Escuela Técnica Superior de Ingenieros
UNIVERSIDAD DE SEVILLA

DESARROLLO DE UN SERVICIO MIDDLEWARE DE ONTOLOGÍAS COOPERATIVAS APLICADO A SISTEMAS EMBEBIDOS DE TRANSPORTES INTELIGENTES

Tesis para optar al grado de
Doctor en Electrónica Tratamiento de la Señal y Comunicaciones

Derlis Orlando Gregor Recalde

Director: Dr. D. Sergio Toral Marín

Sevilla - España
2013

Dedicado a...

A toda mi familia, quienes han estado siempre a mi lado apoyando de forma incondicional todos mis emprendimientos y trabajos de investigación…

Agradecimientos

Agradezco a todas las personas que han contribuido de alguna forma en la realización de éste proyecto en especial a mis padres por haber colaborado, apoyado y confiado siempre en mis investigaciones.

A mis abuelos, Pai y Eliseo, quienes hubiesen querido estar presentes en estos momentos. Estarán siempre vivos y presentes en nuestros corazones y mantendrán eternamente el sentimiento de lealtad, amistad y amor en todos nosotros.

A mi hermano Raúl, por haber aportado tiempo y dedicación en guiarme durante mi estancia en Sevilla.

Finalmente no quiero dejar pasar la oportunidad de agradecer profundamente a mi director de tesis, Dr. Sergio Toral por guiarme en varios aspectos profesionales y apoyar, no solo a mí, sino al resto del equipo de investigadores hacia el camino de logros personales.

Gracias…

Índice general

Índice de Figuras

Índice de Diagramas de Secuencias

Índice de Códigos

Índice de Tablas

Resumen

Las primeras soluciones de comunicación entre ordenadores se basaban en un modelo denominado centralizado en la que un único equipo con una o múltiples CPUs (Central Processing Unit) procesaba todas las solicitudes entre aplicaciones. No obstante, la necesidad de obtención de datos complejos, de cálculos entre aplicaciones de forma rápida, precisa y de coste accesible dio lugar a que éste modelo fuera sustituido por un modelo distribuido que permitiese hacer viable el manejo de grandes estructuras de datos diseminados en múltiples ordenadores que se comunican entre sí a través de una red común. La independencia, distribución y en algunos casos la naturaleza heterogénea de estos ordenadores demandan un sistema de software distribuido que sirva como estándar entre sistemas. La utilización de un software middleware entre la capa software y hardware actúa como una capa de abstracción de software distribuido y ofrece un conjunto de servicios que hace posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. La comunicación entre aplicaciones es crucial ya que las tecnologías de la información están evolucionando hacia una interacción de aplicaciones a través de Internet. Inicialmente, las aplicaciones en Internet surgieron para realizar la comunicación entre humanos o entre humanos y aplicaciones. Una muestra de ello son: Email, Ftp, Web, Telnet, IRC, etc., Hoy en día, la necesidad de interacción entre aplicaciones a través de Internet se está demandando de forma creciente. La electrónica, los sistemas informáticos y los sistemas embebidos están cada vez más integrados en entornos distribuidos. El descubrimiento de servicios es un reto bien reconocido en estos entornos. En la actualidad el desarrollo de aplicaciones SOA (Service Oriented Architecture) resulta un paradigma imperante para la integración dinámica de servicios. Una de las arquitecturas pioneras orientadas a servicios es CORBA (Common Object Request Broker Architecture), que ofrece conectividad de un sistema a través de los ORBs (Object Request Broker), parte medular de su estructura. Cuando se invoca a un servicio, se espera que éste resuelva la petición y produzca los mejores resultados con un rendimiento aceptable y ofreciendo algún tipo de QoS (Quality of Service). A pesar de todo, descubrir y localizar servicios no es tarea fácil y las búsquedas comúnmente carecen de inteligencia. El agente que realiza la petición podría obtener una infinidad de información no relacionada o sin sentido. La localización de la información o algún servicio específico podrían realizarse, por ejemplo, a través de algún servicio Trading de CORBA, donde la representación estaría dada por pares de nombres/valor y las propiedades de éstos cubrirían la referencia al servicio en sí, de forma descriptiva. Pero el Trading Service de CORBA utiliza información almacenada en diferentes componentes de su estructura reduciendo la capacidad de trabajar con entornos dinámicos e intuitivos. Otro inconveniente que presenta este servicio de

CORBA es que no es capaz de trabajar con homonimias, obligando al desarrollador a crear sistemas poco flexibles.

