona al agente emparejador la URI solicitada en el paso anterior.

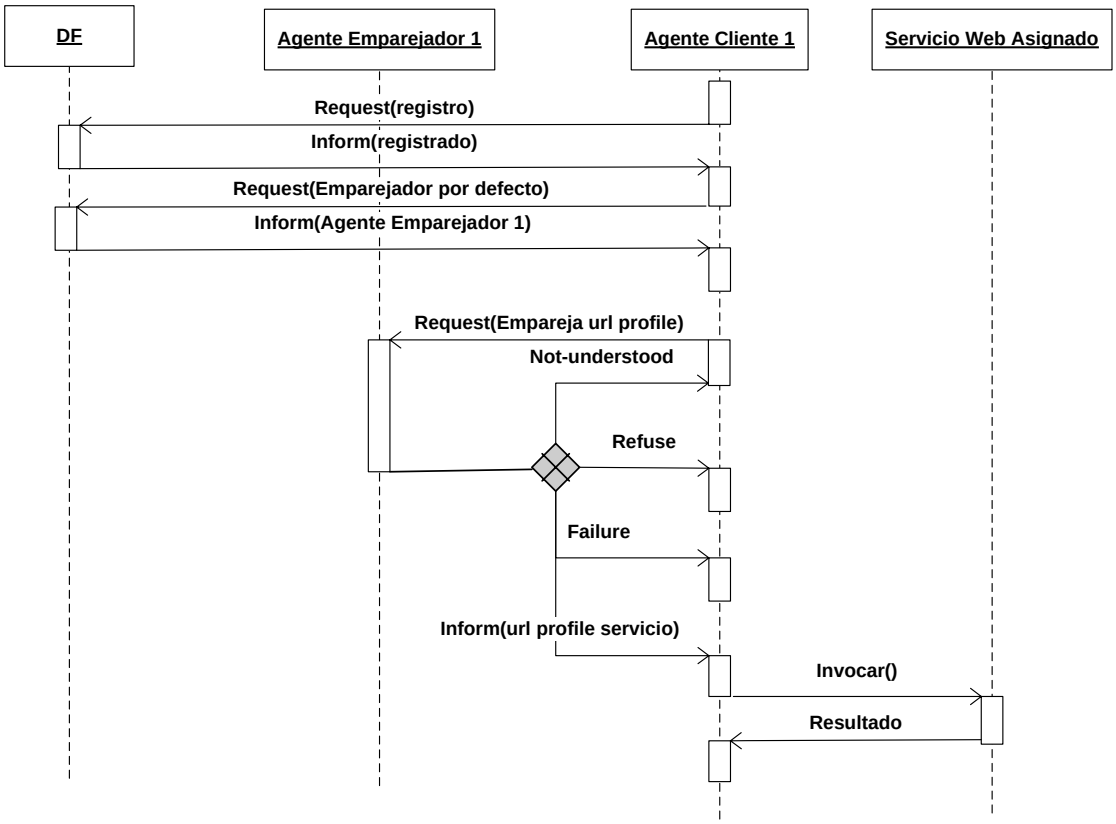


Figura 10.5 Diagrama de protocolo. El cliente es lanzado y hay agentes proveedores y emparejador en el sistema

Por último en la figura 10.5, el cliente es lanzado cuando ya hay un agente emparejador y agentes proveedores en el sistema.

1. Cuando el cliente es lanzado en la plataforma se registra en el facilitador de directorio y le pregunta cuál es el agente emparejador.
2. Una vez el cliente conoce el identificador del agente emparejador le pide que empareje la URI del profile que describe el servicio que busca con las URIs de todos los servicios disponibles.
3. Cuando el agente cliente recibe la URI del servicio que más se parece semánticamente al que busca lo invoca y obtiene el resultado.

Teniendo en cuenta el escenario en el que un agente emparejador es lanzado en la plataforma habiendo un proveedor activo y en el que posteriormente este agente emparejador recibe una petición de emparejamiento por parte de un agente cliente, se puede conocer en más detalle el modelo dinámico del sistema, a través del diagrama de colaboración de la figura 10.6, en la que se pueden ver las interacciones más importantes dentro del sistema multiagente.

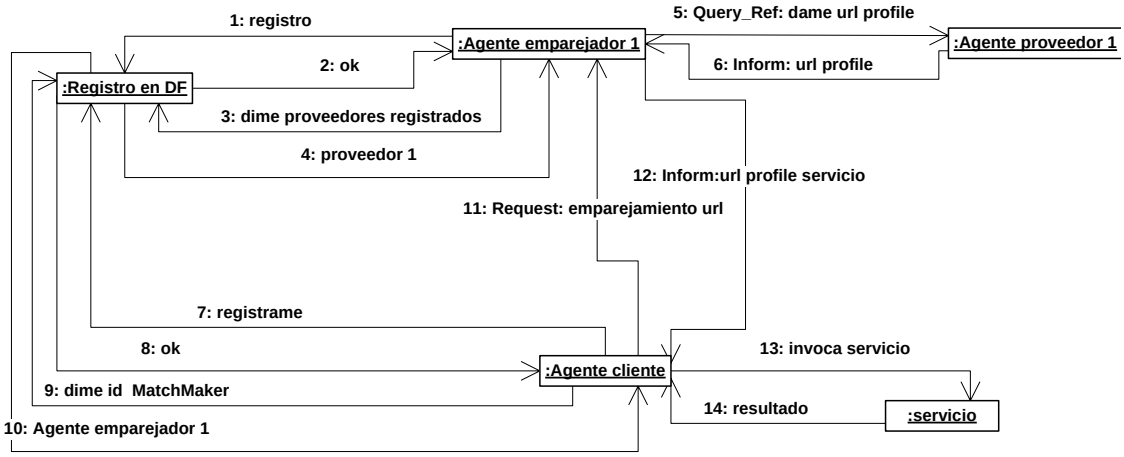


Figura 10.6 Diagrama de colaboración del SMA

La arquitectura elegida para el sistema, junto con los modelos anteriormente comentados, definieron tres elementos independientes que fueron desarrollados como clases típicas de ingeniería del software orientada a objetos tradicional, pero que en realidad se ajustan a lo que en AUML se denomina un "rol de agente". Estos tres roles fueron el de "agente emparejador", "agente cliente" y "agente proveedor". Teniendo en cuenta estos roles, y que la plataforma elegida para la implementación fue JADE, se puede representar el diagrama 10.7 para mostrar el papel de cada elemento en el sistema.

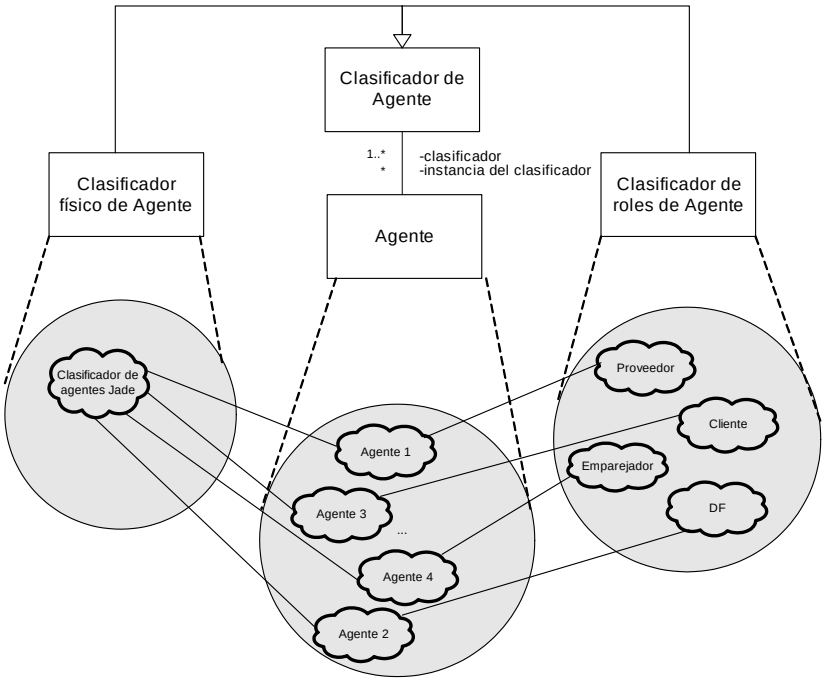


Figura 10.7 Diagrama AUML, roles de los agentes.

10.3 ARQUITECTURA EN CAPAS

Como se observa en la figura 10.8 el sistema puede descomponerse en la siguiente estructura de capas:

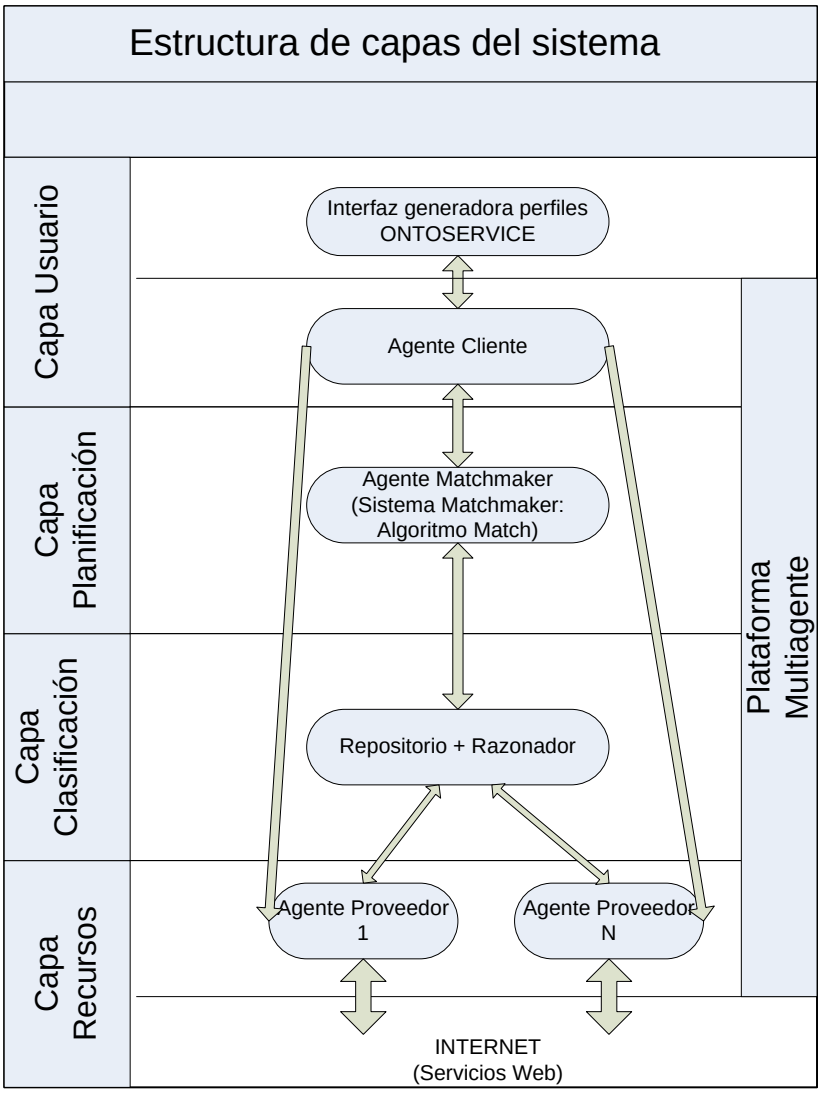


Figura 10.8 Arquitectura en capas del ESWIT

Atendiendo a esta arquitectura en capas se consigue describir más en profundidad el sistema, explicando cada una de ellas así como los elementos que la constituyen.

Como se puede observar la plataforma multiagente está presente en todas las capas aunque no todos los componentes del sistema forman parte de ella. El facilitador de directorio no ha sido representado en nuestra estructura de capas por ser propio de JADE. El sistema está formado por agentes encargados del emparejamiento de

servicios, agentes que representan a clientes dentro de la plataforma y agentes que representan a proveedores de servicios.

10.3.1 Capa de usuario

Esta capa está formada por dos elementos, uno de los cuales (OntoService) ya ha sido explicado extensamente en el capítulo 9. Este elemento no forma parte de la plataforma de agentes debido a que su consumo de recursos no aconsejaba su integración. La solución por la que se optó fue la de crear un agente cliente independiente que recibiese una URI de la descripción del servicio generada por OntoService en función de los datos introducidos por el usuario, pudiendo ser la propia interfaz la que lanzase este agente cliente dentro de la plataforma.

OntoService

La primera necesidad que surgió fue la de una herramienta (OntoService) que combinase la visualización de ontologías con la creación de perfiles de búsqueda. A partir de este análisis se inició como experiencia académica la realización de una herramienta basada en la integración de capacidades para definición de perfiles de SWS con visualización y verificación de consistencia de los conceptos sobre los cuales interactuaría un determinado servicio.

La definición de perfiles semánticos está encaminada a la búsqueda de servicios y por lo tanto, orientado al usuario, no al proveedor de servicios. Por este motivo, únicamente se genera el fichero de perfil (*Profile*) a partir de una plantilla siendo innecesarios los ficheros de *process* y *grounding*.

Agente Cliente:

El agente cliente es el representante del cliente dentro de la plataforma. Indica al agente emparejador la URI de la descripción del servicio que busca. Una vez obtenida la URI de la descripción semántica del servicio web que mejor satisfaga sus necesidades, extrae de los archivos asociados la información necesaria y realiza posteriormente su invocación.

Lo más importante a tener en cuenta es que se debía intentar conseguir un cliente genérico para invocar cualquier servicio, sin necesidad de recompilar la interfaz en cada invocación. Además se debía invocar dinámicamente, sin necesidad de tener un *stub* para cada posible servicio. Esto era necesario porque los servicios disponibles aparecen y desaparecen dinámicamente en la plataforma y sería inviable tener preparado un cliente para cada uno de los posibles servicios que en algún momento determinado pueden estar disponibles.

Para conseguir generar la interfaz gráfica dinámicamente, en función de la descripción del servicio, se optó por utilizar la librería HTMLPane [HTMLPane04], que permite generar una interfaz gráfica mediante un formulario HTML el cual es mostrado en un panel Java. Además permite capturar desde el programa los valores introducidos en sus campos al pulsar el botón de *enviar*. El agente cliente utilizará el protocolo SOAP para invocar el servicio web. Para lograr la invocación dinámica se utiliza WSIF (*Web Services Invocation Framework*) [WSIF04]. Para ello es necesario acceder al archivo grounding asociado al profile con el que el cliente haya sido emparejado, para extraer de

él la URI de la descripción WSDL asociada al servicio. Una vez obtenido el archivo WSDL se procede a la invocación, y el resultado es mostrado al cliente en una interfaz diseñada para ello.

El pseudocódigo para el agente cliente es el siguiente:

```
registrarse en DF
preguntar a DF el identificador del emparejador
enviar petición de emparejamiento al agente emparejador
recibir profile del servicio
procesar profile del servicio para obtener parámetros entrada
generar interfaz gráfica en función del profile del servicio
recoger datos introducidos e invocar servicio
recoger resultado de la invocación
mostrar resultado
```

La figura 10.9 muestra el diagrama de estados correspondiente a este tipo de agente.

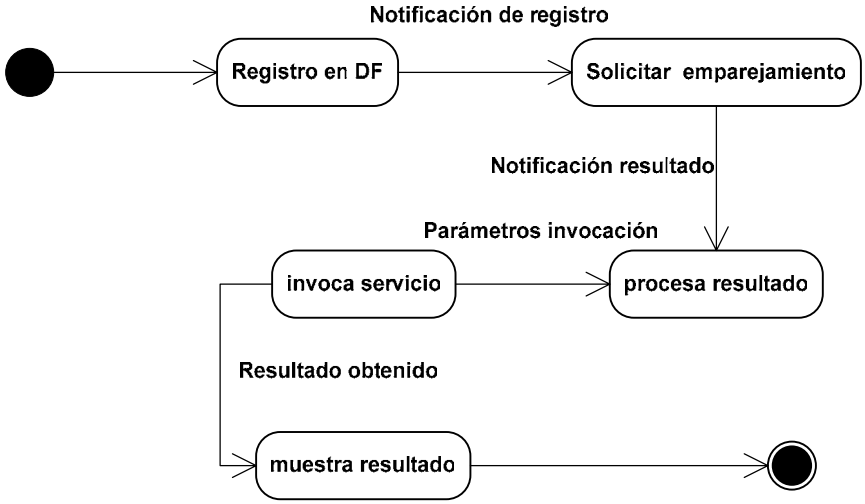


Figura 10.9 Diagrama de estados del agente cliente

10.3.2 Capa de planificación

Agente Emparejador (matchmaker)

Este agente emparejador puede servir como alternativa o complemento a UDDI y tiene como principal característica el hecho de estar basado en semántica.

El modelo dinámico del agente emparejador se puede resumir mediante la figura 10.10 que representa los diferentes estados por los que pasa el agente emparejador.

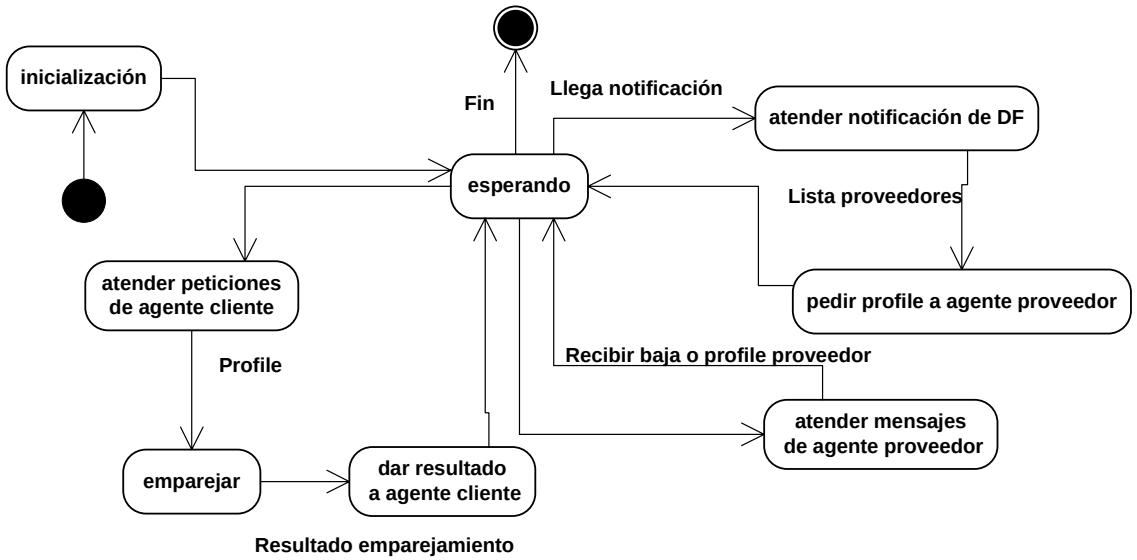


Figura 10.10 Diagrama de estados del agente emparejador

Para facilitar la construcción de este agente se desarrolló un emparejador de descripciones semánticas en un paquete JAVA (sma.Matchmaker) que puede ser utilizado desde cualquier programa que lo necesite, en este caso nuestro agente emparejador. Internamente hace uso del algoritmo de emparejamiento propuesto en el capítulo 7.
Este emparejador se encarga de recibir URIs de descripciones de peticiones de servicios contrastando su contenido con las descripciones de los servicios anunciados para identificar cual de ellos se acerca más a las necesidades del cliente que hace la petición. A cada pareja obtenida se le asigna un peso que indica la proximidad semántica entre los dos conceptos que se relacionan. Este peso se obtiene interrogando a un razonador sobre la jerarquía definida por una ontología de conceptos que define el vocabulario común a utilizar. Las peticiones de clientes y los nuevos anuncios o bajas de servicios ofertados son gestionados dinámicamente. Tanto las ontologías como las descripciones de servicios utilizadas están orientadas a servicios de información sobre tráfico vial, pero el emparejador puede trabajar con cualquier ontología sin necesidad de hacer modificaciones.

La función principal es la de buscar un servicio que se ajuste a las necesidades del cliente, aunque también destacan las tareas de ofertar servicios nuevos o darlos de baja. Por tanto se diferencian dos tipos de actores: clientes y proveedores de servicios. En la figura 10.11 se puede observar los casos de uso para estos dos actores.

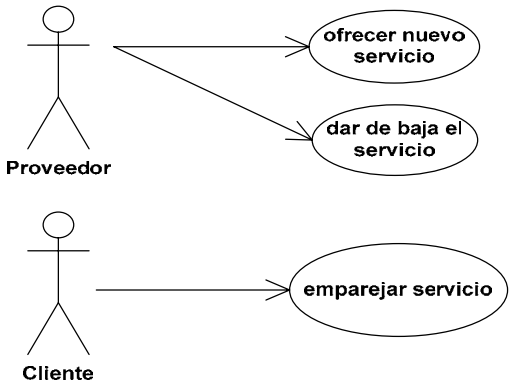


Figura 10.11 Diagrama de casos de uso del emparejador

A partir de los casos de uso representados en la figura 10.11 se definieron los atributos, las relaciones y las clases representadas en el diagrama de la figura 10.12, este diagrama de clases representa el contenido de nuestro paquete emparejador semántico (sma.Matchmaker) del que habíamos hablado al comentar la figura 10.2

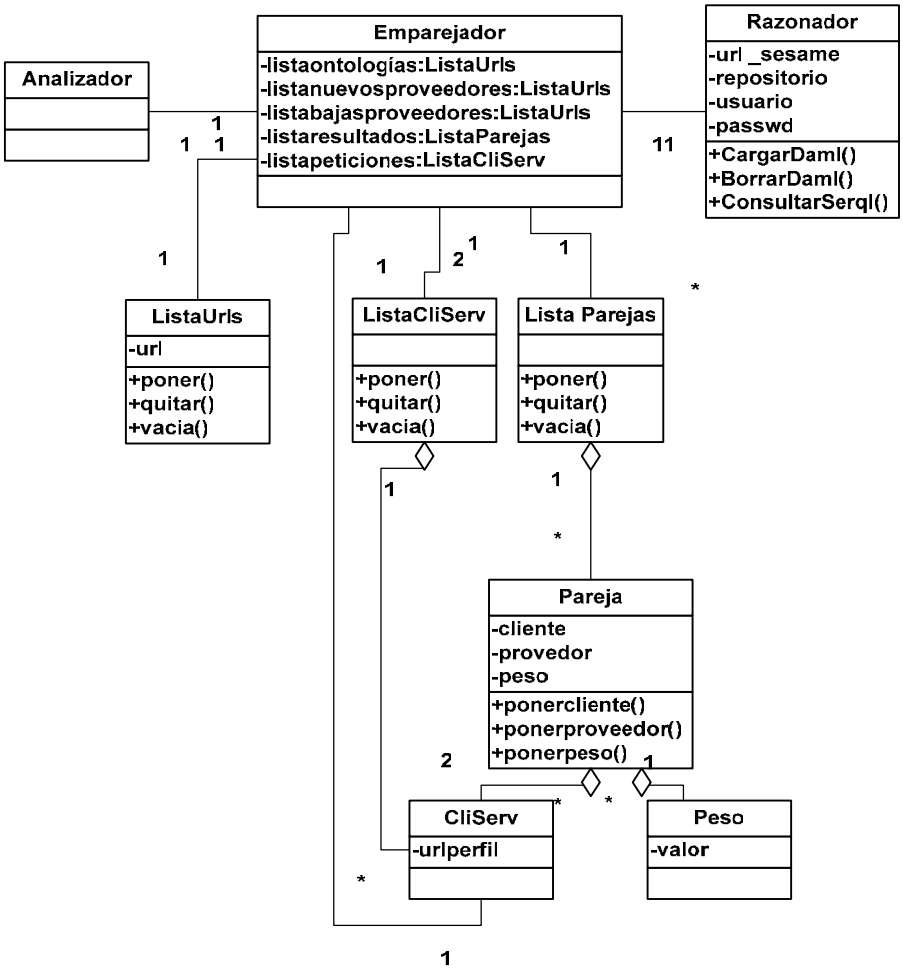


Figura 10.12 Diagrama de clases del emparejador de servicios

La clase principal es la clase *Emparejador*, pero hay clases auxiliares de gran importancia por representar elementos básicos del sistema. *Analizador* representa al "parser" o intérprete utilizado para extraer información de los archivos utilizados, *Razonador* representa a la clase encargada de facilitar la interacción con SEBOR, *CliServ* es una clase que representa tanto a anuncios de proveedores como a peticiones de clientes. *Pareja* es una clase encargada de asociar peticiones y anuncios en función de un *Peso*, y tanto *ListaUrls, ListaCliserv y ListaParejas* con clases encargadas de facilitar la utilización de elementos del tipo *Url, CliServ y Parejas*, respectivamente.

El comportamiento dinámico del emparejador puede resumirse con los diagramas de secuencia de las figuras 10.13, 10.14 y 10.15, y el diagrama de estados de la figura 10.16.

Las figuras 10.13 y 10.14 corresponden a los diagramas de secuencia para los casos de uso alta y baja de un servicio en la plataforma:

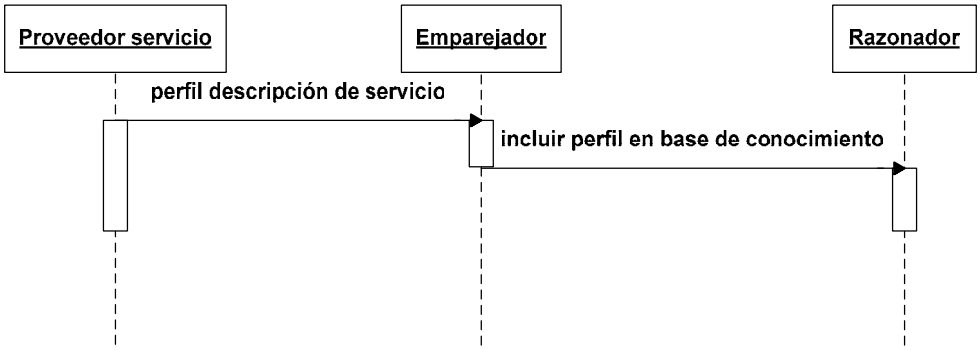


Figura 10.13 Diagrama de secuencia para el caso de uso "nuevo servicio"

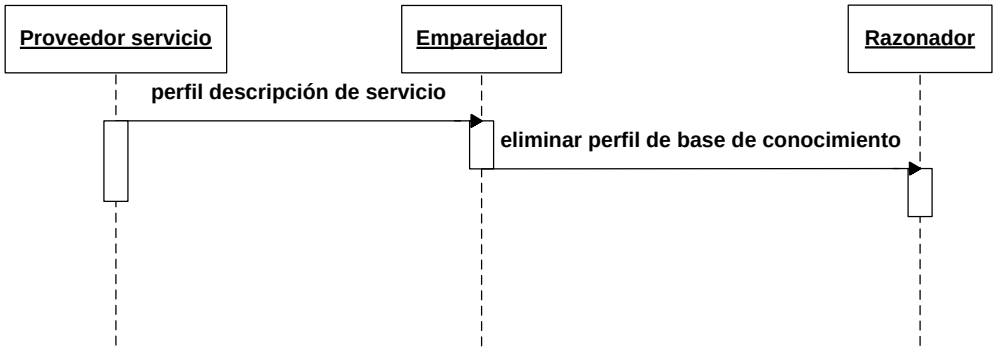


Figura 10.14 Diagrama de secuencia para el caso de uso "dar baja servicio"

El caso de uso más complejo, es el de "emparejar servicio" y está formado por los siguientes pasos:

1. El programa externo actuando como cliente proporciona al emparejador la descripción de un servicio.
2. El emparejador examina la descripción de servicio proporcionada por el cliente y extrae la información necesaria para contrastarla con la de los servicios disponibles en la base de conocimiento.
3. El emparejador interroga al razonador y asigna un peso a las parejas obtenidas en función de su proximidad en la jerarquía definida por la ontología de conceptos.

El correspondiente diagrama de secuencia para este caso se puede observar en la figura 10.15:

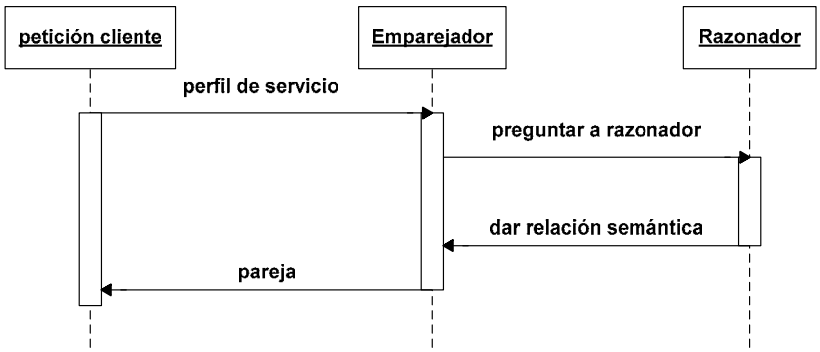


Figura 10.15 Diagrama de secuencia para el caso de uso “emparejar servicio”

El diagrama de estados se observa en la figura 10.16.

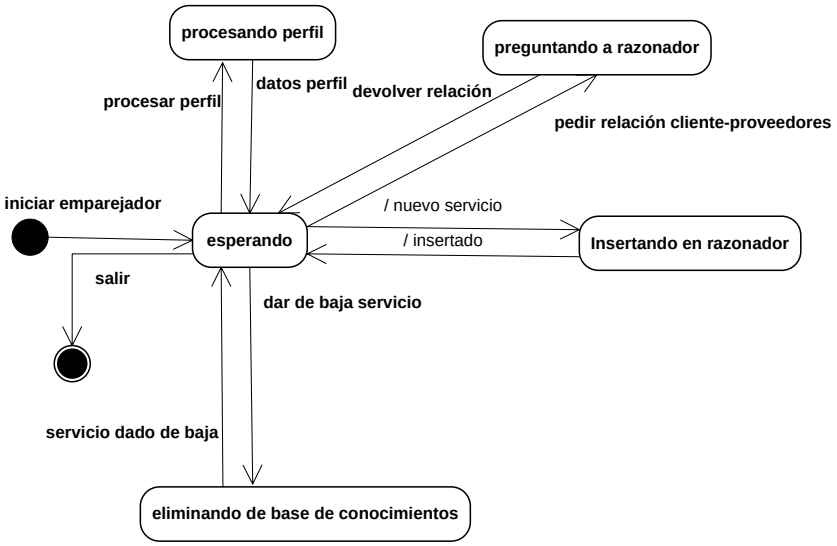


Figura 10.16 Diagrama de estados del emparejador semántico

En la figura 10.17 se puede observar el despliegue del emparejador implementado.

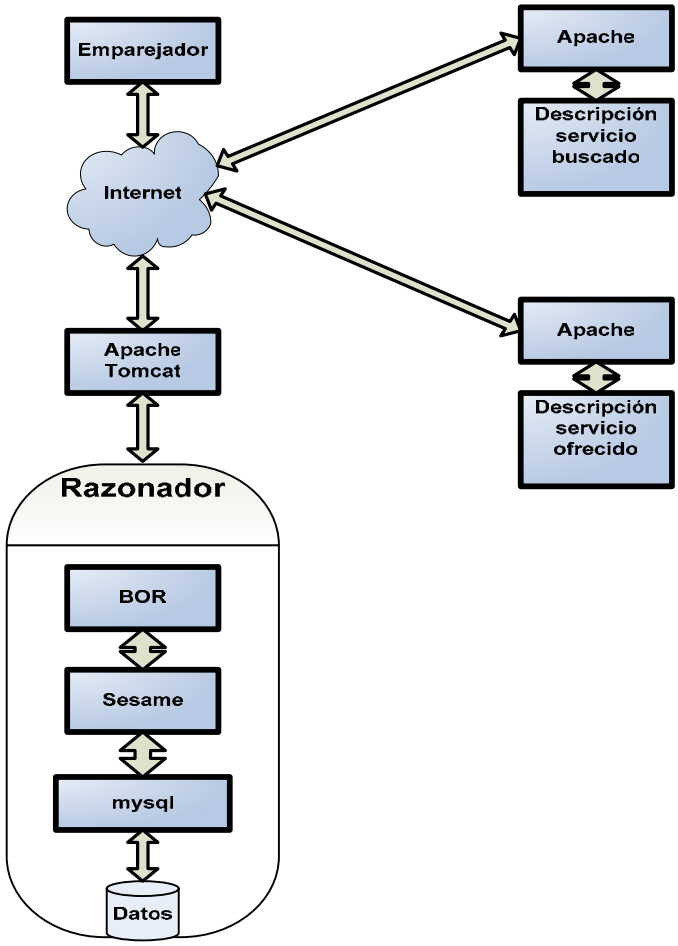


Figura 10.17 Diagrama de despliegue del emparejador semántico

Cada agente JADE se implementa como un único hilo de ejecución por lo que para repartir el tiempo de ejecución de este hilo, hace uso de un planificador no expulsivo, pudiendo ejecutar diferentes tareas concurrentemente. La programación del agente consiste por tanto en definir las tareas (*behaviours*) que necesita para hacer su trabajo. Estas tareas se invocan cuando se produce un evento asociado a ellas (llegada de un mensaje o un salto de un temporizador). El hecho de que este planificador sea no expulsivo implica que cuando se ejecuta una tarea ésta no pueda ser interrumpida hasta que acabe. Esto puede ser un problema muy importante si alguna tarea que necesitemos hacer es demasiado costosa, ya que mientras no finalizase dejarían de atenderse el resto de tareas como por ejemplo las encargadas del tratamiento de mensajes procedentes de otros agentes, y estos mensajes se perderían. El agente emparejador hace uso del paquete que contiene al emparejador (capítulo 7). El algoritmo de emparejamiento que implementa puede ser bastante costoso si el número de proveedores es elevado por lo que aunque los desarrolladores de JADE no aconsejan lanzar hilos desde un agente, no hubo otra alternativa.

Un programa exterior intercambia los datos con el emparejador semántico mediante cuatro listas: una lista donde el programa externo deja los *profiles* describiendo

peticiones de servicios de clientes para que el emparejador los empareje, una lista para que el programa externo deje los *profiles* de los proveedores a dar de baja, otra para las nuevas altas, y otra lista donde el emparejador deje el resultado del emparejamiento. Además, en la llamada recibe como argumento las listas de ontologías a usar y el resto de parámetros necesarios por el paquete emparejador.

Además, el agente y el emparejador están sincronizados por la lista de *profiles* de peticiones. El emparejador permanece dormido hasta que el agente emparejador inserta un *profile* en la lista y le notifica la inserción, por lo que esa lista es tratada como una sección crítica.

Se establecen tres tipos de tareas o *behaviours* para poder implementar las secuencias de comunicación que habíamos definido:

1. Una tarea cíclica basada en un *SimpleBehaviour* para atender las notificaciones de registro del DF,
2. una tarea cíclica para atender los mensajes recibidos tanto de agentes cliente como de agentes proveedor,
3. una tarea cíclica para enviar el resultado del emparejamiento al cliente.

Tanto la primera como la segunda se despiertan cada vez que llega un mensaje nuevo, y ejecutan su método *action()* si el mensaje va dirigido a ellas. Una vez finalizado el tratamiento del mensaje recibido llaman al método *block()* para volver a insertarse en el planificador. Las dos tareas discriminan los mensajes a tratar en función de las performativas.

El tercer comportamiento es distinto a los anteriores debido a que se debe encargar de notificar los resultados del emparejamiento a los clientes. El emparejador sólo tiene un hilo propio[39], y ese hilo está sincronizado con el Agente Emparejador en función de las inserciones de *profiles* de la lista de peticiones de emparejamiento, por lo que no hay forma de que el emparejador notifique al agente cliente que tiene nuevos resultados que entregar. Por lo tanto se optó por utilizar en ese caso un comportamiento controlado por un temporizador que cada vez que se dispare compruebe la lista de resultados para ver si hay alguno nuevo por entregar. En el caso de que sea así lo extrae y se lo envía al cliente. La lista de peticiones es una lista de objetos de tipo Pareja, por lo que solo hay que extraer el identificador del Agente Cliente y enviarle el *profile* con el que haya sido emparejado.

Es el método *setup()* el que se ejecuta al iniciarse el agente, por lo que el pseudocódigo es el siguiente:

[39] JADE recomienda lanzar el menor número de hilos posible.

```
crear lista profiles de peticiones
crear lista de parejas resultado
crear lista URIs de perfiles de prov a insertar
procesar argumentos de entrada
registrarse en DF como tipo MATCHMAKER
listaprov=buscar en DF todos los agentes proveedor
para cada elemento de listaprov
 {
  solicitar profile
 }
lanzar hilo emparejador
suscribirse en DF para ser avisado cuando se registre un proveedor
crear comportamiento0 para atender avisos del DF
crear comportamiento1 para atender peticiones de agentes cliente y proveedor
crear comportamiento2 para comprobar cada N segundos si hay resultados a notificar
insertar en planificador comportamientos 0, 1 y 2
```

Los comportamientos JADE 0 y 1 rellenan una plantilla (MessageTemplate) para indicar los mensajes (tipos de mensajes o "performatives") que deseasen atender, y devolver la respuesta correspondiente en función del estado del diagrama de protocolo en el que se encuentre la conversación.

10.3.3 Capa de clasificación

Forma parte de la plataforma multiagente y se relaciona con las capas de planificación y de recursos. Es la capa esencial en cualquier arquitectura semántica.
La decisión de que servicio se asemeja más semánticamente al servicio buscado se realiza consultando un razonador que infiere sobre ontologías que acumulan conocimiento sobre el dominio del problema. En este proceso intervienen los siguientes elementos que constituyen esta capa:

Razonador

Como razonador se utilizó BOR, el cual está basado en lógica descriptiva y tiene soporte para inferencias sobre instancias y sobre conceptos. Este razonador puede ser usado tanto con ontologías escritas en DAML+OIL (con algunas restricciones), y con ontologías escritas en la especificación OWL Lite. Además se puede incorporar a la aplicación Sesame, para dar soporte de ontologías DAML+OIL y poder realizar razonamiento y todas las funciones descritas en la información almacenada en los repositorios.

Repositorio

Como repositorio se utilizó Sesame, debido a que permite añadir y eliminar información escrita en RDF en los repositorios, y puede almacenar esta información en cualquier base de datos. Los prerrequisitos para poder instalar y utilizar Sesame son dos: un servidor web con soporte para servlets y una base de datos.

El servidor web sirve para poder ejecutar los JSP's que Sesame incluye y que permite utilizar todas las funciones que proporciona. En este caso se ha utilizado Apache Tomcat [Apache04] por su alta compatibilidad con Sesame.

La base de datos sirve para almacenar los datos que se introduzcan en los repositorios de Sesame, y se utilizó MySQL debido a las buenas referencias de compatibilidad con Sesame y ser de código abierto y libre distribución. Para poder utilizar MySQL [MySQL04], solo hace falta copiar los *drivers* JDBC en el mismo directorio donde está el código de Sesame así como crear la base de datos y usuarios necesarios.

Para dar soporte a DAML+OIL se utilizó el plugin de Sesame llamado SeBOR (Sesame+BOR) que acepta modelos de datos semánticos de DAML+OIL en Sesame (ver figura 10.18).
De las nuevas funciones que permite SeBOR, destacan la posibilidad de insertar ontologías DAML+OIL en el repositorio, insertar conocimiento escrito en KRSS [Pat93] así como la posibilidad de extraer las subontologías que se le indiquen.

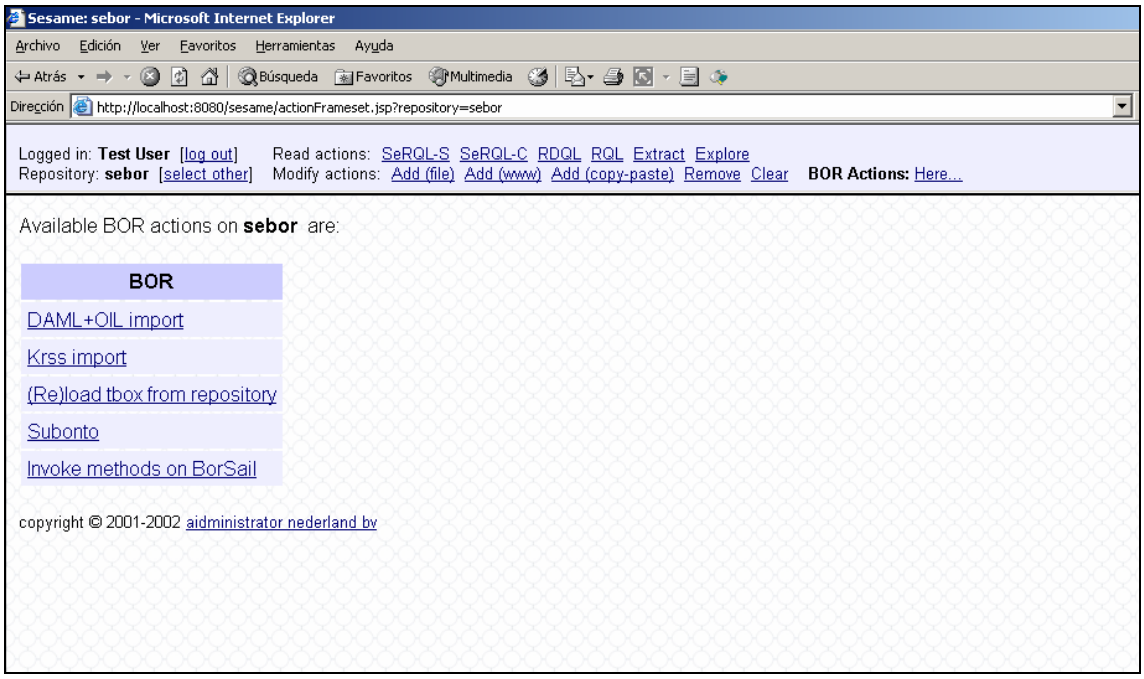


Figura 10.18 Interfaz gráfica Sesame-SeBOR

El código de configuración de un repositorio para DAML+OIL es como el siguiente:

```
<repository id="sebor">
             <title>sebor</title>
             <sailstack>
                    <sail class="com.ontotext.omm.bor.BorSail">
                           <param name="daml.schema.url"
value="http://localhost:8080/sesame/bor/2002/08/daml.fix.xml"/>
                           <param name="bor.schema.url"
value="http://localhost:8080/sesame/bor/2002/07/bor"/>
                    </sail>
                    <sail
class="org.openrdf.sesame.sailimpl.rdbms.RdfSchemaRepository">
                           <param name="jdbcDriver"
value="org.gjt.mm.mysql.Driver"/>
                           <param name="jdbcUrl"
value="jdbc:mysql://localhost:3306/sebor"/>
                           <param name="user" value="root"/>
                           <param name="password" value="eswit"/>
                    </sail>
             </sailstack>
             <!--Access Control List can contain zero or more 'user'
elements-->
             <acl worldReadable="true" worldWritable="true">
                    <user login="testuser" readAccess="true"
writeAccess="true"/>
             </acl>
</repository>
```

En esta configuración se define el repositorio que va a utilizar el agente emparejador para almacenar los perfiles de los Servicios Web correspondientes a los proveedores dados de alta en el sistema multiagente. Se indica que debe usar los *drivers* JDBC de MySQL, así como el nombre y el lugar donde reside la base de datos MySQL que debe emplear para almacenar los datos.

Sistema de Consultas

Las consultas fueron realizadas mediante SeRQL (Sesame RDF Query Language). Con su ayuda fue posible implementar el algoritmo de emparejamiento tal y como queda expuesto en el capítulo 7.

10.3.4 Capa de recursos

Esta capa está representada exclusivamente por los agentes de tipo proveedor y los SW a los que representan.

Agente Proveedor.

Es el más simple del sistema. El proveedor que quiere anunciar un servicio dentro de la plataforma debe lanzarlo pasándole como parámetro la URI de la descripción de ese servicio. Además de registrarse en el DF como tipo proveedor sólo debe planificar un comportamiento JADE que se encargue de enviar la URI del profile del servicio que represente al agente emparejador cada vez que éste se lo solicite, siempre siguiendo los actos comunicativos de los diagramas de protocolo.

El pseudocódigo del agente proveedor es el siguiente:

```
Registrarse en DF como tipo proveedor
Crear comportamiento para atender peticiones de la URI del profile del servicio que representa
Planificar comportamiento
```

El diagrama de estados correspondiente al agente proveedor puede ser visto en la figura 10.19:

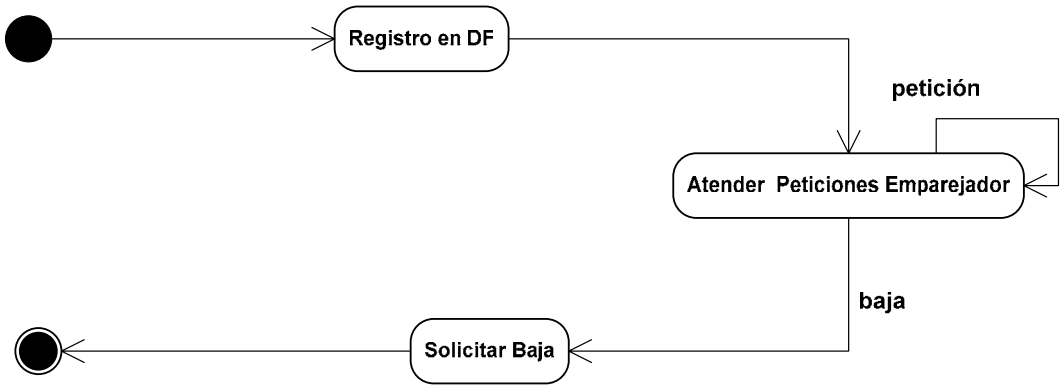


Figura 10.19 Diagrama de estados del agente proveedor

Servicios Web

En la subcapa inferior de las dos que forman la capa de recursos están situados los SW anunciados por los agentes proveedores dentro de la plataforma. Estos SW no están por tanto integrados dentro del sistema multiagente, son SW tradicionales representados por los agentes proveedores dentro de la plataforma.