El trabajo desarrollado en esta tesis se aplica de forma específica al campo de los ITS (Intelligent Transportation Systems), donde la exactitud y rapidez de la información enviada/recibida es sumamente importante. La información confusa o la falta de interoperabilidad pueden resultar en ocasiones desastrosas, dando lugar a accidentes, retrasos y caos en el tráfico dependiendo del área en que se implemente. El descubrimiento dinámico de la información, así como la composición y la invocación de servicios a través de agentes inteligentes serían una potencial solución a estos problemas. La composición de servicios podría permitir que los servicios cooperen mutuamente de forma dinámica. Para ello es necesario que el flujo de información sea gestionado de forma inteligente. La incorporación de ontologías implementadas en un conjunto estandarizado de conceptos como RDF (Resource Description Framework) extendida por mecanismos de especificación de clases y propiedades como RDFS (RDF-Schema) o escritas en un lenguaje OWL 2 (Web Ontology Language), así como un servicio que gestione ontologías de forma cooperativa, jugarían un rol muy importante en un entorno orientado a los ITS. En esta tesis doctoral, el capítulo 1 realiza una introducción general sobre los inicios, evolución y actualidad de los ITS, exponiendo los principales escenarios de aplicación, los sistemas de comunicaciones más utilizados en este entorno y las principales carencias. En el capítulo 2 se introducen las tecnologías middleware existentes, haciendo hincapié en los principales tipos de middleware actuales. Se trata además de la importancia de las ontologías en las actuales infraestructuras de desarrollo software como parte crucial en el manejo inteligente de la información y más específicamente en el campo de los ITS. También se expone la importancia de los servicios colaborativos inteligentes y las ventajas de la incorporación de un servicio semántico en el campo de los ITS. En el capítulo 3 se hace una propuesta para la creación de entornos inteligentes cooperativos en el ámbito de los ITS. Para ello se propone un método para medir y evaluar la carga computacional de CORBA ejecutada sobre plataformas embebidas. También se propone un método específico para la creación de ontologías en el dominio de los ITS utilizando técnicas de recuperación de la información, extracción del conocimiento, procesado del lenguaje natural y el análisis de datos estadísticos. Por último se presenta la metodología para la construcción de un servicio denominado SemancicService (Servicio de Comunicación Semántico) desarrollado en TAO CORBA (The ACE ORB) en asociación con un conjunto de librerías base (Redland RDF - librdf) que provee el soporte para interactuar con ontologías escritas en RDF y RDFS. Se introduce un analizador sintáctico (Raptor RDF Syntax Library - libraptor2) para estudiar secuencias de símbolos a fin de determinar la estructura gramatical y un lenguaje de sintaxis tipo consulta (Rasqal RDF Query Library - librasqal) que se utilizará para construir y ejecutar consultas. El objetivo principal será el de gestionar la

información de la ontología desarrollada e interoperar con servicios implementados en dispositivos embebidos en el campo de los ITS, (videocámaras, semáforos, paneles de tráfico y dispositivos en-vehículos, dispositivos móviles, PDAs, etc.). El SS (Semantic Service) puede ser implementado indistintamente tanto en entornos embebidos basados en entornos embebidos o bien en entornos PC. Las diferentes aplicaciones de tráfico C/S (cliente/servidor), implementados en su mayoría en dispositivos con arquitecturas embebidas, interoperarán con el SS, que será el encargado de proveer y gestionar toda la información y de intermediar entre ellos de manera dinámica e inteligente. SS actuará entonces como un agente inteligente entre proveedores y consumidores de servicios middleware enfocados a la gestión de los ITS. En el capítulo 4 se muestran los resultados obtenidos respecto al coste computacional de la capa middleware sobre dispositivos ARM (Advanced RISC Machines), el rendimiento y escalabilidad de los métodos propuestos para la creación de la ontologías y los resultados, rendimientos y escalabilidad obtenidos en la implementación del Servicio Semántico, comparándolo con el Servicio Trading de CORBA en un escenario real. Finalmente, en el capítulo 5 se dan a conocer las conclusiones de los diferentes capítulos, así como las principales aportaciones realizadas. Se habla sobre la repercusión que podría tener esta tesis en futuros trabajos y se detallan las publicaciones realizadas durante el proceso de investigación, tanto en revistas científicas como en congresos internacionales.