Para comprobar el correcto funcionamiento del ciclo completo de anuncio, descubrimiento e invocación hubo que crear un servicio web de prueba ya que hasta donde llegó la búsqueda bibliográfica no se encontró ningún servicio web relacionado con el área de tráfico. Para ello se optó por adaptar la información referente a previsiones de tráfico de la web de la Dirección General de Tráfico para ser accedida a través de un servicio web creado por nosotros.

El servicio que implementamos tenía dos componentes:

Agente actualizador: La captura de datos en la web y su tratamiento para poder ser utilizados desde nuestro servicio web podría haberse hecho de muchas formas, pero se optó por que un agente que se lanzase también en la plataforma se encargase de este trabajo. En este agente "wrapper" se definió una tarea con un temporizador, en la que se indica un intervalo de tiempo al iniciarla, y cada vez que se cumple este período extrae

información de la web y la deposita en un repositorio SEBOR. La extracción de la información se hace con las librerías WebL y un script generado para ellas que convierte la información obtenida de la web a formato DAML + OIL y posteriormente es almacenada en el repositorio SeBOR.

Servicio de información de tráfico: Tal y como quedó expuesto en el capítulo 8, para generar la estructura básica de nuestro servicio web se utilizaron las herramientas de Apache AXIS y se completó con llamadas al código usado para obtener la información sobre previsiones de tráfico a partir de los datos almacenados en el repositorio SEBOR. El contenedor de servlets usado para Axis fue Apache Tomcat.

10.3.5 Despliegue del sistema

El diagrama de despliegue del sistema con el servicio web creado puede observarse en la figura 10.20.

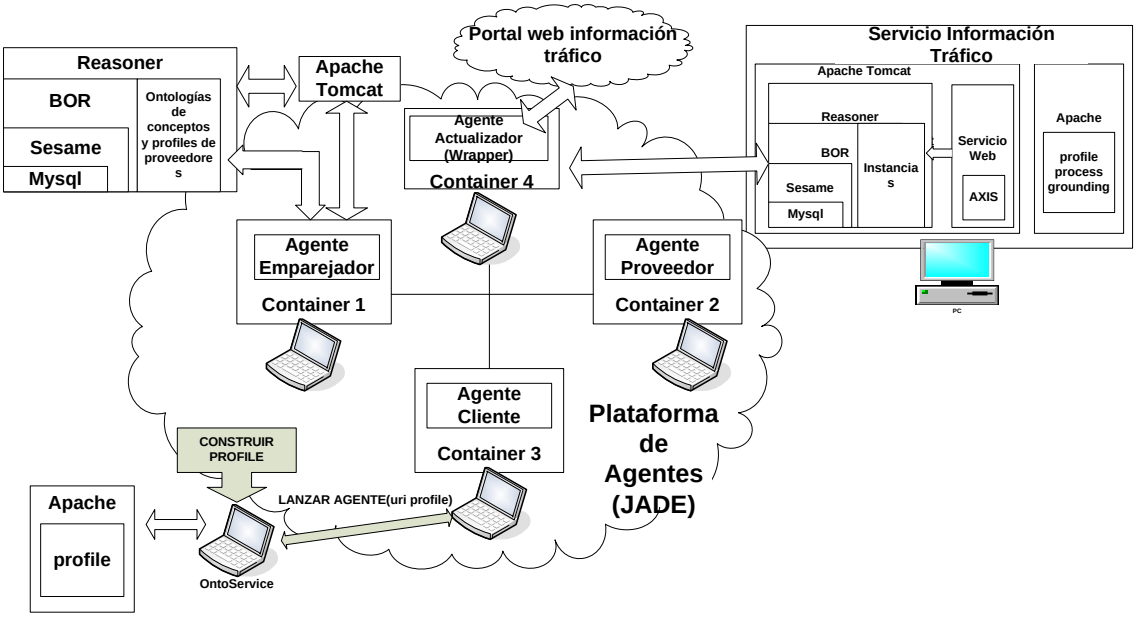


Figura 10.20 Diagrama de despliegue. Arquitectura del sistema.

Se puede ver cómo la plataforma de agentes está distribuida, hay varios contenedores situados en distintos ordenadores que en conjunto forman el sistema multiagente. También puede observarse cómo la aplicación OntoService, utilizada para ayudar a la creación de *profiles* de servicios buscados por clientes e iniciar agentes cliente está fuera de la plataforma de agentes JADE, es por tanto un programa externo. Lo mismo sucede con el servicio web, que está fuera de la plataforma y es accesible a través de un servidor de aplicaciones Apache Tomcat. Los archivos *profile*, *process*, *grounding*, *service* y *wsdl* utilizados para describir servicios son accesibles por web gracias a

servidores web Apache, al igual que los archivos *profile* que describen servicios buscados por clientes.

El razonador es accedido por el emparejador y por el servicio web también a través de Apache Tomcat. El agente actualizador está dentro de la plataforma y cada cierto tiempo extrae información de la web, pero no interviene en las tareas de anuncio, descubrimiento e invocación de SW.

10.4 VERIFICACIÓN DE LA INTEGRACIÓN DEL EMPAREJADOR EN LA PLATAFORMA DE AGENTES

A continuación se detalla el proceso de verificación del sistema realizada para comprobar que el prototipo desarrollado era funcional. Como elemento de testeo se tomaron dos perfiles de servicio especificados mediante sus correspondientes ontologías *profile.* Uno de los *profiles* describía al servicio "previsión de tráfico" y el otro un servicio ficticio llamado *TraficoCatProfile*. El *profile* que describía al servicio previsión fue generado con la herramienta *OntoService* descrita en en el capítulo 9 y el agente cliente fue lanzado desde esta herramienta pasándole como parámetro la URI del *profile* generado.

Descripción del caso de uso:

En la figura 10.21 se puede ver, en primer lugar, cómo se lanzó el contenedor principal de la plataforma, y después el agente *Sniffer* para espiar las comunicaciones entre los agentes que iban a ir siendo iniciados. En la imagen se observa el resto de agentes lanzados por defecto por la plataforma, así como la interfaz gráfica del agente Sniffer. En este caso se indicó al Sniffer que espiase al agente DF (facilitador de directorio).

Una vez hecho esto, se lanzó el agente emparejador (al que llamamos Matchmaker) en otro contenedor (en este caso llamado Container-1). En la figura 10.22 se puede observar cómo se añadieron a la plataforma el nuevo contenedor y el agente.

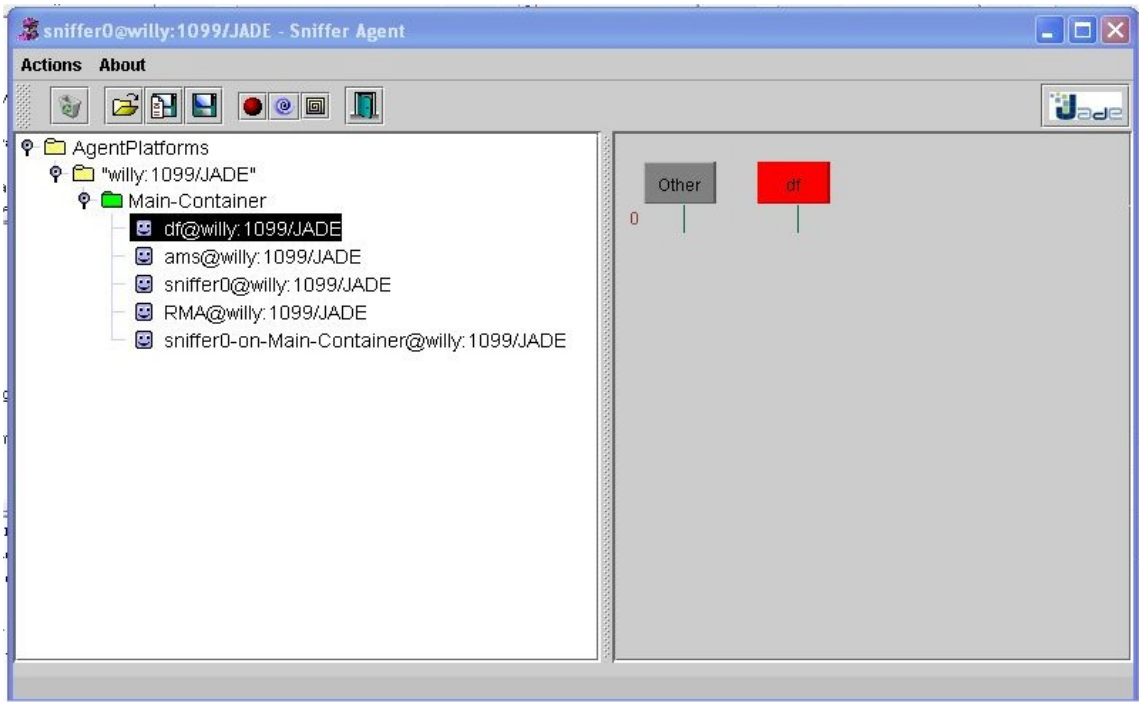


Figura 10.21 Sniffer espiando a DF

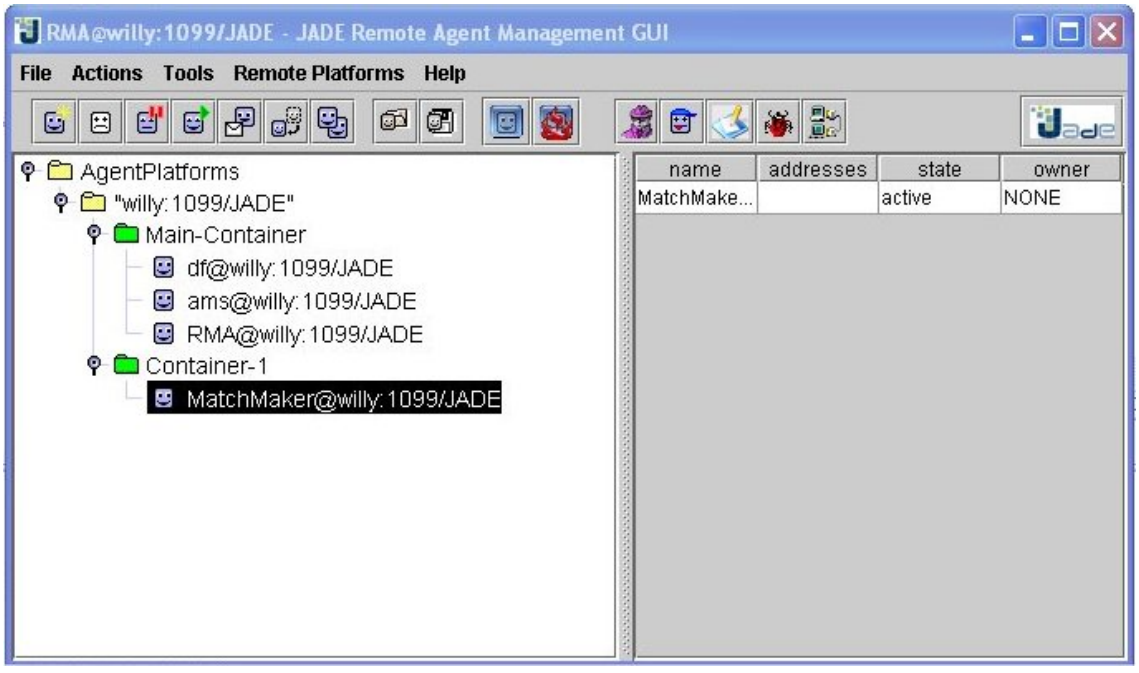


Figura 10.22 Agente emparejador lanzado en un contenedor distinto

La interfaz gráfica del Sniffer fue usada para hacer un seguimiento de todos los elementos que iban apareciendo en la escena.
En la figura 10.23 se puede observar el intercambio de mensajes que se produjo entre el facilitador de directorio y el agente emparejador al aparecer éste último en la plataforma.

La descripción de los diagramas de protocolo seguidos fue presentada en la sección 10.2.1.

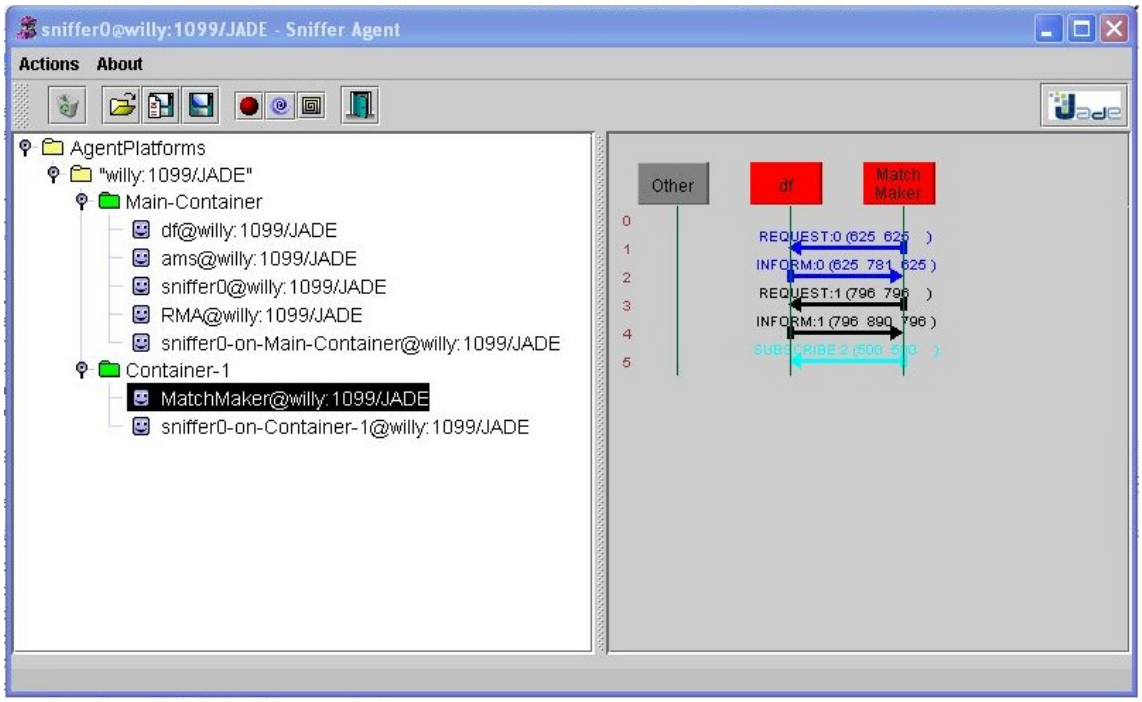


Figura 10.23 Intercambio de mensajes entre DF y Matchmaker

Una vez lanzado el agente emparejador se lanzó el proveedor que representaba en la plataforma a nuestro servicio previsión.
A través de la interfaz gráfica del agente Sniffer, podemos observar el contenido de los diferentes mensajes. Por ejemplo, en la parte derecha de la figura 10.24 puede observarse el contenido del mensaje enviado por el agente *prov1* al agente emparejador, en respuesta al mensaje QUERY-REF que éste le había enviado para preguntarle a qué "profile" representaba dentro de la plataforma.

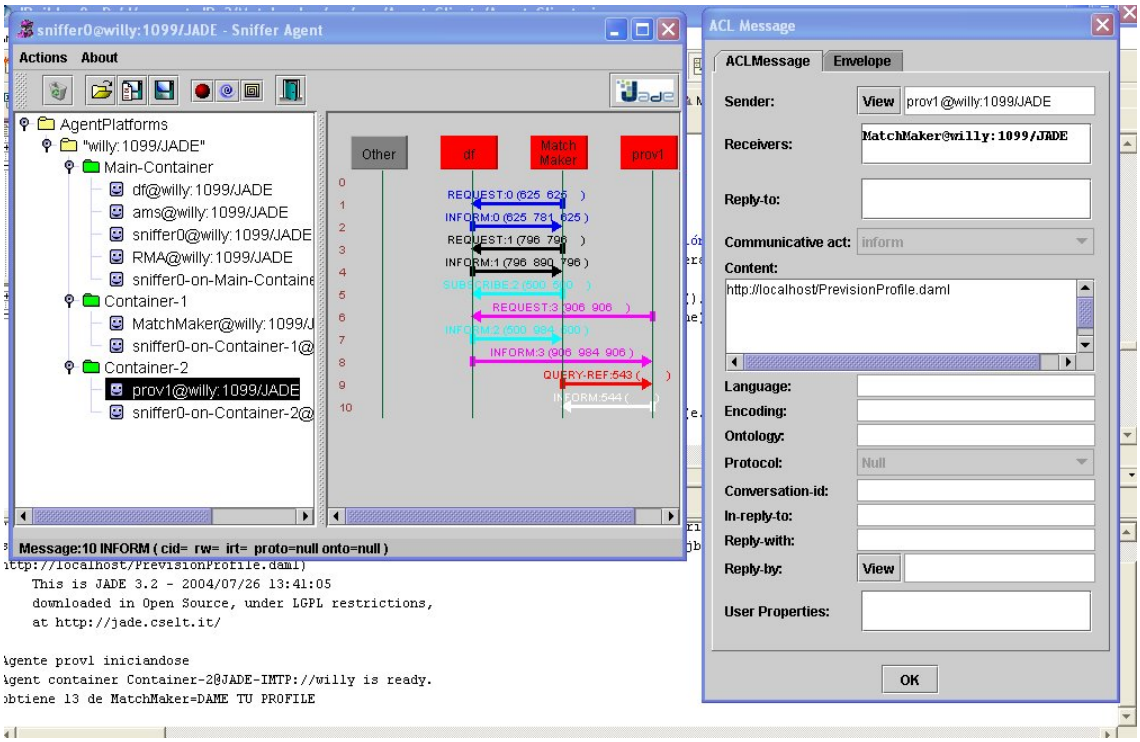


Figura 10.24 Intercambio de mensajes entre los agentes DF-Matchmaker-Proveedor (perfil del servicio)

En la figura 10.25, puede observarse cómo se lanzó otro contenedor con un servicio que representaba a un servicio ficticio llamado TraficoCatprofile, y los diagramas de protocolo que se fueron generando a medida que los agentes intercambiaban mensajes.

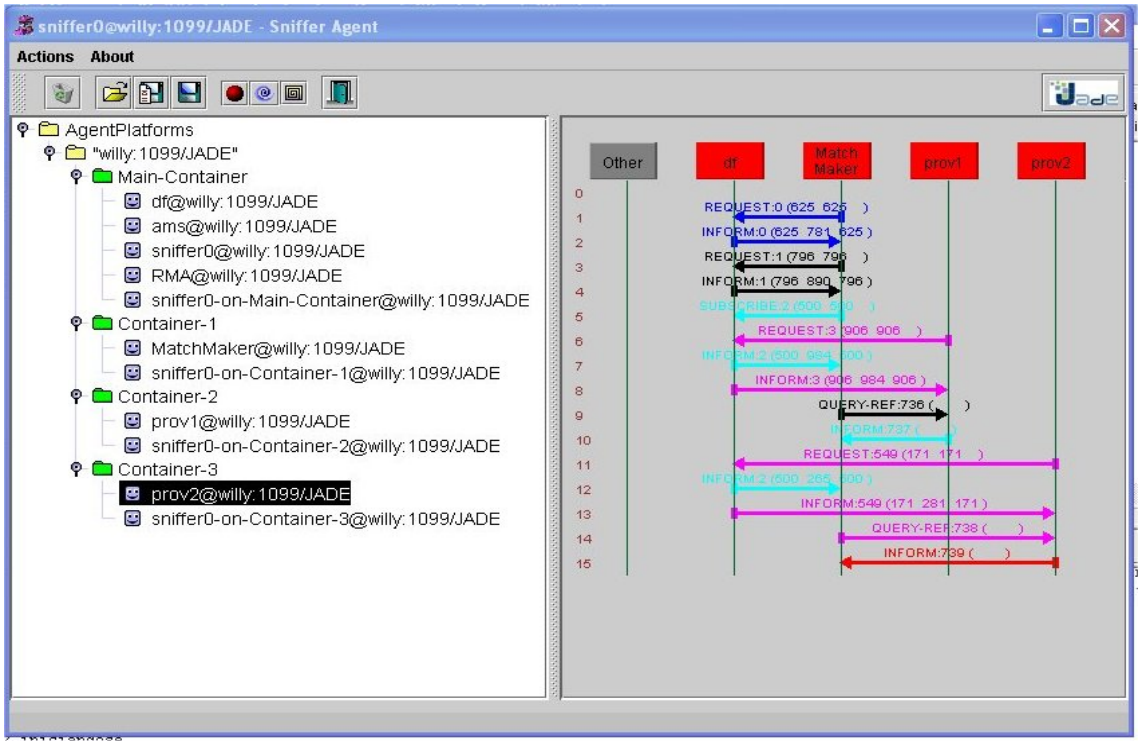


Figura 10.25 Lanzamiento de otro proveedor. Intercambio de mensajes.

El último agente que se lanzó fue el agente cliente, al que se le comunicó como parámetro de invocación la URI de un profile que describía un servicio más próximo semánticamente a nuestro servicio previsión que a TraficoCatprofile.

La figura 10.26 corresponde de nuevo a la interfaz gráfica del Sniffer, y en ella se puede observar cómo el cliente envió un mensaje pidiéndole al agente emparejador que le buscase entre los disponibles un servicio que se pareciese a la descripción dada.

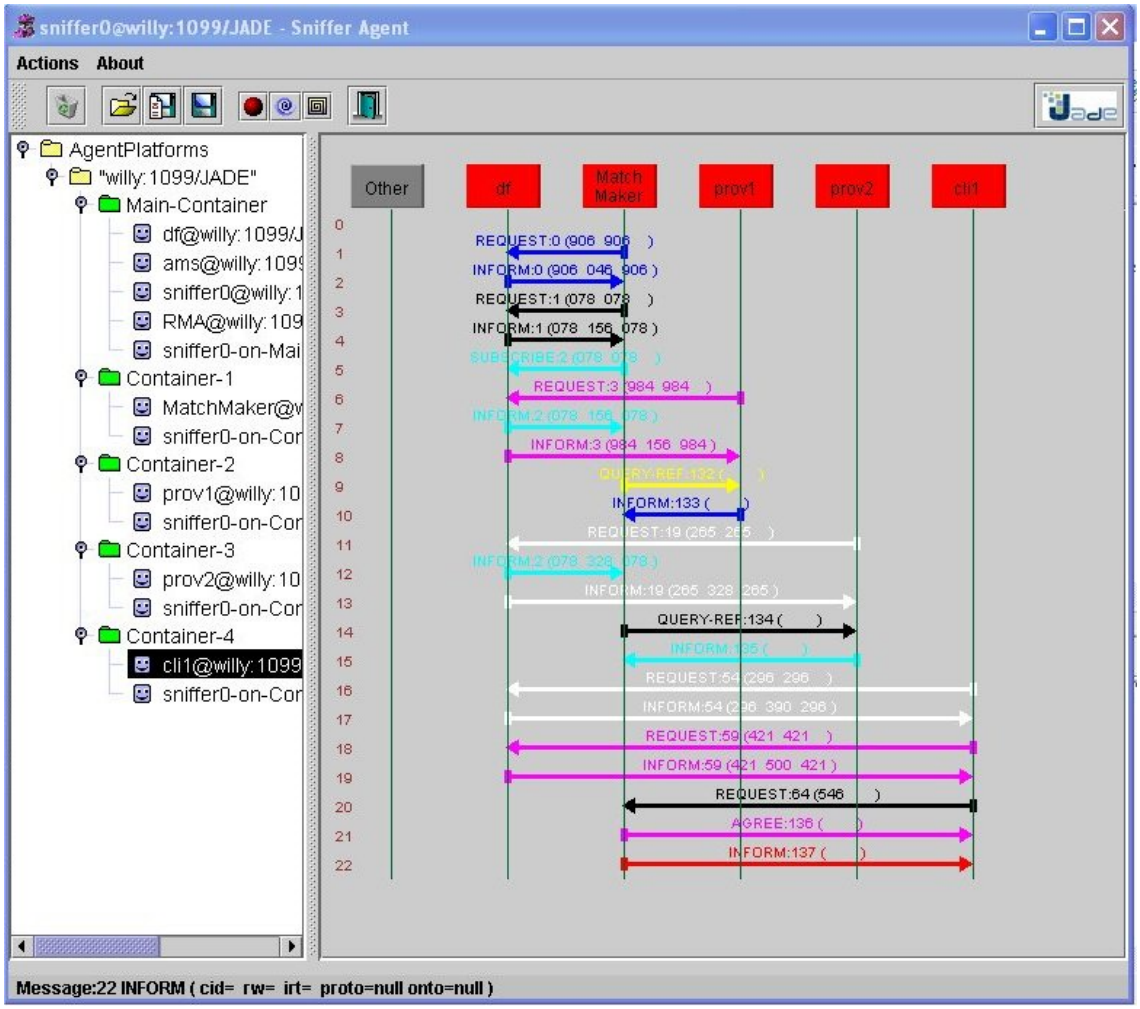


Figura 10.26 Agente cliente envía petición de búsqueda de servicio

El emparejador contrastó el profile recibido del cliente con todos los que tenia en su repositorio y decidió que el correspondiente a nuestro **servicio previsión** es el que más se le acercaba semánticamente.
En ese momento el agente emparejador notificó al agente cliente la URI del servicio previsión y el agente extrajo de este profile, ayudándose de un analizador (*parser*), después que el usuario pulsase sobre el botón generar, la información necesaria para construir dinámicamente una interfaz gráfica para que el usuario introdujese los parámetros necesarios para realizar la invocación.

En la figura 10.27 se puede observar cómo los parámetros son introducidos en la interfaz generada dinámicamente:

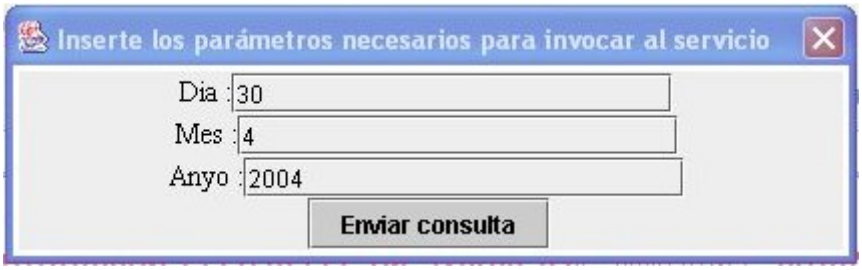


Figura 10.27 Parámetros de entrada en interfaz generada dinámicamente.

En ese momento, el agente cliente obtuvo con el parser el WSDL enlazado con el archivo *grounding* del servicio, lanzó el invocador dinámico y recogió los resultados de la invocación para mostrarlas en su interfaz gráfica, como se puede observar en la figura 10.28:

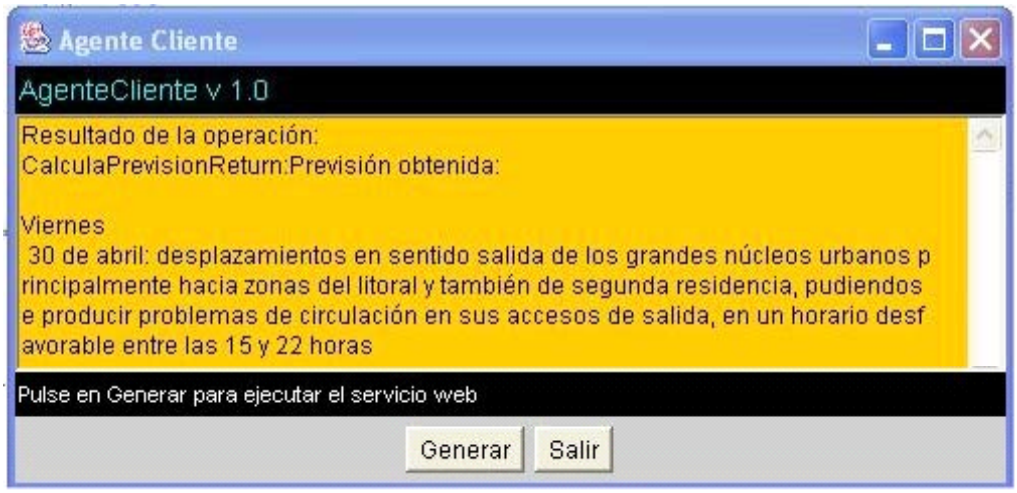


Figura 10.28 Resultado de la operación mostrado por agente cliente en la interfaz gráfica

10.5 CONCLUSIONES

Se ha analizado, diseñado, implementado y probado un sistema que integra un gran número de tecnologías emergentes como los SWS, ontologías y agentes, lo que constituye una aportación académica al desarrollo de la ingeniería de la web del futuro, la Web Semántica.

Las mayores dificultades en el desarrollo del sistema han sido el continuo cambio o aparición de nuevas especificaciones de las tecnologías de la Web Semántica, así como la carencia de documentación de las herramientas.

En este capítulo ha sido abordado el sistema en general, entendido como un sistema multiagente orientado a facilitar, asistir, y optimizar los procesos de anuncio, descubrimiento e invocación de SW relacionados con el área de información sobre tráfico vial.

La elección del modelo de intermediación denominado "*matchmaking*", ha llevado consigo la implementación de un sistema emparejador de SW, que toma como entrada las descripciones semánticas de las capacidades de los servicios.

Se han descrito los diferentes actores que intervienen en el sistema, así como una visión general de éste y las interacciones entre los diversos agentes que lo componen siguiendo determinados protocolos de comunicación establecidos por FIPA.

La plataforma final, permite que agentes clientes soliciten a agentes emparejador la búsqueda de SW acordes con las capacidades semánticas que están buscando, para después ser invocados. De igual forma, los servicios anunciados son representados por agentes proveedores dentro de la plataforma.

Se ha conseguido implementar un cliente genérico capaz de invocar dinámicamente cualquier servicio, sin necesidad de recompilar la interfaz en cada invocación. Necesario, debido al comportamiento de los servicios, ya que éstos aparecen y desaparecen dinámicamente en la plataforma y sería inviable tener preparado un cliente para cada uno de los posibles servicios que en algún momento determinado pueden estar disponibles.

Se ha propuesto una arquitectura en capas y se ha analizado cada una de ellas, así como sus interrelaciones. En cada capa se han descrito los diferentes elementos que la constituyen, distinguiendo a su vez aquellos elementos que pertenecen a la plataforma de agentes de aquéllos externos que interactúan con ella.

Se ha mostrado el diagrama de despliegue de los diferentes componentes que integran la arquitectura (con la inclusión de SW). Mediante este diagrama de despliegue, se ha puesto de manifiesto la naturaleza distribuida de sus elementos, y las interacciones entre cada una de las partes que forman el sistema.

Teniendo en cuenta la gran inmadurez y falta de estabilidad y control de compatibilidad hacia atrás de las herramientas disponibles actualmente para los desarrollos de la Web Semántica en sus herramientas, se ha tenido gran dificultad para conseguir una solución estable.

Las pruebas realizadas han permitido:

- Comprobar la integración del emparejador en la plataforma de agentes y su uso en un caso real.
- Mostrar como el sistema emparejador, y más concretamente la ejecución del algoritmo de emparejamiento propuesto, permitieron elegir correctamente el

servicio registrado que más se ajustaba al requerimiento (perfil de servicio) solicitado por el usuario a través de la interfaz.

- Evidenciar la coordinación de los agentes dentro de la plataforma, y el intercambio de mensajes llevados a cabo en respuesta a los diferentes eventos acontecidos.
- Observar, mediante el uso de herramientas de la plataforma la realización del ciclo completo desde la petición de búsqueda hasta la invocación, ejecución del servicio encontrado y la posterior entrega de información al usuario de la aplicación, mediante el seguimiento de los mensajes entre los agentes involucrados.

SECCIÓN V: DISCUSIÓN, CONCLUSIONES Y TRABAJO FUTURO

CAPÍTULO 11

RESULTADOS Y DISCUSIÓN

11.1 RESUMEN

En este capítulo se presenta un resumen de resultados tanto a nivel de aplicaciones, entendidas como contribuciones a la generación de un nuevo conocimiento, y por otra parte se describen otros resultados que se generaron a partir del proceso de investigación llevado a cabo. Por último se discuten los diferentes resultados obtenidos y las aportaciones realizadas. Finalmente se termina con las conclusiones obtenidas.

11.2 RESULTADOS Y APORTACIONES

Tomando como punto de partida el estudio del estado del arte, y el trabajo de investigación llevado a cabo en esta tesis doctoral, las principales aportaciones que se pueden extraer del trabajo son:

- Dado que hasta donde llegó la búsqueda y revisión bibliográfica de esta tesis, no se encontraron ontologías o vocabularios con valor semántico que aporten significado a los conceptos y sus relaciones, que se pudieran tomar como base en esta investigación, este vacío fue cubierto mediante la **construcción de una infraestructura ontológica cuyo dominio queda enmarcado en la información sobre tráfico vial**. Para delimitar el problema, se consideró la distribución y acceso a la información "previaje" y al manejo de "sucesos" principalmente. De la misma forma se consideraron distintas fuentes oficiales como las provenientes de la organización de estandarización ISO y Reglamentos Generales de Tráfico.

- Tomando como base el análisis de las diferentes metodologías y/o métodos utilizados en la construcción de ontologías, y los beneficios del uso de un modelo semántico formal, se ha propuesto una **extensión a las metodologías previamente existentes para plantear el proceso a seguir en la obtención de un modelo semántico formal a partir de un modelo de Entidad-Relación (ER)**. En la presente aproximación, se han integrado diferentes aspectos de las metodologías actuales así como aquéllos surgidos de lecciones aprendidas de la experiencia. Este método fue aplicado en la construcción de la infraestructura ontológica creada, y puede servir como guía de desarrollo en construcciones ontológicas similares.

- Se ha diseñado, implementado y evaluado, un **algoritmo de emparejamiento**

de SWS, que aprovecha al máximo las capacidades proporcionadas por la ontología de descripción de servicios y que constituye una mejora de los ya existentes. El principal punto de investigación se ha centrado en los diferentes grados de similitud que permiten emparejamientos flexibles, uso de parámetros no funcionales así como la introducción de diversos filtros, algunos de los cuales no habían sido considerados hasta la fecha, en trabajos de otros autores.

En este algoritmo de emparejamiento se amplió el rango de grados de similitud, principalmente con la consideración de nuevas relaciones en la taxonomía de la ontología de dominio utilizada como soporte en el emparejamiento. Paolucci et al. (literatura de referencia) no distinguen el **grado fraternal o de hermanos**, por lo que su propuesta será incapaz de devolver un perfil correspondiente a un servicio si el que más se asemeja al buscado posee parámetros que pertenecen a esta relación. Se ha evidenciado mediante pruebas funcionales que el algoritmo propuesto sí es capaz de localizar un servicio provisto, cuyos parámetros mantienen una relación fraternal con aquellos conceptos en los que se basan los parámetros del perfil de un servicio requerido. Otro tipo de pruebas funcionales determinaron la validez de considerar esta clase de relaciones en lugar de relaciones de subsumción con distancias muy grandes en el árbol taxonómico de conceptos. La consideración de este nuevo grado de similitud ha hecho posible encontrar servicios que podían no haber sido considerados bajo las mismas condiciones por algoritmos previos e incluso encontrar aquéllos que son más similares a los requeridos.
Se han analizado diferentes factores en este tipo de relación, como por ejemplo, el **número y tipo de restricciones comunes** sobre las propiedades, aunque como se describe en la memoria, una vez estudiados todos los casos, se ha podido constatar la gran dificultad que lleva consigo la distinción de las restricciones que puede tener definidas el concepto proporcionado por el cliente. Este análisis debe ser llevado a cabo, antes del comienzo de la fase de emparejamiento, lo cual a su vez ralentiza el proceso de búsqueda. Por tanto, el beneficio que se puede tener en eficacia con esta mejora, respecto la gran complejidad y pérdida en velocidad de ejecución, hizo que esta segunda posibilidad de mejora de la fase de emparejamiento del grado hermano fuera descartada.

El uso de otras decisiones como el establecimiento de filtrados anteriores a la ejecución del algoritmo, así como el uso de otros parámetros no funcionales, mejoraron considerablemente el coste y la elección del servicio más idóneo respectivamente. La decisión de abortar el proceso comparativo con el resto de parámetros cuando no haya semejanza entre parámetros de salida (*outputs*) pertenecientes al perfil del servicio requerido y cualquiera de los pertenecientes a los proveedores, supone también una mejora que debe ser resaltada.

- El planteamiento del uso de SWS como medio de acceso a la información requerida por un usuario, establece que los requerimientos o consultas de éste, sean especificados mediante perfiles de servicio. El usuario deberá construir o especificar el perfil (mediante la subontología profile) que necesite teniendo en cuenta las ontologías base o de conceptos que pertenecen al dominio de búsqueda. La necesidad de construir el perfile mediante un lenguaje que en principio debería ignorar, así como la interacción con las ontologías de dominio

para la especificación de los parámetros descriptivos del servicio deberían ser transparentes al usuario. De la misma manera, la entrega del perfil de búsqueda como entrada en los procesos de emparejamiento llevados a cabo posteriormente en una plataforma de agentes, no deben de ser opacos al usuario.
Por las razones expuestas, ha sido propuesto un **entorno de desarrollo de edición de perfiles de servicio de información de tráfico** específicos (OntoServices) que permite integrar la visualización y verificación de consistencia de ontologías que definan los conceptos con los cuales interactúa un servicio Web. La doble funcionalidad de la herramienta logra resultados positivos, principalmente por dos razones de peso: la primera de ellas es que es posible ajustar, modificar, actualizar y verificar cualquiera de las ontologías usada como soporte en la construcción de los perfiles de servicio, lo que permitirá realizar búsquedas más exactas. La segunda de las razones es la posibilidad de buscar servicios mediante la interacción con el sistema prototipo, lo que permite simplificar el proceso de búsqueda y asistir al cliente en la edición de tediosos perfiles que describan las capacidades del servicio requerido.

- Se ha hecho uso de **técnicas de gestión y seguridad de perfiles de usuario** en la búsqueda de servicios, de tal manera que mediante éstos, se ha logrado optimizar este proceso. En la búsqueda de un servicio, una de las fases más costosas es el proceso de "búsqueda de pareja", en el cual se trata de localizar los servicios realizando un emparejamiento del perfil generado, con los perfiles suministrados por los proveedores de servicios. El uso de una caché que almacene las últimas consultas realizadas permite devolver con rapidez los servicios buscados sin necesidad de ejecutar el algoritmo, incrementando la velocidad del sistema, ya que probablemente los usuarios buscarán un reducido número de tipos de servicios. Para ello, se ha diseñado una arquitectura de almacenamiento remoto que permite que un usuario se pueda conectar desde cualquier lugar al servidor de perfiles y cargar las últimas búsquedas realizadas. De esta forma, un usuario registrado puede beneficiarse del uso de caches que almacenen sus últimas consultas evitando realizar búsquedas innecesarias.

- Se ha expuesto un **marco de trabajo para la conversión de portales Web convencionales de información de tráfico en SWS** de manera que la información aportada pueda ser almacenada con la adición de significado consiguiendo a su vez, potenciar nuevas capacidades que en un principio no existían. La ventaja de hacer uso de esta aportación reside en el gran volumen de información que aparece distribuida por los diferentes portales Web de administraciones o entidades privadas que la ofrecen, y la dificultad y coste para reconvertir este tipo de escenarios. El uso de este marco permitirá solventar este problema aunque en algunas fases del proceso, siga siendo necesaria la intervención por parte del desarrollador de SWS.

Se han mostrado una serie de propuestas que permiten crear y manejar SWS de información de tráfico vial:

- Ha sido propuesto un nuevo **parámetro no funcional**, denominado **valor añadido**, no incluido por la coalición OWL-S en sus ontologías de descripción de servicios. La adición de este nuevo parámetro supone aumentar las

capacidades de descripción de un servicio y por tanto la optimización del proceso de descubrimiento de éstos, en cuanto a que el cliente no solamente obtiene aquel servicio que está buscando, sino otra serie de servicios que complementan al primero, y los cuales pudieran ser necesitados en búsquedas posteriores. De igual manera, desde el punto de vista de proveedor se facilita el anuncio de servicios que complementan o pudieran interesar al servicio anunciado. La adición de este nuevo parámetro aporta una mayor expresividad en cuanto a la caracterización y relación entre servicios.

- Propuesta de una **ontología de categorización de servicios de tráfico**. (*Jerarquía de perfiles*) distinta a las hasta ahora utilizadas como NAICS y UNSPSC. Esto ha constituido mejoras en dos aspectos principalmente: por una parte, la descripción de los servicios que de esta manera serán clasificados bajo esta nueva taxonomía, y por otra parte, su posterior búsqueda mediante filtros que tengan en cuenta la categoría a la que pertenecen. La creación de esta ontología de clasificación de servicios ha cubierto este tipo de infraestructura ontológica a la vez que puede suponer una base sólida en el uso de servicios en los ITS. El fundamento principal de la conclusión anterior reside en el hecho de haber tomado como base el trabajo de estandarización llevado a cabo por el WG TC 204 de ISO.

- Propuesta para **medir la calidad de servicio en servicios de tráfico**. En la especificación de OWL-S se incluyen constructores para expresar parámetros relativos a la calidad de servicio (QoS), sin embargo, no proveen de un conjunto detallado de clases y propiedades que permitan representar esta métrica. Tomando como punto de partida anteriores trabajos y viendo la dificultad de aplicar dichos modelos a los SW de tráfico, se creó un modelo diferente que tuviera en cuenta las características propias de los portales de información de tráfico objeto de estudio. Debido a la no disponibilidad de ninguna escala de calidad inicial para este tipo de servicios, se ha propuesto una **métrica basada en factores de calidad**. De esta manera, cualquier servicio provisto podrá especificar su grado correspondiente, el cual será considerado en la búsqueda si el cliente así lo requiere.

- Se ha presentado y validado una **arquitectura de integración de SWS** de información sobre tráfico vial mediante la construcción de un prototipo software capaz de hacer uso del algoritmo de emparejamiento propuesto, así como la implementación de éste último como un componente dentro de un sistema multiagente que permita la publicación, descubrimiento e invocación de los servicios. En el ámbito de recuperación de información o búsqueda podemos encontrar diferentes tipos de herramientas, destacando aquéllos que hacen uso de agentes inteligentes. El prototipo desarrollado, denominado ESWIT, corresponde a este tipo, y su función principal recae en la búsqueda de información a través de SW, mediante el emparejamiento de éstos, haciendo uso de sus capacidades descritas semánticamente mediante las ontologías de tráfico construidas.

- Para el algoritmo de emparejamiento, se ha construido un **parser que toma como entrada los perfiles especificados mediante DAML-S 0.9 u OWL-S 1.0**

y extrae cada una de las capacidades que describen al servicio (el valor de sus parámetros). La extracción de esta información es necesaria para poder completar la información que poseen las consultas encargadas del emparejamiento semántico de los parámetros de la petición del cliente y los anuncios de los proveedores. El emparejador debe acceder a cada uno de los parámetros, tanto funcionales como no funcionales.

- Para lograr la invocación dinámica de los servicios, una vez descubiertos por el emparejador, fue necesario acceder al archivo grounding asociado al profile del servicio anunciado con el que la petición del cliente hubiese sido emparejada, para extraer de él la descripción WSDL relacionada. El proceso de obtención de la información a partir de este fichero inicial, mediante el seguimiento de los enlaces entre los distintos ficheros que describen al servicio (necesarios para la invocación) estableció la construcción de un nuevo **parser**.

- Además, se ha proporcionado la **descripción semántica de diferentes servicios de tráfico**, como también el tratamiento semántico de la información o recursos que éstos poseen gracias a las ontologías que también se han descrito en esta tesis.

- Se han desarrollado ***scripts* que extraen información de la Web** tomando como soporte las ontologías creadas, mantienendo de esta forma actualizada la información proveniente de ese entorno dinámico y evolutivo como es Internet. En aquellos portales donde la introducción de valores en sus formularios era necesaria para la obtención de resultados, previamente se averiguó cuales eran estos parámetros o variables que determinaban la consulta, para ser incorporados en las llamadas a dichos portales y de esta manera extraer la información en ellos contenida. Estos scripts forman el esqueleto de los agentes ***Wrappers*** implementados en la plataforma.

En la figura 11.1 se puede observar las principales aportaciones y resultados de este trabajo, tanto en el área de Tráfico como en la Computación.