Capítulo 1 - Sistemas Inteligentes de Transporte

1.1. Inicio y Evolución de los ITS

Los ITS surgen en la década de los años 90 como una alternativa sostenible al problema generado por la creciente demanda de movilidad, especialmente en el ámbito urbano e interurbano. Los ITS suponen una apuesta por la movilidad sostenible frente a los problemas de espacio físico, las estrategias tradicionales implementadas, o el incremento de infraestructuras viales y de vehículos que pueden conducir a niveles de insostenibilidad. ITS se presenta como una apuesta para mejorar la gestión del tráfico sobre la base del aumento de la eficacia y eficiencia del transporte y de proveer seguridad a los usuarios. Los ITS son una combinación de información, comunicaciones y tecnologías del transporte en vehículos e infraestructuras. Una combinación que en los últimos años adquiere una enorme transcendencia puesto que las tecnologías actuales de comunicación permiten emitir información móvil en cualquier lugar y en tiempo real. Los ITS comprenden un amplio rango de nuevas herramientas para la gestión de las redes de transporte y también incrementar la disponibilidad de servicios para los usuarios. Estos sistemas están basados en tres características fundamentales: información, comunicación e integración de la información que constituyen el alma de los ITS, haciendo que todos los actores intervinientes estén mejor informados, coordinados y facilitando la toma de decisiones más inteligentes. El mayor desarrollo de los ITS ha tenido lugar en el transporte aéreo por las necesidades de ordenar el espacio aéreo con sistemas de localización y de control de vuelo. El ferrocarril, por sus especiales características de constituir un sistema guiado linealmente, ha sentido el impacto de los ITS desde su origen por razones de seguridad. Para el transporte marítimo los avances tecnológicos se han centrado en el campo de las telecomunicaciones, como ayudas en la navegación y la localización geográfica. La falta de capacidad en el trazado de rutas fluviales ha sido el elemento motor de estos sistemas, interactuando con los puertos como centros de intercambio de la información. Actualmente los ITS se encuentran en pleno desarrollo creciente en los transportes por carreteras y autovías. El congestionamiento de las vías y accesos en las grandes aglomeraciones urbanas y la gravedad de la accidentalidad creciente han impulsado su implicación en estos procesos, y en ellos está experimentando los desarrollos más acelerados dado el enorme espacio de actuación del que se dispone. Los ITS se consolidan como una herramienta adecuada para facilitar una mayor seguridad, eficiencia, sostenibilidad, comodidad y equidad en las redes de transporte. El futuro de los ITS en la red de autopistas se inscribe en el contexto de una política viaria que estará más orientada hacia la gestión segura, eficiente y ambientalmente sostenible de la red, que a la construcción de nuevas vías. En diciembre de 2008, la Comisión de la Unión Europea adoptó un acuerdo para la puesta en marcha de un plan de acción encaminado al despliegue de los ITS en Europa, en coordinación con otros planes que se están desarrollando a escala nacional. A través de esta unión de proyectos de diferentes países se espera que el

plan de acción europeo contribuya de una manera efectiva a cumplir los objetivos establecidos en la política común de transportes, en relación con tres conceptos clave:

1. *Mejora ambiental*. Reducción de efectos ambientales especialmente en lo que se refiere a la emisión de CO_2 y otros gases de efecto invernadero.
2. *Seguridad*. Reducción de la accidentalidad y mortalidad que aún permanecen en tasas muy elevadas.
3. *Eficiencia*. En términos tanto de atenuación de los problemas patológicos de congestión como en cuanto al consumo energético específico de este tipo de actividad.

Este plan de acción de los ITS establece una serie de áreas prioritarias de actuación sobre las cuales se irán desarrollando medidas de varios tipos. Estas áreas y medidas corresponden a:

1. Óptima utilización de los datos viales, del tráfico y desplazamientos
2. Continuidad de los servicios del tráfico y de la gestión de flotas.
3. Seguridad vial y protección del transporte.
4. Integración del vehículo e infraestructuras de transporte.
5. Seguridad y protección de datos y responsabilidades.
6. Cooperación y coordinación de los ITS en Europa.

En las últimas décadas, hemos sido testigos del crecimiento en las infraestructuras y el aumento constante del tráfico. Optimizar los sistemas de gestión del transporte y la tecnología en éste área resulta imperioso para evitar el aumento de accidentes de tráfico y el aumento de las emisiones nocivas para el ambiente. Sólo en la UE (Unión Europea), el congestionamiento en el tráfico hace algunos años movía cerca de € 50.000 millones al año, lo que representa el 0,5 % del PIB comunitario. El número de vehículos por mil habitantes ha pasado de 232 en el año 1975 a cerca de 500 hoy día. El tráfico total de los vehículos en carretera se ha triplicado en los últimos 30 años y, durante la última década, la cantidad de carreteras para el transporte de mercancías creció un 35 % contribuyendo a unos 7.500 Km. Este impacto creciente hace que la red vehicular se vea afectada por atascos en el tráfico día tras día [1]. Las TICs (Tecnologías de Información y Comunicación) están penetrando prácticamente en todos los sectores, así como en el de los ITS. La integración de los ITS tanto en el área tecnológica como de infraestructura representa un tema clave en el desarrollo y el crecimiento de una ciudad, con el fin de mejorar la calidad de vida de las personas y la seguridad vial. Las nuevas tecnologías que se desarrollan en el entorno de los ITS deben ser aplicadas a actividades de gestión de la movilidad de vehículos, peatones, ciclistas y deben permitir la interoperación entre ellas en base a servicios distribuidos que