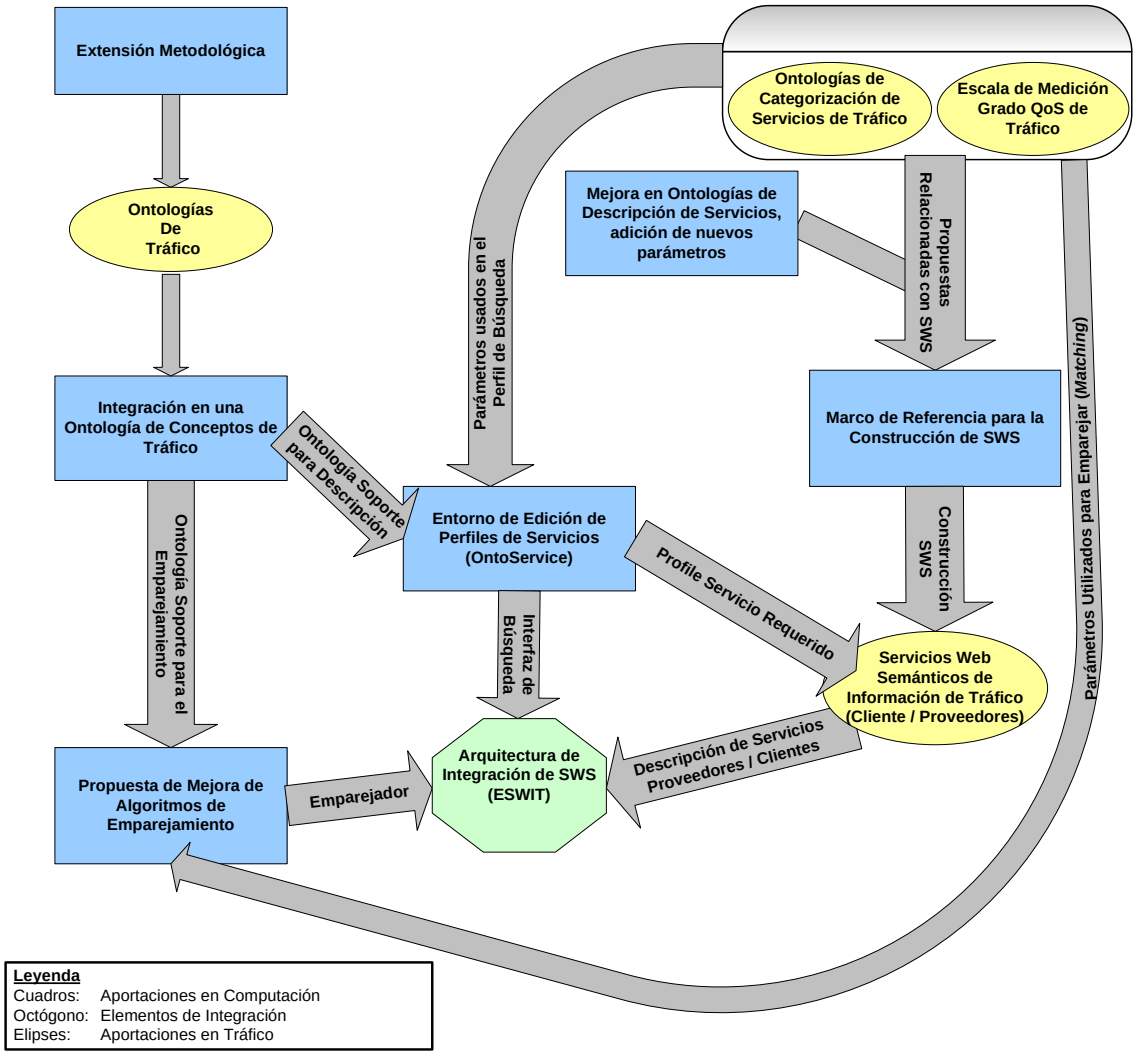


Figura 11.1 Aportaciones y Resultados

11.3 DISCUSIÓN

En el dominio de aplicación de los ITS, la posibilidad de hacer uso de SWS y de sistemas multiagente se convierte en algo especialmente útil, si pensamos por ejemplo, en la conjunción de información proveniente de fuentes heterogéneas en cuanto a contenido y origen. La composición de servicios y el tratamiento del flujo de datos, pueden dar como resultado la asistencia parcial y en el mejor de los casos completa al viajero.

Como ejemplo basado en una situación real, partamos del siguiente caso: la elección de un determinado itinerario de viaje por parte de un usuario, la cual puede estar afectada en el tiempo por determinados factores como las condiciones meteorológicas o las variaciones en el estado de la carretera ocasionadas por diversos factores.

En la actualidad los navegadores de conducción tienen gran dificultad para actualizarse cuando ocurren cambios en el contexto (localización del usuario, condiciones

ambientales, dispositivo de acceso a la información y aspectos sociales) del usuario en algún punto intermedio de la ruta dada inicialmente. El uso de una aproximación de forma automática y transparente al usuario en la obtención de la ruta optima (mediante un servicio calculador de itinerarios y modelos matemáticos), permitiría que el sistema determine si en algún punto de la ruta y en un determinado instante, la conducción pueda verse afectada, mediante la consulta a otros servicios como proveedores de incidencias o servicios de información meteorológica. La gestión automática de la información al usuario, basada en la existencia de planes de gestión de trafico y con la consideración de las variaciones en ese **contexto** y **ubicuidad**, deberían permitir al usuario, la interacción dinámica en el procesamiento de su ruta en el lugar donde se encuentre (en general con el contexto asociado en un instante determinado).

En esta tesis, se han conseguido obtener resultados parciales a esta exposición, ya que la posibilidad de describir cualquier tipo de servicio y la de elección de aquél que más se ajuste por sus características a lo que requiere el usuario (en este caso, cálculo de itinerarios) y de aquellos servicios complementarios (servicio de incidencias, y servicio de meteorología), e incluso la posible composición de éstos, queda ampliamente cubierto como resultado de esta investigación. Sin embargo, la incorporación de un servicio de gestión automática de las variaciones de contexto, junto con la existencia de composición automática de servicios se constituyen en alternativas de solución a tener en cuenta para el tratamiento de esta problemática.

A partir de las pruebas realizadas, se pudo observar como hecho más relevante, que la utilización del algoritmo propuesto hizo posible encontrar servicios basados en similitudes de tipo hermano, las cuales no habían sido estudiadas anteriormente. Aunque este tipo de similitud pudo no haberse tenido en cuenta debido a que en el caso de encontrarse similitudes entre nodos hermanos, posiblemente sería recomendable, desde el punto de vista de diseño, crear un nodo padre intermedio que agrupase las características comunes de los hijos. Sin embargo, el algoritmo propuesto las tiene en cuenta partiendo de la premisa de que es posible que este tipo de similitudes no hayan sido consideradas por el proveedor que definió semánticamente estas relaciones, bien sea por consideraciones de rendimiento (para evitar un grado de profundización demasiado exhaustivo) o simplemente por no haber detectado este tipo de relaciones en la fase de diseño. Como consecuencia de lo anterior, el descubrimiento de servicios basado en algoritmos convencionales dejaría de producir resultados que están incluidos en el rango requerido de respuesta.
Por otra parte, la elección de ontologías en las que no hay conceptos con definiciones completas (condiciones necesarias y suficientes para ser miembro de una clase) puede dar lugar a este tipo de situaciones, ya que los razonadores serán incapaces de establecer clasificaciones de manera automática. En estos casos, el peso de la construcción de una buena y apropiada ontología recaerá en el desarrollador de ésta. Por el contrario, el diseño de ontologías con conceptos definidos permitirá solventar este tipo de problema aunque incrementará notablemente el coste computacional. Por tanto la decisión de incorporar este grado de similitud se ve ampliamente respaldada.

Actualmente, hasta donde llegó el proceso de revisión de esta tesis, no se han encontrado servicios de información de tráfico vial en la Web, y por tanto SWS de esta clase tampoco. El marco de trabajo propuesto para la conversión o adaptación de portales Web a este tipo de servicios, se simplificaría bastante si los portales Web

inicialmente ya hicieran uso de anotaciones semánticas en los lenguajes de marcado de presentación, se definieran nuevos portales (o SWS) con una semántica definida o por último, hicieran uso de SW no semánticos que se pudieran tomar como base.

La transición desde portales Web a una Web de Servicios actualmente solo está siendo asumida por las grandes compañías debido a diversos factores como la falta de una visión clara en la obtención de beneficios de negocio, falta de formación técnica, resistencia al cambio y costo principalmente. Esto está suponiendo una limitación en el posicionamiento y expansión de este tipo de tecnologías.

La labor que está realizando el grupo de trabajo TC204 de ISO, sirve como marco de referencia para el desarrollo y despliegue de aplicaciones ITS basado en la definición de los requisitos y de los estándares apropiados y necesarios. Aunque la consideración de establecer como requerimiento el uso de XML Schema y de SW y la posterior definición de reglas para ser utilizadas por los desarrolladores de ITS, es un paso importante en la estandarización y como consecuencia en la interoperabilidad entre sistemas, todavía no se han planteado la introducción de la semántica tal y como se ha propuesto en esta tesis.

En el desarrollo de este trabajo, he podido constatar la importancia de establecer puntos de parada obligatoria en la investigación para reflexionar sobre lo que en cada momento se estaba debatiendo, antes de dejarse llevar por las distintas corrientes tecnológicas que evolucionan rápidamente sin ni tan siquiera asentarse en sus respectivas áreas.

La aparición de diferentes tecnologías y como consecuencia de ello, diferentes herramientas, ha motivado hasta la fecha que los resultados obtenidos de su utilización no sean totalmente compatibles entre ellas. Hay que tener presente que algunos resultados sirven como parámetros de entrada en otras herramientas, como un eslabón más en la consecución del objetivo perseguido. Generalmente, la interoperabilidad de las aplicaciones que trabajan con ontologías (construcción, razonamiento, repositorios etc.) no es mala, pero siempre hay pequeños detalles que se han de modificar para que una ontología sea totalmente portable de un entorno a otro. Quizás, la estandarización por parte de organizaciones como ISO, OASIS y W3C ayude en este sentido. Por ejemplo, Sesame permite guardar ontologías codificadas en RDF en la base de datos utilizada como repositorio, pero la mayoría de editores de ontologías aunque son capaces de crear documentos RDF u OWL, sin embargo no todos ellos son válidos para Sesame.

Si bien la Web es un gran espacio de información, no puede llegar a ser considerada una base de datos global, debido a que no todos los documentos presentados en ella tienen asociada una estructura o semántica. Por tanto, como una capa intermedia para construir la WS, se requiere la realización de anotaciones semánticas de los contenidos y de las relaciones existentes en ella.

Existen muy pocos trabajos reportados en la literatura científica sobre el tratamiento multilingüe de ontologías y discusiones sobre si el conocimiento base existente en la red debe estar definitivamente estructurado en inglés, siendo los demás idiomas una simple capa adicional. Por otra parte, hay pocos trabajos sobre la forma de mapear el conocimiento entre el inglés y otras lenguas, lo que posiblemente puede estar

causado por el liderazgo de investigadores anglosajones en el desarrollo de la WS, lo que conlleva a que no haya un gran interés en el desarrollo de este mapeo. Resulta importante destacar la necesidad de analizar la forma de evitar la pérdida de peculiaridades de los lenguajes y culturas, así como la forma de preservar la cultura digital para el futuro.

Consorcios, foros u organizaciones como el W3C tiene gran importancia en el proceso de ayuda en la construcción de una Web más organizada y útil para toda la sociedad. Las universidades deberían participar más activamente en este tipo de consorcios donde hasta ahora la presencia es mayoritariamente de empresas, y donde si es que se puede decir que ha habido algún aporte del mundo universitario éste ha sido principalmente a través de miembros vinculados a compañías de la industria Informática.
Es importante que la Universidad participe en este proceso de construcción de la Web del mañana y dé su aporte desde la "Universalidad" de ésta, como un auditor de la sociedad en lo referente a lo que hacen las empresas que dominan el mercado.
Esto, como lo que comenté anteriormente, es igualmente válido en relación con otros foros u organizaciones como la de ISO.
Por eso pienso que mi trabajo en esta tesis doctoral constituye un pequeño grano de arena en el proceso de construcción, evaluación, y análisis crítico de iniciativas como la Web Semántica y más aún su aplicación al dominio de los ITS.

11.4 CONCLUSIONES

A la vista del trabajo realizado en esta tesis, se ha podido constatar que la Web Semántica está todavía en un proceso de inmadurez, y que por lo tanto queda aún un largo camino por recorrer. Necesita de aplicaciones reales e industriales que muestren todo su potencial y no simplemente ejemplos donde se llegue a lo más a usar RDF o cualquier lenguaje basado en él, sin tener en cuenta el resto de capas que conforman su arquitectura. La conjunción de esfuerzos y resultados por parte de las diferentes comunidades completarán el proyecto. El uso de tecnologías complementarías más consolidadas como los SW permitirán cubrir parte de los objetivos. Debido a esta consolidación y al futuro que se prevé para este tipo de tecnologías, sobre todo en el área B2B, se puede diagnosticar que la integración de la WS y los SW suponen un pequeño escalón en el proceso de alcanzar la visión inicial de WS y el cual puede ser cubierto sin esperar la completa realización de ésta.

La aplicación de los SWS (producto de esta integración) en los ITS, me ha permitido ver con claridad las ventajas de su uso en cuanto a la consecución de los objetivos iniciales marcados, pero también las grandes dificultades que conlleva hacer uso de tecnologías emergentes no consolidadas. El principal escollo a salvar ha sido la constante evolución de las tecnologías, el gran abanico de posibilidades para la ejecución de cada uno de los objetivos parciales y sobretodo la falta de sensibilización que a fecha de hoy, poseen los distintos organismos relacionados con el tráfico vial, y ajenos a la gran revolución que está sufriendo la Web. Sin embargo, gracias a la interacción del mundo académico con el de las administraciones públicas, auguro que en un futuro próximo la aplicación de técnicas como las tratadas en esta tesis serán adoptadas en los ITS, al igual que ocurrió con tecnologías predecesoras como XML.

En lo referente al dominio de aplicación de información de tráfico ofrecida al viajero (TIS), el trabajo expuesto en esta tesis, supone una iniciativa totalmente novedosa en cuanto a su aplicación, puesto que las líneas actuales no contemplan el uso de la semántica como alternativa en la distribución o publicación de la información. Actualmente los proveedores de servicios de tráfico así como los CIT, han considerado el uso de lenguajes de marcado XML para sus dominios específicos así como tecnologías de servicios tradicionales, sin embargo, las ventajas del uso de una semántica común en los vocabularios permitiría no solo la obtención de los mismos resultados sino un incremento en cuanto a eficiencia por las razones expuestas a lo largo de la memoria.

Las diferentes pruebas de funcionalidad realizadas al prototipo desarrollado han permitido verificar que el objetivo general marcado ha sido satisfecho. Principalmente, se ha constatado como ha sido posible la realización del ciclo completo desde la petición de búsqueda por parte de un cliente hasta la invocación y ejecución del servicio encontrado y la posterior entrega de información al usuario de la aplicación, gracias al seguimiento de los mensajes entre los agentes involucrados. Mediante estas pruebas ha sido posible comprobar la integración de todos los elementos propuestos en esta tesis, especialmente las ontologías creadas, así como el algoritmo de emparejamiento.

El algoritmo presentado, supone un extensión y mejora de anteriores propuestas enmarcadas en la misma línea de investigación, ya que tras el análisis de los resultados obtenidos de las pruebas realizadas se puede concluir que en determinadas ocasiones este algoritmo propuesto ha conseguido obtener servicios que más se ajustaban por sus características a los requerimientos de un determinado cliente, que los obtenidos por otras aproximaciones ampliamente referenciadas, aunque a veces esto haya repercutido en un incremento en el coste computacional.

Se ha podido comprobar que el hecho de hacer uso en un principio de ontologías en castellano no supuso un agravio en cuanto el desarrollo del prototipo y además se pudo comprobar el *mapeo ontológico* entre ontologías de diferentes idiomas.

En está investigación se ha constatado la gran inmadurez de estándares y de plataformas tecnológicas en cuanto a los SW, así como la ausencia de un metodología común y normas preestablecidas para el diseño de servicios. No obstante, los SW crearán nuevas oportunidades de negocio, aportando como beneficios el poder compartir datos, aplicaciones y procesos entre proveedores y clientes. Es de resaltar la importancia de los SW en la comunicación de las aplicaciones y los Sistemas de Información sin importar el lenguaje en el que fueron desarrollados o la plataforma en la que se ejecuten.

La integración de ontologías de tráfico vial ha contribuido no solo a la consecución de los objetivos que inicialmente fueron tomados, sino a la certificación de que el uso de la semántica puede ser aplicado en cualquier entorno o dominio.

CAPÍTULO 12

TRABAJO FUTURO

A continuación se describen las posibles extensiones del trabajo, que establecen nuevas líneas abiertas que pueden ser objeto de desarrollo en un futuro próximo:

- Ampliación de la Ontología de QoS con la adición de nuevos factores, así como la introducción de elementos de cuantificación que permitan establecer de forma automática las mediciones de calidad, sin la necesidad actual de entidades ajenas al sistema. Esta labor sería llevada a cabo mediante agentes, consiguiendo de esta forma un alto grado de objetividad.

- El sistema multiagente y el algoritmo de emparejamiento están orientados a la descripción y localización de SW de información de tráfico vial, por lo que una posible tarea podría ser la ampliación del rango de categorías de servicios disponibles, para abarcar todo el rango de servicios de información al viajero como localización, restaurantes, talleres, gasolineras, hospitales etc., e incluso cualquier otro tipo de servicio fuera del ámbito del tráfico.

- Considerando el algoritmo propuesto, éstas son algunas posibles mejoras que se pueden realizar:

    - Uso de técnicas de recuperación de información como TF-IDF reduciendo el dominio de aplicación de estas técnicas, para pasar de documentos completos a simplemente contenidos de ciertos parámetros etiquetados. Por ejemplo, el uso de esta técnica podría ser usada para determinar la frecuencia de aparición de ciertos elementos considerados clave, en aquellos parámetros que describen el servicio mediante el uso de lenguaje natural (p.e. parámetro textdescription).

    - Aplicar diferentes grados de similitud en los parámetros no funcionales correspondientes a tipo concepto, como pueda ser el radio geográfico. Por ejemplo, esta idea permitiría determinar aquellos servicios cuya área de aplicación no solamente coinciden exactamente con la especificación de un lugar geográfico dado, sino que puede incluir o pertenece al radio especificado por el cliente (p.e. ofrecer servicios en Cataluña, cuando inicialmente se requerían servicios en la provincia de Barcelona).

    - Hacer uso de reglas que permitan condicionar las búsquedas atendiendo a las precondiciones que pudieran formar parte de las descripciones de los servicios. De esta forma solo serían seleccionados aquellos servicios que realmente puedan ser ejecutados si previamente se cumplen sus

condiciones preestablecidas.

- En cuanto a una optimización en el uso de perfiles de usuario, se podría considerar el uso de la información aportada por los proveedores al anunciar sus servicios, para determinar si pudiera ser considerada de interés, atendiendo a la especificación del perfil de algún usuario. En el caso de que así fuera, éste podría ser avisado mediante por ejemplo, un servicio de alertas. Con esto se obtendrían servicios persistentes. El uso de parámetros de tiempo que permitan validar la caducidad de los requerimientos de los usuarios sería de igual forma recomendable.

- Adaptación del entorno de desarrollo para que éste puede ser utilizado no solo por clientes que realizan peticiones sino también por proveedores para anunciar sus servicios. Esto llevaría consigo la automatización o la asistencia en la creación del resto de subontologías que describen al servicio, y la generación de procesos compuestos.

- Un aspecto importe para ser considerado como línea de investigación futura es la aplicabilidad del factor multimodal (diferentes formas de acceso) a este tipo de sistemas. Esto fue descrito como uno de los requisitos expuestos en la sección 6.2 de la presente memoria. Como objetivos concretos, enmarcados en esta línea, se podría destacar el hecho de que cualquier servicio de Web, pueda ser consultado (buscado previamente) mediante cualquier interfaz gráfica e incluso de voz. Para lograr este objetivo se debería adaptar la infraestructura multiagente para permitir el acceso desde diferentes dispositivos (p.e. móviles), que permitan establecer comunicaciones ubicuas.

- En vistas a la internacionalización que sufren las ontologías y por tanto al gran auge del idioma anglosajón en este sentido, sería conveniente el desarrollo de propuestas de modelado y herramientas que tradujeran las ontologías en diferentes idiomas, estableciendo un mapeado entre ellas. De esta forma, el desarrollador de ontologías solo debería preocuparse de cómo describir los términos en su propia lengua ante las limitaciones de expresión que puedan acontecer de no hacerlo así y a su vez, cualquier sistema o usuario podría acceder a la información en el idioma de interés.

- Mejoras en cuanto a la composición de las consultas realizadas, debido a que en esta tesis se plantea la necesidad de la creación de perfiles de servicio como único medio de requerir su búsqueda. No obstante, se debería considerar el análisis de consultas expresadas mediante lenguaje natural o con reconocimiento biométrico, para que de forma totalmente transparente al usuario, este tipo de consultas puedan ser descompuestas y se permita su orientación a uno o varios servicios que pudieran ser requeridos.

- Enlazando con la anterior propuesta, el sistema debería de ser capaz de componer servicios dinámicamente para ser ofrecidos al cliente como un único servicio Web. El uso de agentes denominados ontológicos permitirían obtener las ontologías adecuadas para efectuar este proceso.

- Construcción de herramientas que automaticen las tareas de creación y despliegue de SWS de cualquier índole a partir de sitios Web convencionales. En este proceso todavía hay una fuerte dependencia de la interacción con el desarrollador.

- Proponer y validar metodologías, así como herramientas que hagan uso de éstas, que permitan modelar primero conceptualmente los distintos elementos que conforman la descripción semántica de un servicio, para posteriormente obtener un modelo formal mediante el cual sea posible comprobar su consistencia y posteriormente exportar al lenguaje de descripción que más se adecue a nuestros propósitos.

- Buscar otras alternativas de modelado de SWS, partiendo de las ya existentes para sus principales componentes: Modelado de SMA, Ingeniería Ontológica, Sistemas de Información Web etc.

- Determinar el tratamiento de la semántica oculta existente en portales dinámicos. Su descubrimiento y procesamiento.

- Construcción de un sistema prototipo que englobe todas las capas que conforman la Web Semántica, principalmente aquéllas que no están consolidadas en este momento como son las de Proof y Trust. Esto permitiría obtener un marco de seguridad y confianza en el uso de la información obtenida a partir de los SWS de tráfico creados y la realización completa de esta nueva concepción de la Web.

- Esta tesis ha tomado como punto de partida el uso de la Web Semántica y su aplicación a los ITS. Como línea de investigación futura, sería interesante determinar si la resolución de los problemas planteados y los objetivos marcados podrían haber sido satisfechos sin haberlo enfocado desde este punto de vista. El establecimiento de métricas y un análisis de resultados permitirían decantar la balanza a favor de una u otra elección.

GLOSARIO

API	Application Programming Interface
ATIS	Advanced Traveler Information Systems
AUML	Agent Unified Modeling Language
B2B	Bussines-to-Bussiness
B2C	Bussines-to-Consumer
CGI	Common Gateway Interface
CMU	Carnegie Mellon University
DAML	DARPA Agent Markup Language
DAML-S	Daml Services
DARPA	Defense Advanced Research Projects Agency
DATEX	DATa EXchange
DGT	Dirección General de Tráfico
DL	Description Logics
DOM	Document Object Model
DQL	DAML Query Language
DTD	Document Type Definition
ebXML	Electronic Bussines eXtensible Markup Language
EER	Extended Entity Relation
FaCT	Fast Classification of Terminologies
FIPA	Foundation Intelligent Physical Agent
FO	Frame Ontology
FOL	First Order Logic
GPRS	General Packet Radio Service
GSM	Global System for Mobile
HTML	Hypertext Markup Language
IA	Inteligencia Artificial
IO	Inputs, Outputs
IOPE	Inputs, Outputs, Preconditions, Effects
ISO	International Organization for Standardization
ITS	Intelligent Transportation Systems
JADE	Jave Agent DEvelopment framework
JSP	Java Servlet Pages
JVM	Java Virtual Machine
KAON	Karlsruhe Ontology
KB	Knowledge Base
KIF	Knowledge Interchange Format
MIP	Multi-Annual Indicative Programme
NAICS	North American Industry Classification System
OCML	Operational Conceptual Modeling Language
ODE	Ontology Design Environment

OIL	Ontology Inference Layer
OKBC	Open Knowledge Base Connectivity
OWL	Web Ontology Language
OWL-S	OWL Services
PHP	Hypertext Preprocessor
PMV	Panel de Mensajes Variables
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDQL	RDF Data Query Language
RDS-TMC	Radio Data System-Traffic Message Channel
RNE	Radio Nacional de España
RPC	Remote Procedure Call
RQL	RDF Query Language
RuleML	Rule Markup Language
SAE	Society of Automotive Engineers
SeBOR	Sesame BOR
SCT	Servei Català de Trànsit
SeRQL	Sesame RQL
SGML	Standard Generalized Markup Language
SHOE	Simple HTML Ontology Extensions
SMA	Sistema Multiagente
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SW	Servicio Web
SWRL	Semantic Web Rule Language
SWS	Servicios Web Semánticos
TF-IDF	Term Frecuency-Inverse Document Frecuency
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
UNSPSC	United Nations Standar Products and Services Code
URI	Uniform Resource Identifier
URL	Uniform Resource Locators
W3C	World Wide Web Consortium
WML	Wireless Markup language
WS	Web Semántica
WSDL	Web Services Description Language
WWW	World Wide Web
XML	eXtensible Markup Language
XOL	XML-based Ontology Exchange Language
XSL	eXtensible Style Language
XSLD	XML Schema Definition Language
XSLT	XSL Transformation

BIBLIOGRAFÍA

[Abe01] Abela C. "DAML enabled Web Services and Agents in the Semantic Web", Thesis. CSAI Department University of Malta. 2001.

[Ale01] Alexaki, S.; Christophides, V.; Karvounarakis, G.; Plexousakis D.and Tolle K. "The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases". 2nd International Workshop on the Semantic Web (SemWeb'01), in conjunction with Tenth International WWW Conference (WWW10), pp. 1-13, Hong Kong. Mayo 2001

[Ali02] Al-Ali, Rashid, et. al. "Grid Service Discovery Using QoS Properties" Computing and Informatics, vol.21, pages 363-382. 2002.

[Alm02] Almaer, D. "Creating Web Services with Apache Axis". Disponible en: http://www.onjava.com/lpt/a/1578. 2002

[A-Match04] A-Match: Disponible en: http://www-2.cs.cmu.edu/~softagents/a-match/index.html Última visita noviembre 2003.

[Ank02] The DAML Services Coalition: Ankolenkar, Anupriya; Bustein, Mark; Hobbs Jerry R; Lassila, Ora; Martin, David L.; McDermott, Drew; McIlraith, Sheila A., Narayanan Srini; Paolucci, Massimo; Payne, Terry R. and Sycara Katia. "DAML-S: Web Service Description for the semantic Web". Appeared in The First International Semantic Web Conference (ISWC), 2002.

[Apache04] The Jakarta Site – Apache Tomcat. Disponible en: http://jakarta.apache.org/tomcat/ Última vez accedido el día 06/07/2004.

[Ari00] Arisha, K.; Eiter, T.; Kraus, S; Ozcan, F.; Ross, R. and Subrahmanian V.S. "Impact: Interactive Maryland platform for agents collaborating together". IEEE Intelligent Systems, 14(2), 2000.

[Baa03] Baader, F; McGuiness, D; Nardi, D and Patel-Schneider, P. "The Description Logic Handbook Theory, Implementation and Applications". Published January 2003 ISBN: 0521781760.

[Bal03] Balke Wolf-Tilo, Wagner Matthias. "Towards Personalized Selection of Web Services" en The Twelfth International WWW Conference en Budapest, mayo 2003.

[Ban03] Bansal, Sharad and Vidal, Jose. "Matchmaking of web services based on the Daml-s service model" en AAMAS 2003.

[BATIK04] Batik SVG Toolkit. Disponible en: http://xml.apache.org/batik/ Última visita enero de 2004.

[Bay97] Bayardo R.J. et al., "InfoSleuth: Agent-based semantic integration of information in open and dynamic environments, in Proceedings of the ACM SIGMOD

International Conference on Management of Data, volume 26,2, pp. 195–206, New York, (1997). ACM Press.

[Bell03] Bellifemine, F.; Caire, G.; Poggi, A. and Rimassa G., "JADE a White Paper". Exp- Volume 3 nº 3 – September 2003. Disponible en http://exp.telecomitalialab.com/upload/articoli/V03N03Art01.pdf

[Ber96] Bernaras, Laresgoiti I. and Corera J. Building and Reusing Ontologies for Electrical network Applications. Proceedings of the 12th ECAI", 1996. Pages: 298-302

[Ber98] Berners-Lee, Tim. Semantic Web Road map. Disponible en: http://www.w3.org/DesignIssues/Semantic.html. 1998.

[Ber01] Berners-Lee, Tim; Hendler, James and Lassila, Ora. "The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities", Scientific American. 2001.

[Ber04] Berners Lee, Tim W3C 2004 Talks. Disponible en: http://www.w3.org/2004/Talks/0412-RDF-functions/slide4-0.html

[Bla96] Blaze, M.; Feigenbaum, J. and Lacy J. "Descentralized Trust Management", in IEEE Conference on Security and Privacy, 1996, Oakland California, USA, Disponible en: http://www.crypto.com/papers/policymaker.pdf

[Bor97] Borst, W.N., "Construction of Engineering Ontologies". University of Tweenty. Enschede, NL- Centre for Telematica and Information Technology. (1997).

[BOR02] Simov, Kiril and Jordanov, Stanislav. "BOR: a Pragmatic DAML+OIL Reasoner"; Deliverable 40, On-To-Knowledge project. Disponible en http://www.ontotext.com/BOR, Junio 2002.

[Borg03] Borgida, A.; Lenzerini, M. and Rosati, R., Capítulo "Description Logics for Databases", The Description Logic Handbook, Theory, Implementation and Applications.

[Bot03] Botti, Julián V., "Estudio de métodos de desarrollo de sistemas multiagente". Revista Iberoamericana de Inteligencia Artificial. Nro. 18. Valencia, España, 2003.

[Bou05] Boulmakoul, Andel and Bruten, Janet. "Bringing the Web to its full potential (diagram of Semantic Web-Enabled Services vision)" http://www.hpl.hp.com/research/ssrc/competitive/web_services/swws_image.jpg Última visita enero 2005.

[Bra78] Brachman, R. J. "Structured inheritance networks". In W.A. Woods and R. J. Brachman editors, Research in Natural language Understanding, Quarterly Progress report No. 1, BBN Report No 3742 pages 36-78, Bolt, Beranek and Newman Inc., Cambridge 1978.

[Bra85] Brachman R. J. and Schmolze J. G. “An Overview of the KL-ONE Knowledge Representation System”. Cognitive Sciece, 9(2): 171-216, 1985.

[Bro02] Broekstra, Jeen; Kampman, Arjohn and van Harmelen, Frank. “Sesame: a Generic Architecture for Storing and Querying RDF and RDF Schema”. 1st International Semantic Web Conference (ISWC2002), June 9-12, 2002. Sardinia, Italy.

[Bru91] Brustoloni, Jose C., "Autonomous Agents: Characterization and Requirements," Carnegie Mellon Technical Report CMU-CS-91-204, Pittsburgh: Carnegie Mellon University. 1991.

[Cag97] Caglayan, A. and Harrison, C., “Agent Source Book– a complete guide to Desktop, Internet and IntranetAgents” , Wiley, 1997.

[Cal98] Calvanese, Diego; De Giacomo, Giuseppe; Lenzerini, Maurizio; Nardi, Daniele and Rosati, Riccardo “Information Integration: Conceptual Modeling and Reasoning Support” CoopIS98, pag 280-291, 1998.

[Car02] Cardoso, Jorge., and Sheth Amit. "Semantic e-Workflow Composition." Journal of Intelligent Information Systems (JIIS). 2002.

[Car02b] Cardoso, Jorge; Miller, John; Sheth, Amit and Arnold, Jonathan “Modeling Quality of Service for Workflows and Web Service Processes “ Technical Report# 02-002 v2, LSDIS Lab, Computer Science, University of Georgia, December 2002.

[CARS03] Condition Acquisition and Reporting System. Disponible en: http://www.carsprogram.org/public.htm Última visita septiembre 2003

[Cas03] Castells, Pablo. “La web semántica” Disponible en: http://www.ii.uam.es/~castells/publications/castells-uclm03.pdf. Universidad Autónoma de Madrid. 2003.

[Cec02] Ceccaroni, L. and Ribiere, M., “Experiences in Modeling Agentcities Utility-Ontologies with a Collaborative Approach”. In: Ontologies in Agent Systems Workshop, Autonomous Agents and Multi-Agent Systems Conference 2002, Bologna, Italy, July 2002.

[Cer02] Cerami Ethan “Web Services Essentials ;Distributed Applications with XML-RPC, SOAP, UDDI & WSDL”; ISBN 0-596-00224-6. 2002.

[Cerebra04] “Cerebra” Disponible en: http://www.semtalk.com/cerebra_construct.htm. Última vez accedido el día 04/07/2004

[Che04] Chen, Zhou; Liang-Tien, Chia and Bu-Sung, Lee. DAML-QoS Ontology for Web Services. Web Services QoS ontology, IEEE International Conference on Web Services (ICWS'04), June 6-9, 2004, San Diego, California, USA

[CMU04] CMU, Carnegie Mellon University Disponible en: http://www-2.cs.cmu.edu/~softagents/daml_Mmaker/daml-s_matchmaker.htm. Última visita noviembre 2004.

[Col03] Colucci, S.; Di Noia, T.; Di Sciascio, E.; Donini, F. M. and Mongiello, M. "Logic Based Approach to Web Services Discovery and Matchmaking". In Proceedings of Modeling E-services Workshop at 5th International Conference on Electronic Commerce, (ICEC'03). Pittsburgh, October 3, 2003.

[Cor00] Corcho, O. and Gómez-Pérez, A. "A Roadmap to Ontology Specification Languages", Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management, October, 2000 at http://delicias.dia.fi.upm.es/articulos/ocorcho/ekaw2000-corcho.pdf. EKAW 2000.

[Cor02] Corcho, Oscar; Fernández-López, Mariano and Gómez-Pérez, Asunción "Methodologies, tools and languages for building ontologies. Where is their meeting point?." NH Elsevier. Data Knowledge Engineering. 46(2003) 41-64.

[Dac03] Daconta, Michael C.; Obrst, Leo, J. and Smith, Kevin, T "The semantic Web. A guide to the future of XML, Web services, and knowledge management" (pag 196-197) 2003.

[DAML01]. "A revised example ontology", Disponible en: http://www.daml.org/2001/03/daml+oil-ex

[DAML02] Language feature comparison. Disponible en: http://www.daml.org/language/features.html.

[DAMLS03] DAML-S (and OWL-S) 0.9 Draft Release, Disponible en: http://www.daml.org/services/daml-s/0.9/

[DAMLS05] "Semantic Web Services Home Page" Disponible en: http://www.daml.org/services. Última vez accedido en enero de 2005

[DC05] The Dublin Core Metadata Initiative. Disponible en http://dublincore.org/ , Última visita 2 Febrero 2005

[Dec96] Decker, K.; Sycara, K. and Williamson, M. "Matchmaking and Brokering." Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96), Dec., 1996.

[Dec97] Decker, K.; Sycara, K. and Williamson, "Middle-Agents for the Internet". Proc. 15th IJCAI, pages 578-583, Nagoya, Japan, August 1997.

[Den02] Denny, Michael "Ontology Building: A Survey of Editing Tools" Disponible ne: http://www.xml.com/pub/a/2002/11/06/ontologies.html .November 06, 2002.

[Dig03] Dignum, V. and Weigand, H. "Toward and Organization Orientes Design Methodology for Agent Societies". Capítulo 9 del Libro: Intelligent Agent Software Engineering, IDEA Publishing, 2003. Pags 197-198.

[Dog02] Dogac, Asuman; Laleci, Gokce; Kabak, Yildiray and Cingil, Ibrahim. "Exploiting Web services Semantics: Taxonomies vs Ontologies". Middle East Technical University, Ankara (Turquía). 2002.

[DUET03] DUET. Disponible en: http://codip.grci.com/Tools/Tools.html Última visita septiembre de 2003

[Dum00] Dumbill, Edd. "Putting RDF to Work". Article on XML.com. 08-09-2000.

[Eas02] Eastlake, D.; Reagle, J. and Solo, D., "XML-Signature Syntax and Processing". W3C Recommendation. Disponible en: http://www.w3.org/TR/xmldsig-core/ 2002.

[EON02] Conference on ontology tools: Evaluation of Ontology-based Tools Workshop (EON2002) at the 13th International Conference on Knowledge Engineering and Knowledge Management, at (http://km.aifb.uni-karlsruhe.de/eon2002/). September, 2002

[ERTICO98] ERTICO Committee on DAB-based Multimedia, "DAB-based Multimedia ITS Applications, ITS Applications Strategy for Implementation" Disponible en http://www.tongji.edu.cn/~yangdy/its/ERTICO/dab_its.pdf

[Eur04] Europa ITS. The Tempo Programme. An ITS Programme for 2001-2006.En http://europa.eu.int/comm/transport/themes/network/english/its/html/its_activities_mip.html

[Euz96] Euzenat, J., "Corporative memory through cooperative creation of knowledge bases and hyper-documents", in: Proc. 10th Knowledge Acquisition for Knowledge-BasedSystems Workshop (KAW96), Ban., 1996.

[FACT04] "The FaCT system" Disponible en: http://www.cs.man.ac.uk/~horrocks/FaCT/. Última vez accedido el día 15/06/2004.

[Fer03] R. Ferraz; S. Labidi and B. Wanghon. "A semantic matching method for clustering traders in B2B Systems", First Latin American Web Congress (LA-WEB 2003) IEEE Computer Society 0-7695-2058-8/03.

[Fer99] Fernández López, M. "Overview Of Methodologies For Building Ontologies". Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5) Stockholm, Sweden, August 2, 1999.

[Fik02] Fikes, Richard; Jenkins, Jessica; Frank, Gleb and McIlraith, Sheila. "JTP: A Query Answering System for Knowledge Represented in DAML". UltraLog Meeting. Julio 2002. En http://www.ksl.stanford.edu/software/JTP/ Última vez accedido el día 04/07/2004.

[Fik03] Fikes, Richard; Hayes, Pat and Horrocks, Ian. "Daml Query Language abstract specification". Disponible en http://www.daml.org/2003/04/dql/dql abril 2003. Última vez accedido el día 06/07/2004.

[Fik04] Fikes, Richard; Hayes, Patrick and Horrocks, Ian, "OWL-QL, A Language for Deductive Query Answering on the Semantic Web". 2004

[FIPA02] Foundation for Intelligent Physical Agents. FIPA Abstract Architecture Specification, http://www.fipa.org/specs/fipa00001/, 2002.

[FIPA04] Foundation for Intelligent Physical Agents. Fipa agent management specification. Technical Report SC00023K, Foundation for Intelligent Physical Agents, Geneva, Switzerland, marzo 2004. Standard.

[FIPA04b] "Welcome to the Foundation for Intelligent Physical Agents" http://www.fipa.org/. Última vez accedido el día 04/07/2004.

[Flo99] Flores-Mendez, Roberto A. "Hacia una estandarización de los marcos de trabajo para Sistemas Multiagentes". 1999.

[Flog04] "Frame Logic" En http://flora.sourceforge.net/aboutFlogic.php. Última vez accedido el día 04/07/2004

[FOAF04] FOAF Vocabulary Specification Namespace Document 2 Sept 2004 - FOAF Galway Edition

[Fran96] Franklin, S. and Graesser, A. "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents." Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996 Disponible en: http://www.msci.memphis.edu/~franklin/AgentProg.html

[Fri99] Fridman Noy, Natalya and Musen Mark, A. "SMART: Automated Support for Ontology Merging and Alignment", KAW'99Twelfth Workshop on Knowledge Acquisition, Modeling and Management, Alberta, Canada, 4-7:1-20.

[Fri01] Fridman Noy, Natalya and McGuinness Deborah, L. "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001

[Gar04] Gartner Group. En http://www4.gartner.com/pages/section.php.id.2038.s.8.jsp

[GEO03] RDFIG *Geo vocab workspace* http://www.w3.org/2003/01/geo/

[Gil02] Gil, Yolanda and Ratnar, Varun. USC Information Sciences and Computer Science Department. "A comparison of (Semantic) Markup Languages". Proceedings of the 15th International FLAIRS Conference, Pensacola Beach, Florida, May 14-16, 2002

[Gob02] Goble, Carol. "Ontology Engineering & Ontology Acquisition". The University of Manchester. En http://www.semanticgrid.org/presentations/ontologies-tutorial/GGFpart4.ppt

[Gom96] Gómez-Pérez, A. "A Framework to Verify Knowledge Sharing Technology. Expert Systems with Application". Vol. 11, N. 4. 1996. PP: 519-529.

[Gom96b] Gomez-Perez, A.; Fernandez-Lopez, M. and de Vicente, A., "Towards a Method to Conceptualize Domain Ontologies", in: ECAI96 Workshop on Ontological Engineering, Budapest, 1996.

[Gom01] Gómez-Pérez, Asunción, "Evaluation of Ontologies". International Journal of Intelligent Systems 16(3):391-409. 2001.

[Gom04] Gómez-Pérez, Asunción; Fernández-López; Mariano and Corcho, Oscar "Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web", ISBN: 1-85233-551-3 Springer.

[Gom04b] Gómez-Pérez, Asunción; González-Cabero, Rafael, and Lama, Manuel: "A Framework for Description, Composition, and Evaluation of Semantic Web Services". IEEE Intelligent Systems. Special Issue on Semantic Web Services., 2004.

[Gon01] Gonzalez-Castillo, Javier; Trastour, David and Bartolini, Claudio. "Description Logics for MatchMaking of Services". HP Technical Reports ,October 30 th, 2001

[Graphviz04] Graphviz. Disponible en: http://www.research.att.com/sw/tools/graphviz/. Última visita Junio 2004

[Gru93a] Gruber, T.R., "Towards Principles for the Design of Ontologies used for Knowledge Sharing", Proc. Of International Workshop on Formal Ontology, Padova, Italy, 1993. Ed N. Guarino, Available as Technical Report KSL-93-94, Knowledge Systems Laboratory, Standford University.

[Gru93b] Gruber, T.R., "A Translation Approach to Portable Ontology Specifications". Knowledge Acquisition Journal, Vol. 5 pp. 199-200,1993.

[Grun95] Gruninger, M. and Fox, M.S., "Methodology for the design and evaluation of ontologies" in: Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, 1995.

[Grun02] Gruninger, M. & Uschold, M. "Ontologies and Semantic Integration" (Technical Paper). Seattle: The Boeing Company. 2002.

[Gu02a] Gu, Xiaohui and Nahrstedt, Klara. "A Scalable QoS-Aware Aggregation Model for Peer-to-Peer Computing Grids". Proceedings of IEEE International Symposium on High Performance Distributed Computing, Edimburgo, Escocia, 2002.

[Gu02b] Gu, Xiaohui, et al. "An XML-based Quality of Service Enabling Language for the Web", Journal of Visual Language and Computing (JVLC), special issue on multimedia languages for the Web. 13(1), pp. 61-95, Academic Press, Feb. 2002.

[Gu05] Gu, Xiaohui and Nahrstedt , Klara, "Distributed Multimedia Service Composition with Statistical QoS Assurances", to appear in IEEE Transactions on Multimedia, 2005.

[Gua98] Guarino, N., "Formal Ontology and Information Systems. Formal Ontology in Information Systems", Proc. Of the 1st International Conference, Trento, Italy, 6-8 june 1998, Ed. N. Guarino, IOS Press.

[Haa04] Haase, Peter; Broekstra, Jeen; Eberhart, Andreas and Volz, Raphael, "A comparison of RDF query languages" In Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 2004. November 2004.

[Har01] van Harmelen, Frank; Patel-Schneider, Peter F. and Horrocks Ian, editors. "An annotated version of the example ontology". Disponible en: http://www.daml.org/2001/03/daml+oil-walkthru

[Hau02] Haustein, Stefan and Pleumann, Jörg. "Is Participation in the Semantic Web Too Difficult?." In I. Horrocks and J. Hendler (editors), The Semantic Web - First International Semantic Web Conference, No. 2342, pages 448 ff., Springer, 2002.

[Hen00] Hendler, James and Deborah L. McGuinness. "The Darpa Agent Markup Language". IEEE Intelligents Systems, 15(6):67-73, 2000.

[Hen02] Hendler, J.; Golbeck, J. and Parsia, B. "Trust Networks on the Semantic Web". Disponible en: http://www.mindswap.org/papers/Trust.pdf , 2002.

[Her03] Herman Ivan, W3C Head of Offices Helsinki, "Introduction to the Semantic Web". 6 May, 2003.

[Hon02] Honrubia López, F.J., "Introducción a las Ontologías". Escuela Universitaria de Albacete. 2002.