cooperen entre sí. Un sistema de tráfico cooperativo haría uso de los datos de operaciones desde el momento de la recopilación de la información, automatizando la toma de decisiones ante situaciones que requieran intervención inteligente del entorno urbano. El aspecto de cooperación se toma desde el punto de vista pasivo o activo de los dispositivos en el momento de la adquisición de los datos y la necesidad de compartirlos con otros sectores y actores del mismo sistema de tránsito urbano. Un plan de tráfico cooperativo representa una solución tecnológica de vanguardia en los diferentes niveles de la seguridad vial. Uno de los principales enfoques de esta tesis es el desarrollo de servicios tecnológicos cooperativos más eficientes, dotando a los dispositivos tradicionales de un mecanismo inteligente de búsqueda de servicios y datos competentes al sector del transporte y la seguridad vial en entornos ITS. La idea principal en la cooperación en sistemas distribuidos ITS se origina con el concepto de la colaboración en la conducción vehicular con los servicios disponibles en las carreteras, donde los vehículos interactúan con el entorno, y el entorno en sí actúa inteligentemente en base a acontecimientos en el tráfico. Las decisiones y la cooperación entre dispositivos se realizan a partir de la recopilación de datos. Sin duda, la incorporación de nuevas tecnologías en vehículos, el acondicionamiento de entornos ITS y de las carreteras es una necesidad. Los ITS, junto con la infraestructura y los equipamientos de la red constituirán elementos esenciales de esa política. Las autoridades del Ministerio de Fomento Español y las Sociedades Concesionarias, en actuación coordinada con las instituciones de la Unión Europea, están resueltamente comprometidas en la evolución de la red hacia un futuro en el que los ITS desempeñarán su misión con plenitud en términos de seguridad, eficiencia y sostenibilidad ambiental [2].

1.1.1. Sistemas V2V y V2I

Uno de los principales retos en los ITS es el tráfico cooperativo. La idea de la cooperación dentro de los ITS se origina con el concepto de la conducción cooperativa que respalda el concepto de autopistas automatizadas, donde los vehículos reciben señales de entrada del entorno vial. Las primeras ideas documentadas sobre carreteras automatizadas ya fueron presentadas en 1960 por el laboratorio de investigación de la General Motors [3]. En su visión, las ruedas delanteras del vehículo son posicionadas automáticamente para responder a señales captadas por los amortiguadores montados en la parte delantera del coche. Hoy en día, el principal objetivo de la conducción cooperativa es enfocarse en la previsión y detección temprana de riesgos. Esto se realiza en mayor medida por medio de sistemas y dispositivos de comunicación inalámbrica. Estos se extienden a lo largo de la carretera, autovía o autopista a la vista de los conductores y advierten de situaciones potencialmente peligrosas. En consecuencia, el objetivo de estos enfoques es el de proporcionar a los conductores la oportunidad de adaptar la

velocidad del vehículo de forma temprana y aumentar la distancia de seguridad entre vehículos en una situación de peligro. Las principales áreas de investigación de los ITS se centran en los siguientes puntos:

- El intercambio de información de tráfico entre vehículos y el entorno vial constituye un amplio campo multidisciplinar de investigación centrado en mejorar la seguridad vial. Las estadísticas basadas en estos estudios indican que la causa principal de la mayoría de los accidentes en carretera es la velocidad excesiva y la lenta reacción del conductor en situaciones críticas [4]. La gestión del tráfico pretende utilizar las capacidades de la red vial en toda su extensión y armonizar el flujo vehicular.
- Gestión de eventos bajo demanda en caso de posibles incidentes o situaciones donde la intervención del sistema cooperativo es necesario [5].
- Sistemas de alerta temprana ante posibles incidentes en carretera [6].
- Apoyo al conductor en carriles de incorporación e intersecciones.

Con el creciente desarrollo de la electrónica y la posibilidad de usar sistemas embebidos con cada vez mayores capacidades de procesamiento, el concepto de cooperación se extendió de la idea original de conducción cooperativa a la de sistemas distribuidos ITS. La idea principal de la cooperación en sistemas distribuidos ITS se basa en la colaboración en la conducción vehicular con los servicios disponibles en entornos urbanos, interurbanos, metropolitanos y rurales, donde los vehículos interactúan con el entorno, y el entorno en sí actúa inteligentemente en base a acontecimientos en el tráfico. Entre los trabajos previos en este sentido cabe destacar el realizado por Mitropoulos y otros [4], que presentan un sistema llamado WILLWARN basado en una conducción cooperativa mediante seguridad electrónica que permita prevenir riesgos mediante aplicaciones de detección "Vehicle-Hazard" a-bordo y comunicaciones V2V/V2I. Con ello se pretende disminuir una de las causas principales de los accidentes en carretera, que es la lenta reacción del conductor en situaciones críticas. Sin embargo, el sistema propuesto por Mitropoulos se enfoca exclusivamente en la gestión de mensajes de alertas al conductor sobre la situación de peligro de forma ad-hoc, dejando de lado la calidad y la representación de la información. Thomas y otros [5], presentan un esquema de predicción para eventos recurrentes y un esquema de detección para incidentes basados en flujos de datos que son recolectados en intersecciones urbanas. Sostienen que es necesaria la gestión de eventos bajo demanda en caso de posibles incidentes, pero no validan los resultados de su análisis con datos reales de detección de incidentes, ni tampoco definen como presentan ni captan la información, ni cómo la procesan o la almacenan. Los principales retos en las arquitecturas distribuidas ITS actuales se basan principalmente en la heterogeneidad de los dispositivos hardware y software, y la heterogeneidad en las redes de comunicación. En lo que tiene que ver con los dispositivos hardware son comunes

los retos de incompatibilidad en la representación de los datos, los problemas de sincronización y la diversidad de controladores. Las aplicaciones y servicios software presentan problemas en la multiplicidad de lenguajes de programación, diferentes versiones de una misma aplicación o servicio, la competitividad entre software propietario y libre, problemas de entendimiento y las base de datos distribuidas. La heterogeneidad en las redes de comunicación se debe a que existe una gran diversidad de protocolos de red, arquitecturas de implementación y la incompatibilidad entre las redes distribuidas con las redes tradicionales.

En Europa, el concepto de conducción cooperativa fue primeramente promovido y dirigido por el proyecto EUREKA del programa CE PROMETEO (1987-1995), [7]. En este proyecto las actividades se centraron primero en la comunicación bi-direccional carretera-vehículo, siguiendo luego con las comunicaciones V2V (Vehicle to Vehicle). Desde entonces, el foco de la industria europea del automóvil fueron los sistemas de seguridad independientes en vehículos tales como el ESP (Electronic Stability Program) lanzado por Bosch en 1995 y más adelante diferentes aplicaciones ADAS (Advanced Driver Assistance Systems) iniciando por los ACC (Control de Crucero Adaptativos) en el 2000. Aunque en EE.UU. y Japón las actividades en la conducción cooperativa continuaban, no fue sino hasta hace unos años que los fabricantes de equipos y proveedores "redescubrieron" el concepto de conducción cooperativa basados en la necesidad de una mayor asistencia al conductor. Recientemente se han realizado una serie de proyectos de investigación y están en marcha varias de las comunicaciones V2V en Europa, Japón y EE.UU. Entre ellos destacan algunos prototipos de aplicaciones de alerta V2V tales como el recientemente demostrado por los sub-proyectos WILLWARN (Wireless Local Danger Warning) e INTERSAFE de PReVENT [7], donde presentan un sistema de alerta al conductor sobre posibles riesgos de choque a través de mensajes. Estos prototipos aún no proporcionan soluciones maduras para su uso comercial. Hoy en día y de común acuerdo, las normas para la comunicación V2V y V2I (Vehicle to Infrastructure) en Europa están en preparación por el ETSI [8] (European Telecommunications Standards Institute), así como los sistemas como C2C-CC [9] (Consorcio de Comunicación Car2Car), establecidas para promover la idea de normas europeas comunes para la comunicación C2X. El C2C-CC es una organización sin ánimo de lucro iniciada por los fabricantes europeos de vehículos y está abierta a los proveedores, organismos de investigación y otros socios. El C2C-CC está dedicado al objetivo de aumentar aún más la seguridad del tráfico vial y la eficiencia por medio de las comunicaciones entre vehículos. La normalización recibió un considerable impulso a través de la asignación de una banda de frecuencia dedicada para la comunicación de seguridad de vehículos en Europa. Se ha reservado 30 MHz entre 5.875 y 5.905 GHz para uso exclusivo de comunicación de la información pertinente a seguridad. Esta asignación de frecuencias está muy en línea con la banda de frecuencias reservada en los EE.UU. para DSRC (Dedicated