[Hor01] Horrocks Ian, "Description Logis. Tutorial" given (jointly with Ulrike Sattler) at IJCAR-2001, Siena, Italy, June 19, 2001.

[Hor02] Horrocks Ian, "Existing & Emerging Technologies." The Monet Consortium IST-2001-34145 Noviembre de 2002 Deliverable D03 (Public).

[Hor03] Horrocks, Ian; Patel-Schneider, Peter F. and van Harmelen, Frank."From SHIQ and RDF to OWL:The Making of a Web Ontology Language". 2003.

[HTMLPane04] HTMLPane. Disponible en http://www.holub.com/. Última visita junio 2004.

[Hua00] Huang, Z.; Eliens, A.; van Ballegooij, A.; and de Bra, P., "A taxonomy of web agents (extended version)", Disponible en http://www.cs.vu.nl/~eliens/research/@archive/refs/wasa2000.pdf

[IBM04] IBM Web services standards and technologies for developers, http://www.ibm.com/developerworks/webservices Última visita noviembre 2004.

[IEEE96] IEEE Standard for Developing Software Life Cycle Processes. IEEE Computer Society. New York (USA). April 26, 1996.

[IMP03] IMPACT. Interactive Maryland Platform for Agents Collaborating Together. En: http://www.cs.umd.edu/projects/impact/ Última visita noviembre 2003.

[INF03] The InfoSleuth model http://www.argreenhouse.com/InfoSleuth/ Última visita noviembre 2003

[INT04] Intel Web Services, http://www.intel.com/ebusiness/webservices/ Última visita noviembre 2004.

[Ise99] Isern,D. "Avaluació d'entorns de desenvolupament de Sistemes Multiagent". Universitat Rovira i Virgili, Septiembre, 1999. PFC Enginyeria Tècnica Informàtica de Sistemes. Disponible en: http://www.fut.es/~aisern/pfcs.htm

[ISO03] http://www.iso.org/ Ultima visita noviembre 2003.

[ISO04] ISO TC 204/SC N, ISO/WD 14813-1 Intelligent Transport Systems -- Reference Model Architecture(s) for the ITS sector -- Part 1: ITS Fundamental Services. Preparatory, 12-09-04.

[ISO05] Agenda: ISO TC204 WG1 Meeting 17-21 January 2005, Sydney, Australia.

[ITS04]. ITS España . http://www.itsespana.com/ Última visita diciembre de 2004.

[JADE04] Guia de Programación. JADE 3.2 http://jade.tilab.com/doc/programmersguide.pdf. Actualización julio 2004.

[Jen98] Jennings, N.R.; Sycara, K. and Wooldridge, M. "A Roadmap of Agent Research and Development". In: Autonomous Agents and Multi-Agent Systems Journal, N.R. Jennings, K. Sycara and M. Georgeff (Eds.), Kluwer Academic Publishers, Boston, 1998, Volume 1, Issue 1, pages 7-38.

[JENA05] Jena 2. "A Semantic Web Framework" Disponible en http://www.hpl.hp.com/semweb/jena.htm, Última visita 15 de febrero de 2005.

[JessKb04] DAMLJessKB. En: http://edge.cs.drexel.edu/assemblies/software/daml-jesskb/damljesskb.html Última vez accedido el día 16/06/2004

[Jos04] Josang, Audun; Ismail, Roslan and Boyd, Colin, "A Survey of Trust and Reputation Systems for Online Service Provision" University of Queensland, Australia, 2004.

[KAON04] The Karlsruhe Ontology and Semantic Web Framework. "Developer's Guide for KAON 1.2.7". January 2004.

[Kar99] Karp, R.; Chaudhri, V. and Thomere, J. "XOL: An XML-Based Ontology Exchange Language" 1999, Disponible en http://www.ai.sri.com/pkarp/xol/xol.html Última vez accedido el día 04/07/2004

[Kle00] Klein M. et al., "The Relation between Ontologies and Schema-Languages: Trans-lating OIL Specifications to XML Schema," Proc. Workshop on Applications of Ontolo-gies and Problem-Solving Methods, 14th European Conf. On Artificial Intelligence, Berlin, 2000.

[Kle01] Klein, M. and Bernstein A.. "Searching for Services on the Semantic Web using Process Ontologies." in The First Semantic Web Working Symposium (SWWS-1). 2001. Stanford, CA USA.

[Kle03] Klein, Michael and König-Ries, Birgitta, "A Process and a Tool for Creating Service Descriptions based on DAML-S", 4th VLDB Workshop on Technologies for E-Services (TES'03).

[Lei03] Lei Li and Horrocks Ian. "A software Framework For Matchmaking Based on Semantic Web Technology". Twelfth International World Wide Conference (WWW2003), pages 331-339, ACM, 2003.

[Len90] Lenat, D.B. and Guha, R V. "Building Large Knowledge-based systems. Representation and Inference in the Cyc project." Addison-Wesley, Reading, Massachusetts. 1990.

[LOOM99] "Loom" Disponible en: http://www.isi.edu/isd/LOOM/LOOM-HOME.html. Última vez accedido el día 04/07/2004

[Llo05] Llorens Sánchez, José. "Formatos de Intercambio de Información Bibliográfica". Primera Parte. Departamento de Sistemas Informáticos y Computación. Facultad de Informática. UPV. Licenciatura en Documentación. 2005.

[McD04] DRS: McDermott Drew "A Set of Conventions for Representing Logical Languages in RDF" January 12, 2004.

[Mae02] Maedche, Alexander. University of Karlsruhe, Germany, "Ontology Learning for the semantic Web", Kluwer Academic Publishers. 2002.

[Mag02] Magkanaraki, A.; Karvounarakis, G.; Anh, T.T; Christophides, V. and Plexousakis, D., "Ontology Storage and Quering." Technical Report Nº 308. IST-2001-29243.

[Mal02] Malik Ayesha , M2002 XML-Journal, "XML, Ontologies, and the semantic Web, The second generation of the web"

[Man02] Mani, Anbazhagan and Nagarajan, Arun "Understanding quality of service for Web services, Improving the performance of your Web services". IBM,India, Enero 2002.

[Man04] Mansour, et. al. "Intelligent Agents Review". Proceedings of The Six International Conference on Information Integrations and Web-bases Applications & Services – IIWAS2004. Yakarta, Indonesia, Sept. 2004. Pags 745-750.

[Mar99] Marais Hannes and Rodeheffer Tom, "Automating the Web with WebL", Compaq Systems Research Center. 1999.

[Mar02] Martin, David; Paolucci, Máximo and McIlraith, Sheila. Lista de Servicios Web. Disponible en: http://lists.w3.org/Archives/Public/www-ws/2002Mar/0009.html Marzo 2002.

[Mar02b] Martin, David; "DAML-S: Bringing Semantics to Web Services", AAAI Workshop, July 2002.

[Mar03] Martin David. Lista de Servicios Web. Disponible en: http://lists.w3.org/Archives/Public/www-ws/2003May/0028.html , Mayo 2003.

[Mar03b] Martin, David. Lista de Servicios Web. Disponible en: http://lists.w3.org/Archives/Public/www-ws/2003Aug/0029.html, Agosto 2003.

[Mar04] Martin, David "OWL-S 1.0 release announcement" 01/09/04 Disponible en: http://www.daml.org/listarchive/daml-all/0309.html

[Mar05] Martín, Gregorio y Martín Benítez, Isabel. "Curso de XML. Introducción al lenguaje de la Web". ISBN: 84-205-4245-8. Editorial PEARSON Prentice Hall. 2005.

[Mar05b] Martín, Gregorio y Carrillo, Eduardo. "El papel de las tecnologías XML en la Nueva Web". Revista Novática, Nº 173, Enero-Febrero de 2005.

[Max04] Maximilien, E.M and Singh, M.P. "A framework and ontology for dynamic web services selection". IEEE Internet Computing, Sept/Oct 2004.

[McB01] McBride B., "Jena: Implementing the RDF Model and Syntax Specification". In: Steffen Staab et al. (eds.): Proceedings of the Second International Workshop on the Semantic Web- SemWeb2001. May 2001

[McI01] McIlraith, S.; Son, T.C. and Zeng, H. "Semantic Web Services" IEEE Inteligent Systems. Special Issue on the Semantic Web. 16(2):46-53, March/April, 2001.

[Min74] Minsky Marvin, "A Framework for Representing Knowledge" MIT-AI Laboratory Memo 306, June, 1974.Reprinted in The Psychology of Computer Vision, P.

Winston (Ed.), McGraw-Hill, 1975. Shorter versions in J. Haugeland, Ed., Mind Design, MIT Press, 1981, and in Cognitive Science, Collins, Allan and Edward E. Smith (eds.) Morgan-Kaufmann, 1992 ISBN 55860-013-2.

[Moh02] Mohsin Waqar. "Semantic Discovery of Web Services", 2002.

[Mor00] Moreno Ortiz, Antonio. "Diseño e implementación de un lexicón computacional para lexicografía y traducción automática". ISSN: 1139-8736 Depósito Legal: B-35510-2000 Disponible en http://elies.rediris.es/elies9/4-1.htm

[MS04] Microsoft Corporation Disponible en: http://msdn.microsoft.com/webservices/understanding/default.aspx Última visita Diciembre 2004.

[Mye02] Myerson Judith M., "IBM Advancing the Web Services Stack" Disponible en http://www-106.ibm.com/developerworks/webservices/library/ws-wsa/?open&l=484,t=gr, 2002.

[MySQL04] "MySQL: The World's most popular Open source database". Disponible en: http://www.mysql.com Última vez accedido el día 06/07/2004.

[NAICS04] NAICS: North American Industry Classification Systems. http://www.census.gov/epcd/www/naics.html Última vez accedido el día 06/09/2004.

[Nar03] Narayanan, S. and McIllarith,S. "Analysis and Simulation of Web Services", Computer Networks, vol 42, Nro 5. 2003, pag 675-695.

[Nec91] Neches, R.; Fikes, R.E.; Finin T.; Gruber T.R.; Senator T., and Swartout W.R. "Enabling technology for knowledge sharing". AI Magazine. Vol.12 (3) p.p. 36-56, 1991

[Ngu02] Nguyen T., Giang and Dang T., Tung "Agent Platform Evaluation and Comparison". Pellucid 5FP IST-2002-34519, Jun 2002

[NI03] Network Inference, Description Logics White Paper Disponible en http://www.networkinference.com/ Última visita noviembre 2003.

[NOAA04] NOAA, En: http://www.csc.noaa.gov/metadata/text/whatismet.htm Última visita julio 2004

[Noi03] Di Noia, T.; Di Sciascio E.; Donini, F.M. and Mongiello M. "A System for Principled Matchmaking in an Electronic Marketplace". WWW2003.

[OCML99] "OCML: Operacional Conceptual Modelling Language Page", en: http://kmi.open.ac.uk/projects/ocml/. Última vez accedido el día 04/07/2004

[ODESWS05] ODE SWS A Toolset for Design and Composition of Semantic Web Services. Disponible en http://kw.dia.fi.upm.es/odesws/ Última visita 30 enero 2005.

[Ogd23] Ogden, C. K., and Richards, I. A., "Meaning of meaning". New York: Harcourt & Brace, 1923.

[OIL02] "Welcome to the Oil Page" Disponible en: http://www.ontoknowledge.org/oil/. Última vez accedido el día 30/05/2004.

[OilEd04] OilEd Ontology Editor web page. En: http://oiled.man.ac.uk/) Última visita Diciembre 2004.

[Omi00] Omicini, A. "From Objects to Agent Societies: Abstractions and Methodologies for the Engineering of Open Dsitributed Asystems." WOA 2000 – Dagli Oggetti agli Agenti: Tendenze evolutive dei sistem software, Bologna: Pitagora Editrice.

[OKBC95] "Open Knowledge Base Connectivity Working Group" http://www.ai.sri.com/~okbc/. Última vez accedido el día 04/07/2004

[Ont02] "A survey on ontology tools: OntoWeb report on a comparative study of 11 ontology editors plus several other ontology tools", May, 2002 IST-2000-29243 at http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D13_v1-0.zip.

[Ont02b] Ontology Language Standardisation Efforts. Editor Sean Bechhofer Jan 2002.

[OntoEdit04] OntoEdit. Disponible en: http://ontoserver.aifb.uni-karlsruhe.de/ontoedit/ Última visita noviembre 2004

[Ontol04] Ontolingua. Disponible en http://www.ksl.stanford.edu/software/ontolingua/. Última visita noviembre 2004

[OntoSaurus04] OntoSaurus. Disponible en: http://www.isi.edu/isd/ontosaurus.html Última visita noviembre 2004

[OTAP05] Open Travel data Access Protocol http://www.itsproj.com/otap/about.html, Última visita enero 2005

[OTK02] Welcome to OnToKnowledge. Disponible en: http://www.ontoknowledge.org/index.shtml. Última vez accedido el día 04/07/2004

[OWL02] "Web Ontology Language (OWL), Guide Version 1.0", W3C Working Draft 4 November 2002.

[OWL04] "OWL Web Ontology Language Reference", W3C Recommendation 10 February 2004. Disponible en: http://www.w3.org/TR/2004/REC-owl-ref-20040210/

[OWL04b] "OWL Web Ontology Language Semantics and Abstract Syntax". W3C Recommendation 10 February 2004 Disponible en: http://www.w3c.org/TR/owl-semantics

[OWLS03] "OWL-S 1.0 Release 2003-11", En http://www.daml.org/services/owl-s/1.0/

[OWLS04] "OWL-S 1.1 Release 2004-11", En http://www.daml.org/services/owl-s/1.1/

[OWLSE05] Editor OWL-S from CMU, Disponible en: http://www.daml.ri.cmu.edu/tools/details.html Última visita enero 2005.

[OWLSEdit04] Editor OWL-S from University of Malta, Disponible en: http://staff.um.edu.mt/cabe2/supervising/undergraduate/owlseditFYP/OWLSEdit.html Última visita noviembre 2004.

[OWLSML04] Tools for OWLS, Disponible en: http://www.mindswap.org/2004/owl-s/index.shtml Última visita noviembre 2004

[OWLSRI04] Editor OWL-S from SRI International. Disponible en: http://owlseditor.semwebcentral.org/ISWC-Poster.pdf, presented at 7th International Protégé Conference. Última visita noviembre 2004

[Pan04] Pan Feng. "OWL Time Ontology" Disponible en: http://www.isi.edu/~pan/daml-time/time.owl, Última vez accedido el día 04/07/2004.

[Pao02a] Paolucci, Massimo; Kawamura, Takahiro; Payne, Terry R. and Sycara, Katia. "Semantic Matching of Web Services Capabilities". In The First International Semantic Web Conference (ISWC), 2002.

[Pao02b] Paolucci, Massimo; Kawamura, Takahiro; Payne, Terry R. and Sycara, Katia. "Importing the semantic Web in UDDI.", Proceedings of Web Services, E-business and Semantic Web Workshop. 2002.

[Pao03] Paolucci, Massimo; Sycara, Katia and Kawamura, Takahiro, "Delivering semantic Web Services", WWW2003.

[Pao03b] Paolucci Massimo and Sycara Katia "Toward a Semantic Web e-commerce" Witold Abramowicz, Gary Klein (eds.), Business Information Systems, Proceedings of BIS 2003, Colorado Springs, USA

[Pat93] Patel-Schneider, Peter F. and Swarout, Bill "Description-Logic Knowledge Representation." Nov. 1993.

[Pay01] Payne, Terry R.; Paolucci, Massimo and Sycara Katia "Advertising and Matching DAML-S Service Descriptions", En: Semantic Web Working Symposium, 2001.

[Pay02] Payne, Terry R. and Abela, Charlie. Lista de Servicios Web. Disponible en: http://lists.w3.org/Archives/Public/www-ws/2002Jan/0008.html . Enero 2002.

[Pay04] Payne, T. and Lassila, O. "Semantic Web Services". IEEE Intelligent Systems. Special Issue on Semantic Web Services. 2004.

[Pas92] Pastor López Oscar. Tesis Doctoral. “Diseño y desarrollo de un entorno de producción automática de software basado en el Modelo Orientado a Objetos.” Valencia, 4 abril de 1992.

[Pellet03] Pellet OWL Reasoner. En http://www.mindswap.org/2003/pellet/index.shtml

[Pil03] Piliora, Thomi; Tsalgatidou, Aphrodite and Batskis, Alexandros. “Using WSDL/UDDI and DAMLS in Web Service Discovery”. WWW2003

[PROTEGE04] The Protégé Ontology Editor and Knowledge Acquisition System Disponible en: http://protege.stanford.edu/ Última visita julio 2004

[PWI04] PWI 24531. “Using XML in ITS Standards, Data Registries and Data Dictionaries”. TC204 WG1 Japan.

[PWI05] PWI “Using Web Servicies (machine-machine delivery) for ITS service delivery”.Vancouver

[Qui68] Quillian, M. R. "Semantic Memory", en M. Minsky (ed.) Semantic Information Processing. Cambridge, Mass: The MIT Press, 27-70. 1968.

[RACER04] Racer. Disponible en: http://www.cs.concordia.ca/~haarslev/racer/ Última visita noviembre 2004

[RDF04a] “Concepts and Abstract Syntax”. W3C Recommendation 10 February 2004 , Disponible en: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[RDF04b] “RDF/XML Syntax Specification (Revised) Resource Description Framework (RDF)”, W3C Recommendation 10 February 2004, Disponible en: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/

[RDFS04] “RDF Vocabulary Description Language 1.0: RDF Schema”, W3C Recommendation 10 February 2004 Disponible en: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

[RDQL04] RDQL Query de HP. En http://www.hpl.hp.com/semweb/rdql.htm Última vez accedido el día 06/07/2004.

[Rea02] Reagle, Joseph M. “Key Free Trust in the Semantic Web: Finding Bacon's Key”. Disponible en:http://www.w3.org/2002/03/key-free-trust. 2002.

[Rec02] Rector Alan L. “OilEd Normalised Ontology Tutorial- Biomedical Version”, Oiled Ontology Editor web page (http://oiled.man.ac.uk/) Última visita julio 2004

[Rec03] Rector Alan L. “Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms” including OWL K-CAP’03, October 23–25, 2003, Sanibel Island, Florida, USA. pp 121-8 Copyright 2003 ACM 1-58113-583-1/03/0010

[Rec04a] Rector Alan L. "Defining N-ary Relations on the semantic Web: Use With Individuals", W3C Working Draft 21 julio 2004.

[Rec04b] Rector Alan L. "Representing Specified Values in OWL:value partitions and value sets". W3C Working Draft 3 agosto 2004.

[RET01] RETSINA MATCHMAKER Disponiblr en: http://www-2.cs.cmu.edu/~softagents/matchmaker.html Última visita 15-Noviembre 2003

[RQL] "The RDF Query Language" Disponible en: http://139.91.183.30:9090/RDF/RQL/ Forth Institute of Computer Sciences. Última vez accedido en 06/07/2004

[Rib00] Ribière M. and Charlton P "Comparison of ontology languages: Ontology Overview from Motorola Labs with a comparison of ontology languages", December, 2000, disponible en http://www.fipa.org/docs/input/f-in-00045/f-in-00045.pdf .

[Rim03] Rimassa Giovanni. "Runtime Support for Distributed Multi-Agent Systems", Disponible en http://jade.cselt.it/papers/Rimassa-PhD.pdf. In Ph. D. Thesis, University of Parma, Jan 2003.

[Rob02] Roberts Angus. "An Introduction to OilEd". Geodise DAML + OIL Workshop 19-20 February 2002 , Disponible en Internet en http://oiled.man.ac.uk/tutorial/

[RTML03] Road Traffic Message Application Markup Language , Transport Protocol Experts Group . Disponible en: http://www.tpeg.org/pdf/standardisation. Última visita noviembre 2003.

[RuleML03] "The Rule Markup Initiative". Disponible en: http://www.ruleml.org/ Última visita diciembre 2003.

[RWML03] "Road Web Markup Language" Disponible en: http://rwml.its-win.gr.jp/eng/Draft-080/RWML-080-1.html Última visita noviembre 2003

[Sam01]. Samper, J. J.; Moreno, P.;. Sánchez, I.; Carrillo, E.; Cirilo, R. y Tomás, V.R. "Servicios Pre/On Trip de información al usuario basados en tecnología WAP. Experiencia en el Servei Cátala de Trànsit", II Congreso Nacional sobre Sistemas Inteligentes de Transporte y Explotación de Carreteras, 2001

[Sam02] Samper, J.J.; Cervera, A.; Sánchez, I.; Carrillo, E. y Martínez, J. "Desarrollo de un portal de voz de información de tráfico, para el Servei cátala de Trànsit, orientado a la conducción segura". ISBN : 84-89875-36-7 III Congreso Nacional sobre sistemas inteligentes de transporte.

[Sam03] Samper, J.J.; Cervera, A.; Sánchez, I. and Carrillo, E., "Development of voice services to provide traffic information (application in the SCT)" Congress: 10th World congress and exhibition on Intelligent Transport Systems and Services, 2003.

[Sam03b] Samper, J.J.; Cervera, A.; Carrillo, E. and Sánchez, I., "Speakeasy. Talking congestion" , Pages: 68-70 Magazine: Traffic Technology International. ISSN 1356-9252 Year: 2003.

[San02] Sánchez, I.; Samper, J.J.; Tomás, V.R.; Martínez J. J. and Carrillo E.,"Information on the move", Pages: 36-38 Magazine: Traffic Technology International. ISSN 1356-9252 Year: 2002

[Sat03] Sattler, U.; Calvanese, D. and Molitor, R., The Description Logic Handbook, Theory, Implementation and Applications, "Relationships with other formalisms, Semantic Data Models", pag 164, 2003.

[SeRQL04] Aduna B.V., "Chapter 6. The SeRQL query language, rev. 1.1" from User Guide for Sesame Updated for Sesame release 1.1 Copyright © 2002-2004. Sirma AI Ltd. Disponible en http://www.openrdf.org/doc/users/ch06.html. Última vez accedido el día 06/07/2004.

[Sha93] Shadbolt, N.; Motta, E. and Rouge A. "Constructing knowledge based systems". IEEE Software, 10(6):34–38, 1993.

[SHOE99] "SHOE: Simple HTML Ontology Extensions" http://www.cs.umd.edu/projects/plus/SHOE/. Última vez accedido el día 04/07/2004

[SIG04] SIGNATURE Disponible en: http://www.w3.org/Signature Última visita Diciembre 2004.

[Sir04] Sirin, E.; Parsia, B. and Hendler J., "Filtering and Selecting Semantic Web Services with Interactive Composition Techniques". IEEE Intelligent Systems, August, 2004

[SOAP03] SOAP Version 1.2 Part 0: PrimerW3C Recommendation 24 June 2003, Disponible en: http://www.w3.org/TR/soap12-part0/

[Sow00] Sowa, John F., "Knowledge Representation: Logical, Philosophical, and Computational Foundations", Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[SPARQL05] SPARQL W3C. Disponible en: http://www.w3.org/TR/rdf-sparql-query/. Última visita enero 2005.