Short Range Communication), lo que permite utilizar el mismo conjunto de chips basándose en el estándar IEEE 802.11p. Además de los 35 MHz, reservados para la comunicación de seguridad del vehículo, sólo 40 MHz adicionales se han reservado entre 5.855 - 5.875 GHz y 5.905 - 5.925 GHz para aplicaciones de tráfico, pero no hay un uso exclusivo de éstos para comunicaciones C2X.

La mayoría de los proyectos de investigación relacionados con el área de los ITS son usualmente los relacionados con las comunicaciones V2V y V2I [10]. Entre estos, el proyecto FleetNet [11] y su siguiente proyecto NoW (Network on Wheels) [12], investigan la integración de Internet en redes vehiculares. Esta integración requiere un alto soporte de movilidad, comunicación eficiente, el descubrimiento de servicios y el soporte de aplicaciones heredadas. FleetNet utiliza IPV6 basado en soluciones de direccionamiento. La arquitectura propuesta contiene gateways de Internet fijas a lo largo de la carretera con dos interfaces que conectan las redes vehiculares con Internet.

Otros proyectos trabajan directamente con sistemas cooperativos con el fin de aumentar la seguridad vial y la eficiencia en el tráfico. CVIS (Cooperative Vehicle Infrastructure Systems, http://www.cvisproject.org), SAFESPOT (Cooperative Systems for Road Safety, http://www.safespot-eu.org) y COOPERS (Co-operative Systems for Intelligent Road Safety, http://www.coopers-ip.eu) son algunas de las iniciativas financiadas por el Sexto Programa Marco, [13]. CVIS tiene como objetivo el diseño, desarrollo y testeo de tecnologías necesarias para permitir a los vehículos comunicarse de forma segura con otros vehículos y con la infraestructura en carretera más próxima. Para lograr esto, CVIS utiliza el estándar internacional CALM (Communications Air-interface, Long and Medium range) que aún se encuentra en desarrollo [14]. SAFESPOT tiene como objetivo mejorar la seguridad vial a través de un asistente, "Safety Margin", el cual detecta situaciones críticas con antelación. Safety Margin se define como la diferencia de tiempo entre el tiempo de detección de un peligro potencial y el tiempo real del accidente, si no se ha hecho nada para evitarlo. En SAFESPOT, este concepto puede ser testeado basándose en la concepción de sistema cooperativo, el cual utiliza comunicaciones V2V y V2I y la tecnología IEEE 802.11p. Finalmente, en la visión de COOPERS, los vehículos se encuentran conectados a través de una comunicación continua wireless con la infraestructura en carretera. Estos intercambian datos e información relevantes para cada segmento específico de la autopista o carretera con el fin de incrementar la seguridad y habilitar la gestión del tráfico de forma cooperativa.

1.1.2. Sistemas de gestión del tráfico

Mientras que la conducción cooperativa hace uso de los datos proporcionados tanto por los sistemas de sensores en-vehículos y sistemas de colección de datos en infraestructuras, el concepto de tráfico cooperativo tiene una visión más global del tráfico en sí. Los sistemas de tráfico cooperativos hacen uso de datos, recopilan, transforman y comparten la información en todos los niveles durante viajes y llevan a cabo la toma de decisiones en el transporte. El aspecto de cooperación se inserta en la capacidad de los diferentes actores para adquirir datos y compartirlos con otros sectores y actores del sistema de tránsito de forma pasiva o activa. Las formas elementales del tráfico cooperativo representan hoy día varios subsistemas de gestión de tráfico que emplean técnicas VMS (Variable Message Sings), ampliamente utilizados en los últimos veinte años en carreteras europeas. El objetivo principal de los sistemas de gestión del tráfico ha sido mejorar la eficiencia y la seguridad de la red de carreteras mediante la armonización del flujo de tráfico y la regulación del comportamiento individual del conductor. Los siguientes tipos de subsistemas de gestión del tráfico son algunos de los desarrollados e implementados en diferentes proyectos ITS:

- Control de Velocidad / Control de Secciones de Tráfico (relacionado con el tráfico o con la gestión del clima)
- Gestión de Incidentes
- Advertencias de Peligro / Alertas de Congestionamiento / Alertas de Cola
- Control de Rampas
- Control de la Red de Tráfico incluyendo Cambios de Itinerario
- Control de Carriles
- Control de Carriles Reversibles
- Control de Tráfico en Túneles
- Control de Tráfico en Puentes

Estos sistemas poseen la capacidad de optimizar las carreteras en un 10-40% y reducir accidentes con heridos graves en un 10-40%, dependiendo del sistema y las características de su ubicación. El desarrollo de estos sistemas se ha concentrado en sitios con altos volúmenes de tráfico, que con frecuencia se encuentran afectados por incidentes y en donde el impacto de ellos se espera sea el mayor. Los sistemas de gestión del tráfico son comúnmente controlados a través de una base de datos en tiempo real proporcionado por los sistemas de vigilancia. Los datos típicos requeridos incluyen, por ejemplo, datos del tráfico en cruces e intersecciones (velocidad, volumen y ocupación), datos sobre el estado de las carreteras y la superficie (seco / húmedo / nieve / hielo), datos del tiempo en secciones de carretera (viento y ráfagas, precipitaciones, temperatura, humedad, etc.), detección automática de incidentes, y datos del tiempo de viaje, etc. Los operadores en los

centros de gestión del tráfico operan y supervisan los sistemas basados en estrategias de gestión del tráfico en general, desarrollado para situaciones en donde normalmente los escenarios se repiten. La verificación del estado de la red de carreteras es la responsabilidad de estos operadores y los eventos son captados casi siempre a través de cámaras CCTV. La gestión del tráfico entre fronteras también ha sido implementada. Las soluciones de gestión del tráfico han sido en su mayoría basadas en tecnologías y soluciones muy maduras. Los programas Europeos en I+D en realidad no han incluido grandes proyectos desde mediados de 1990. Las principales novedades en los planes de gestión del tráfico, centro de operaciones, la incorporación de las VMS y la evaluación del sistema desde entonces han tenido lugar dentro de los proyectos denominados Euro-Regionales apoyados por el programa TEN-T de la Agencia Ejecutiva de la Red Transeuropea de Transporte de la Unión Europea. En la actualidad, estos proyectos han sido integrados en un periodo de siete años iniciales y en tres fases estableciendo objetivos hasta el 2020 [Figura 1] con la participación de 23 países europeos. El presupuesto total para el desarrollo del proyecto de la Unión Europea, EasyWay, es de 1.500 millones de euros entre el período 2007-2013 [15].

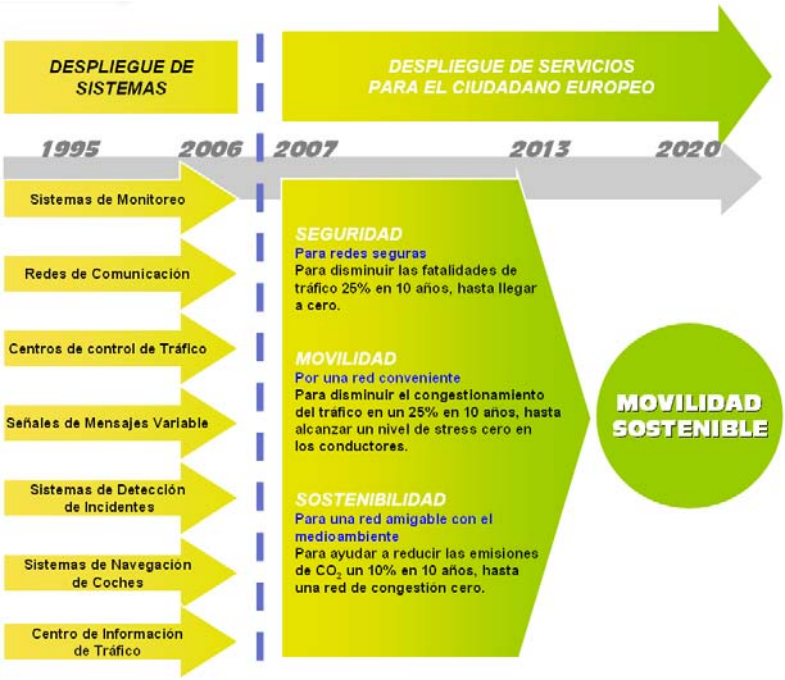


Figura 1. Desarrollo del sistema EasyWay

EasyWay ha establecido objetivos claros en su programa, y proporciona el conjunto de servicios ITS necesarios para implementar en toda Europa tales como: Servicio de Información al Viajero, Servicios de Gestión del Tráfico, Servicios de Carga y Servicios TICs conectados, entre otros. La infraestructura EasyWay es una plataforma eficiente que permite a los actores europeos movilidad para lograr un despliegue coordinado y combinado de estos servicios paneuropeos.