[Sta01] Staab, S.; Schnur, H.P.; Studer, R. and Sure, Y., "Knowledge processes and ontologies", IEEE Intelligent Systems 16 (1). 2001.

[Ste98] Steve, G; Gangemi A. and Pisanelli D. "Integrating Medical Terminologies with ONIONS Methodology". Disponible en: http://saussure.irmkant.rm.cnr.it ,1998.

[Ste03] Stevens, Robert; Wroe, Chris; Bechhofer, Sean; Lord, Phillip; Rector, Alan and Goble, Carole. Conference Review, "Building ontologies in DAML + OIL, Comparative and Functional Genomics", Comp Funct Genom 2003; 4: 133–

141.Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cfg.233

[Stu98] Studer, R.; Benjamins, R. and Fensel, D. "Knowledge Engineering: Principles and Methods". DKE 25(1-2).pp:161-197. 1998.

[Sur04] Sure, York and Studer, Rudi, "Towards the Semantic Web, Ontology-Driven Knowledge Management", Cap 3 "A methodology for ontology-based knowledge management" pag. 33-42. 2004.

[SVG03] Scalable Vector Graphics (SVG) 1.1 Specification W3C Recommendation 14 January 2003. Disponible en: http://www.w3.org/TR/SVG11/

[Swa97] Swartout, B.; Ramesh, P.; Knight, K. and Russ, T., "Toward Distributed Use of Large-Scale Ontologies", AAAI Symposium on Ontological Engineering, Stanford, 1997.

[SWOOP04] SWOOP Disponible en: http://www.mindswap.org/2004/SWOOP/ Última visita noviembre 2004

[SWRL04] "SWRL 0.6: Semantic Web Rules Languages" Disponible en: http://www.daml.org/2004/04/swrl/rules-all.html Última vez accedido el día 04/07/2004

[SWRLEd05] Editor de SWRL para OWL. Plugin de Protégé. Disponible en http://protege.stanford.edu/plugins/owl/swrl/ Última visita 30 de enero 2005

[SWSIG05] Semantic Web Services Interest Group. Disponible en: http://www.w3c.org/2002/ws/swsig/ Última visita 29 de enero 2005

[SWSI05] Semantic Web Services Initiative (SWSI) Disponible en: http://www.swsi.org/ Última visita 29 enero 2005.

[SWSL05] Semantic Web Services Language (SWSL) Committee, Disponible en: http://www.daml.org/services/swsl/ Última visita 29 enero 2005

[Syc99] Sycara, K.; Klusch, M.; Widoff, S. and Lu, J., "Dynamic Service Matchmaking Among Agents in Open Information Environments". Journal ACM SIGMOD Record, 1999.

[Syc01] Sycara, K., "Brokering and Matchmaking for Coordination of Agent Societies: A Survey". In Coordination of Internet Agents, A. Omicini et al. (eds.), Springer., 2001.

[Syc02] Sycara, Katia; Widoff, Seth; Klusch, Matthias and Lu Jianguo, "LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace." In Autonomous Agents and Multi-Agent Systems, 5, 173–203, 2002.

[Tab02] Taboada, M.; Argüello, M.; Des, J. and Martínez, D., "An use case for DAML+OIL: an ontology in the ophthalmology domain". 2002.

[TDML03] Traffic Data Markup Language Disponible en: http://www.travelerinformation.com Última visita septiembre 2003

[TEN-T00] "Deployment of Intelligent Transport Systems on the Trans-European Road Network". Report of the TEN-T Expert Group on ITS for Road Traffic Management. European Commission. April 2000.

[TIML03] Traveler Information Markup Language. Disponible en: http://www.mitretek.org/its/TripInfo/intro.html Última visita septiembre 2003

[TMML03] Traffic Model Markup Language Disponible en: http://www.ce.ufl.edu/trc/Research/tmmlspec.pdf Última visita septiembre 2003

[Tom04]. Tomás, V. R; García, J.F. y Samper, J.J. "Las nuevas Tecnologías y su aplicación en el Tráfico", "Capacidades Humanizadoras de las TIC" V Jornadas de Informática y Sociedad . Facultad de Ingeniería – ESIDE y el Aula de Etica de la Universidad de Deusto. 2004.

[TranXML03] "XML for Transportation-Related Transactions" Disponible en: http://www.transentric.com/products/commerce/tranxml.asp Última visita septiembre 2003.

[Tras01] Trastour, D.; Bartolini, C. and Gonzalez-Castillo, J. "A Semantic Web Approach to Service Description for Matchmaking of Services", Proc. 1st Semantic Web Working Symposium, CA. 2001.

[Tras02] Trastour, D.; Bartolini, C.; Chris Preist C. HP-Laboratories. "Semantic Web Support for the Business-to-Bussines E-Commerce Lifecycle". WWW2002.

[TRIDENT02] Trident Project. Disponible en http://www.ertico.com/activiti/projects/trident/home.htm, Última visita 17-02-05.

[Tyr00] Tyrone, Grandison and Morris, Sloman. "A survey of Trust in Internet Applications". Imperial College, Department of Computing. 2000.

[UDDI04] "UDDI, Universal Description, Discovery and Integration". Disponible en: http://www.uddi.org/specification.html Última visita septiembre 2004

[UNSPSC04] "UNSPSC: United Nations Standard Products and Services Codes". Disponible en: http://www.unspsc.org/ . Última vez accedido el día 09/09/2004

[UPM00a] UPM. Departamento de Ingeniería Civil: Transportes. ITS: "Beneficios de los ITS" http://www.caminos.upm.es/ict/Seminarioits2000/apartados/

[UPM00b] UPM. Departamento de Ingeniería Civil: Transportes. "Normalización". Seminario ITS http://www.caminos.upm.es/ict/Seminarioits2000/apartados/

[UPM00c] UPM. Departamento de Ingeniería Civil: Transportes. "El desafio técnico en la Unión Europea" ITS http://www.caminos.upm.es/ict/Seminarioits2000/apartados/

[Usc95] Uschold, M. and King, M., "Towards a methodology for building ontologies." In Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95, Montreal, Canada. 1995.

[Usc98] Uschold, M.; King, M; Moralee, S. & Zorgios, Y. "The Enterprise Ontology. Konowledge Eng. Rev." 13(1) pp. 32-89, 1998.

[UV03] Actividades de la Universidad de Valencia para el año 2003. ARTS-MIP. Memoria Técnica

[UV03b] Actividades de la Universidad de Valencia para el año 2003. SERTI-MIP. Memoria Técnica

[UWO04] UWO CS457:Computer Networks II. WAN Technologies and Techniques Disponible en http://www.csd.uwo.ca/courses/CS457a/outline.html. Última visita mayo 2004.

[Van96] Van Heijst, G.; Schereiber, A.T. and Wieling, B.J. " Using Explicit Ontologies in KBS Development" International Journal of Human an Computer Studies, 1996

[Vas01] Vasudevan Venu. "A Web Services Primer". Abril 2001. Disponible en: http://www.xml.com/pub/a/2001/04/04/webservices/index.html

[WebODE04] WebODE. Disponible en: http://delicias.dia.fi.upm.es/webODE/ Última visita noviembre 2004.

[WebOnto04] WebOnto. Disponible en: http://webonto.open.ac.uk/webonto/ Última visita noviembre 2004.

[Wei97] Weigand, H.,"Multilingual Ontology-Based Lexicon for News Filtering –The TREVI Project", en K. Mahesh (1997): 138-159.

[Woo97] Wooldridge, M and Jennings, N. "Agent Technology: Foundations, Applications and Markets". Ed. Springer, 1997.

[Woo99] Wooldridge M.. "Multiagent Systems: A modern approach to distributed artificial intelligence". The MIT Press., Cambridge, Ma., 1999.

[WSA04] Web Services Architecture W3C Working Group Note 11 February 2004. Disponible en http://www.w3.org/TR/ws-arch/

[WSCI04] "Web Services Choreography Interface (WSCI) 1.0 Specification" Disponible en: ftp://ftpna2.bea.com/pub/downloads/wsci-spec-10.pdf

[WSDL04] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Working Draft 3 August 2004

[WSFL04] Web Services Flow Language (WSFL). Disponible en: http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf Última visita noviembre 2004.

[WSIF04] Apache WSIF. Disponible en: http://ws.apache.org/wsif/. Última visita noviembre 2004.

[W3C01] W3C 2001 http://www.w3.org/2001/sw/

[W3C02] Constantinescu, I. and Faltings, B. World Wide Web Consortium "Efficient Matchmaking and Directory Services". Technical Report No IC/2002/77, En: http://www.w3.org/2002/ws/ . Noviembre, 2002.

[W3C04] World Wide Web Consortium (W3C). Web Services Glossary, W3C Working Group Note 11. Disponible en: http://www.w3c.org/TR/ws-gloss/, Feb 2004.

[XML04] eXtensible Markup Language (XML) 1.1 W3C Recommendation 04 February 2004, edited in place 15 April 2004.

[XMLS01] XML Schema Part 2: Datatypes W3C Recommendation 02 May 2001 Disponible en: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[XMLS04] XML Schema Part 0: Primer Second Edition W3C Recommendation 28 October 2004. Disponible en http://www.w3.org/TR/xmlschema-0/

[XQuery04] "XQuery 1.0: An XML Query Language". W3C Working Draft 29 October 2004. Disponible en: http://www.w3.org/TR/xquery/ Última vez accedido el día 04/11/2004.

[Zar95] Zaremski, Amy Moormann and Wing Jeannette M. "Signature matching: a Tool for Using Software Libraries", ACM Transactions on Software Engineering and Methodology, 1995.

[Zar97] Zaremski, Amy Moormann and Wing Jeannette M. "Specification Matching of Software Components", ACM Transactions on Software Engineering and Methodology (TOSEM), 1997.

PUBLICACIONES

Derivadas de la tesis:

- Este artículo describe las principales características y componentes de la herramienta desarrollada con el fin de integrar la definición de perfiles de servicios web semánticos, basada en los lenguajes de descripción de servicios, con la visualización y verificación de consistencia de las ontologías necesarias para describir los parámetros, así como la realización del proceso de emparejamiento.

- "Integrating Ontologies Visualization with the Edition of Profiles in Semantic Web Services: OntoService". J. Javier Samper, Arturo Cervera, Eduardo Carrillo, J.J Martínez. Enviado a: Radiomatics - Journal on Communications Engineering. ISSN : 1693-5152

- Este artículo muestra las principales características del algoritmo de emparejamiento de SWS propuesto. Además se detallan los principales componentes relacionados con el proceso de implementación. Finalmente, se describe la implementación de comparaciones entre parámetros de perfiles de servicio mediante consultas realizadas a la base de conocimiento.

- "Algoritmo de emparejamiento de perfiles en Servicios Web Semánticos". José Javier Samper, Eduardo Carrillo, Juan José Martínez. Enviado a: "Revista Colombiana de Computación" UNAB - Colombia. ISSN 1657 - 2831

- En este artículo se resume el trabajo realizado en el diseño y construcción de un sistema multiagente basado en estándares FIPA que tiene como objetivo facilitar, asistir y optimizar los procesos de anuncio, descubrimiento e invocación de servicios Web relacionados con el área de tráfico. Se hace énfasis en la creación del sistema prototipo desde el punto de vista de sistemas multiagentes.

- "ESWIT: Emparejador de Servicios Web de Información de Tráfico". J. Javier Samper, J. José Martínez, Francisco Matas, Eduardo Carrillo. Enviado a: "Inteligencia Artificial, Revista Iberoamericana de I.A.".Publicación periódica distribuida por la Asociación Española para la Inteligencia Artificial (AEPIA). ISSN: 1137-3601 (c)

- Este artículo presenta una revisión del estado actual del desarrollo de sistemas multiagente (SMA), se describen los conceptos básicos, componentes y metodologías de desarrollo y plataformas para desarrollo de SMA. Posteriormente se presenta el concepto de servicios web semánticos y finalmente se resume la aplicación de los SMA en el descubrimiento automático de servicios web.

- "Estado actual del desarrollo de sistemas multi-agentes y su aplicación en el descubrimiento automático de servicios Web", Eduardo Carrillo, J. Javier Samper, J.José Martínez. Enviado a: Congreso Internacional en Inteligencia Computacional. Montería, Colombia Agosto 10 -12 de 2005

- Este artículo describe el estado del arte de los vocabularios existentes para información de tráfico, así como el desarrollo de un esquema de representación del conocimiento para el dominio del tráfico vial, con una semántica bien definida expresada mediante la creación de ontologías. Adicionalmente se describe como se han utilizado ontologías para describir semánticamente servicios de tráfico que se definan. A su vez, se detalla una extensión a metodologías para ingeniería ontológica existentes para la incorporación de procesos de reutilización de descripciones basadas en modelos E-R. Por último también son tratados aspectos de información multilingüe mediante ontologías.

- "Una aproximación al problema multilingüe en ontologías para la Web Semántica: Caso de Aplicación en los ITS". J. J. Samper, E. Carrillo Enviado a: Congreso Internacional en Inteligencia Computacional.Montería, Colombia.Agosto 10 -12 de 2005.

- En este artículo se presenta un interfaz gráfico genérico que permite tareas de control en el dominio de transporte. Las ontologías de tráfico vial desarrolladas en esta tesis, han servido para desarrollar los sistemas objeto y para proporcionar el conocimiento semántico.

- "An eXtensible Graphic Container with integrated Ontological Layer for Transportation Systems". V.R. Tomas, L. van den Berg, J.J. Samper, E. Carrillo. To be published in The 6th IEEE International Conference on Computer-Aided Industrial Design & Conceptual Design. CAID &CD 2005 Delft, The Netherlands, June, 2005.

- Este artículo trata dos aspectos principales: por una parte el desarrollo de las ontologías de tráfico, y por otra el entorno de edición de perfiles y ontologías. Centra su interés en los aspectos de representación y visualización de las ontologías creadas, haciendo uso de recomendaciones aportadas por el consorcio W3C, como Scalable Vector Graphics.

- "A Tool to Visualize and Export Road Traffic Information Based on Ontologies in Multiple Formats such as SVG", J. J. Samper, E. Carrillo, A. Cervera, J. J. Martínez. International Conference on Multimedia, Image Processing and Vision Computer Madrid (SPAIN), March 30 - April 1, 2005. Published by International Association for the Development of Advances in Technology ISBN: 84-933971-5-6. ISSN:1698-1073

- Este artículo describe el proceso de desarrollo de ontologías de tráfico vial y la descripción semántica de servicios Web de información de tráfico basados en OWL-S. También se describe el algoritmo de emparejamiento propuesto en esta tesis, y su integración junto con el resto de elementos en la arquitectura.

- "Multiagent System to automate the search of Road Traffic Information Web Services". J.J. Samper, J.J. Martínez, E. Carrillo, A. Cervera. IADAT International Conference on Automation, Control and Instrumentation. ISBN: 84-933971-2-1. Bilbao, Spain, February, 2005.

- Este artículo describe las principales características y componentes del entorno de edición desarrollado con el fin de integrar la definición de perfiles de servicios web semánticos basada en los lenguajes DAML-S y OWL-S, con la visualización y verificación de consistencia de las ontologías que definen los conceptos con los cuales interactúa un servicio web.

- "ONTO-SERVICE: Una Herramienta para la Edición de Perfiles y Visualización de Ontologías en Servicios Web Semánticos". J. J. Samper, A Cervera, E. Carrillo. WebMedia/LA-Web 2004, the Joint Conference the 2nd Latin American Web Congress and the 10th Brazilian Symposium on Multimedia and the Web. Octubre 12-15, Ribeirao Preto, Brasil. Pag 304-306, ISBN 85-7669-010-1.

Relacionadas

- "A Web Tool to use traffic Management Plans". V. Ramón Tomás, J.F. Garcia, J.J. Samper, E. Carrillo WSEAS TRANSACTIONS on CIRCUITS Issue 1, Volume 2, January 2003 ISSN:1109-2734
- "Development and evaluation of a Tool for Dynamic WAP Applications". E.Carrillo, J.J Samper, V.Ramon Tomas, L. Van Den Berg.WSEAS TRANSACTIONS on COMMUNICATIONS Issue 1, Volume 2, January 2003 ISSN:1109-2742
- "Development of Voice Services to Provide Traffic Information (application in the SCT)". J.J. Samper, A. Cervera, I. Sánchez, E. Carrillo.10th World congress and exhibition on Intelligent Transport Systems and services, Año 2003
- "Realización de un sistema de alertas de tráfico por voz en el Servei Català de Trànsit", A. Cervera., J. J. Samper, E. Carrillo, V. R. Tomás. Pages: 28-31 Magazine: Equipamiento y Servicios Municipales ISSN 1131-6381 Año 2003
- "Speakeasy. Talking congestion", J.J. Samper, A. Cervera, E. Carrillo, I. Sánchez. Pages: 68-70 Magazine: Traffic Technology International. ISSN 1356-9252 Año 2003
- "Improving Traffic Management Plans using HTML and XML". V. Ramón Tomás, J. F. García, J. J. Samper, E. Carrillo, C. Cambres. International Conference on Electronics & Hardware Systems ISBN 960-8052-65-3, 2002 by WSEAS, 2002
- "Integration of Voice Services in Internet Applications", E. Carrillo, J. J. Samper, J.J. Martínez. Fourth International Conference on Information Integration and Web_based Applications & Services iiWAS2002, ISBN: 3-936150-18-4, Society for Modelling and simulation International. SCS-European Publishing House 2002
- "Bringing Internet to the Wireless World". E. Carrillo, R. Cirilo, J.J. Martínez, J.J. Samper, G. Martín. Publicación COMMUNICATIONS WORLD, Pages: 70-74, ISBN: 960-8052-38-6 Book: Electrical and Computer Engineering Series. A series of reference books and Textbooks. COMMUNICATIONS WORLD. N. Mastorakis Editor (http://www.worldses.org). Año:2001

ANEXO

CONSULTAS SERQL:

################################## C1 ##################################
Consulta que comprueba que dos conceptos son hermanos (mismo padre), pero no coinciden en el mismo tipo de restricción sobre una propiedad y rango común

```
Select distinct X, Y, Prop1, Z, Q
From {X} <daml:subClassOf> {Padre},
     {Y} <daml:subClassOf> {Padre},
     {X} <rdfs:subClassOf> {Rest},
     {Y} <rdfs:subClassOf> {Rest2},
     {Rest} <rdf:type> {<daml:Restriction>},
     {Rest2} <rdf:type> {<daml:Restriction>},
     {Rest} <daml:onProperty> {Prop1},
     {Rest2} <daml:onProperty> {Prop1},
     {Rest} Q {A},
     {Rest2} Z {A}
Where X = <ns8:Clase1>
and Y = <ns8:Clase2>
and Q = <daml:hasClass>
and Z = <daml:toClass>
```

Si se obtienen resultados son hermanos pero no coinciden en el mismo tipo de restricción sobre una propiedad común, mientras que si no devuelve resultados, este tipo de relación no existe entre ambos conceptos.

##############################C2 y C2' ##################################
Consultas que comprueban que dos conceptos son hermanos y que coinciden en el mismo tipo de restricción sobre una propiedad:

```
Select distinct X, Y, Prop1, Z
From {X} <daml:subClassOf> {Padre},
     {Y} <daml:subClassOf> {Padre},
     {X} <rdfs:subClassOf> {Rest},
     {Y} <rdfs:subClassOf> {Rest2},
     {Rest} <rdf:type> {<daml:Restriction>},
     {Rest2} <rdf:type> {<daml:Restriction>},
     {Rest} <daml:onProperty> {Prop1},
     {Rest2} <daml:onProperty> {Prop1},
     {Rest} Z {A},
     {Rest2} Z {A}
Where X = <ns8:Clase3>
and Y = <ns8:Clase2>
and Z = <daml:hasClass> --> PUEDE SER CAMBIADO POR <daml:toClass>
(C2')
```

Si se obtienen resultados son hermanos y coinciden en el mismo tipo de restricción sobre una propiedad común, mientras que si no devuelve resultados este tipo de relación no existe entre ambos conceptos

############################## C3 y C3' ################################
Consulta que comprueba que dos conceptos son hermanos que coinciden en el mismo tipo de restricción sobre una propiedad, y en el que uno de los dos conceptos tiene además una restricción del otro tipo sobre la misma propiedad y con el mismo rango.

Importante: este caso, si está la ontología razonada podría no darse, porque en determinadas circunstancias el concepto de cliente se clasificaria como padre del concepto del proveedor.

```
Select distinct X, Y, Prop1, M, N, Rng
From {X} <daml:subClassOf> {Padre},
     {Y} <daml:subClassOf> {Padre},
     {X} <rdfs:subClassOf> {Rest},
     {Y} <rdfs:subClassOf> {Rest2},
     {Rest} <rdf:type> {<daml:Restriction>},
     {Rest2} <rdf:type> {<daml:Restriction>},
     {Rest} <daml:onProperty> {Prop1},
     {Rest2} <daml:onProperty> {Prop1},
     {Rest} M {Rng},
     {Rest2} N {Rng}
Where X = <ns8:Clase1>
and Y = <ns8:Clase3>
and M = <daml:hasClass> --> PUEDE SER CAMBIADO POR <daml:toClass>
(C3')
```

Si devuelve 2 líneas de resultados (una para hasClass y otra para toClass) es porque se cumplen las condiciones de la consulta, si devuelve menos de 2 líneas, los conceptos no mantienen este tipo de relación de hermanos.

################################## C4 ################################
Consulta que comprueba que ambos conceptos son hermanos y que poseen los dos tipos de restricciones en común respecto a una misma propiedad y rango.

```
Select distinct X, Y, Prop1, M, N, Rng
From {X} <daml:subClassOf> {Padre},
     {Y} <daml:subClassOf> {Padre},
     {X} <rdfs:subClassOf> {Rest},
     {Y} <rdfs:subClassOf> {Rest2},
     {Rest} <rdf:type> {<daml:Restriction>},
     {Rest2} <rdf:type> {<daml:Restriction>},
     {Rest} <daml:onProperty> {Prop1},
     {Rest2} <daml:onProperty> {Prop1},
     {Rest} M {Rng},
     {Rest2} N {Rng},
     {Rng} <rdf:type> {<daml:Class>},
     {Rng} <rdfs:subClassOf> {<daml:Thing>}
Where X = <ns8:Clase3>
and Y = <ns8:Clase3>
```

Si la consulta devuelve resultados, ambos conceptos hermanos cumplen las condiciones para este tipo de relación. Si no devuelve ningun resultado, no cumplen estas condiciones.