El desafío para la industria es presentar la próxima generación de sistemas ubicuos, redes e infraestructuras para servicios de comunicación, la informática y los medios de comunicación para facilitar la toma de decisiones en viajes. Estas infraestructuras nuevas de comunicación permitirán la aparición de una gran variedad de tecnologías, servicios, modelos e información de negocio capaz de conectarse en todas las ramas de la tecnología de forma dinámica y transparente con múltiples dispositivos, redes, proveedores y servicios en el tráfico cooperativo.

1.1.3. Entornos de aplicación de los ITS

Las aplicaciones implementadas en áreas metropolitanas son aquellas donde los programas se ejecutan principalmente en áreas urbanas y suburbanas. El tráfico urbano es uno de los problemas que más influyen en la calidad de vida de los residentes en las ciudades y áreas metropolitanas. Uno de los problemas acentuados en las últimas décadas por el desarrollo es el uso del automóvil privado en detrimento del transporte público. Otro de los problemas no menos importantes es la generalización del modelo de ciudad difusa, una tipología urbana genuinamente norteamericana pero imitada en Europa por las grandes ciudades y las ciudades de tamaño medio que han alcanzado su desarrollo actual tras un proceso descentralizador. Ofrecer opciones de transporte público planeadas de manera inteligente para zonas metropolitanas de crecimiento rápido y caótico puede reducir en gran medida el congestionamiento en el tráfico, ahorrando costos y reduciendo emisiones nocivas para la salud. Este método evidentemente debe ir acompañado de una estrategia en el campo de los ITS.

Aunque las zonas rurales representan pequeñas porciones de la población, éstas son una parte importante del sistema de transporte. El ochenta por ciento del total de carreteras y autopistas se encuentra en las zonas rurales en los EE.UU. Se estima que estos trayectos vehiculares representan un 40% en kilómetros recorridos. A diferencia de las zonas metropolitanas, el medio rural tiene un conjunto diferente de prioridades y necesidades que se reflejan en distancias más largas y menor volumen de tráfico. Además, los conductores no están familiarizados con el entorno y los tiempos de respuesta en situaciones de emergencias representan un verdadero desafío. Muchos de los servicios ITS implementados en áreas metropolitanas

también pueden ser aplicados en el medio rural. La iniciativa de incorporar las tecnologías ITS en áreas rurales representa una alternativa relativamente nueva, con aumentos en niveles de actividad en los últimos años. El objetivo principal de la implementación de los ITS en áreas rurales es el de proveer mayor seguridad y protección en respuestas a emergencias.

Los vehículos comerciales también están experimentando grandes beneficios gracias a la implementación de los ITS. Las mejoras en la eficiencia administrativa, la inversión en infraestructura y mejoras en carreteras así como la recopilación de datos, benefician en gran medida la seguridad y reduce costos de operación. En la actualidad, muchas empresas están equipando sus propias flotas con sistemas personalizados que proporcionan grandes ventajas competitivas. Los recientes avances en ITI (Intelligent Transportation Infraestructure) en relación con los CVO (Commercial Vehicle Operations) y la participación de sectores privados permiten la difusión de las tecnologías ITS en este campo. Esta tendencia incluye una serie de tecnologías y sistemas que favorecen el control y las comunicaciones a bordo de vehículos comerciales. Los dispositivos de comunicación y gestión de flota incluyen comunicaciones por satélite, teléfonos móviles, beepers, sensores, ordenadores a bordo, etc.

1.2. Comunicación de los ITS

La tendencia de los ITS claramente se basa en la integración de todas las comunicaciones de datos en una única red de comunicaciones. Esta evolución ha alcanzado ya el mercado en el caso de redes fijas de comunicaciones, pero en lo que respecta a la integración con las comunicaciones móviles su desarrollo todavía es reducido. Desde el surgimiento de los ITS, varias tecnologías inalámbricas han sido propuestas. Las primeras comunicaciones en rangos de acciones cortos y largos fueron los radio modems en frecuencias UHF y VHF. Las comunicaciones de corto alcance (menos de 500 metros) se pueden lograr implementando protocolos IEEE 802.11 [16], promovidos por la Sociedad de Transporte Inteligente de Norte América y el Departamento de Transporte de los Estados Unidos. En Europa, estos sistemas de rangos cortos y los ad-hoc están regulados por el CEN DSRC [17] (Comité Europeo para la estandarización) dedicado a las comunicaciones en rangos cortos Europeos en 5.9 GHz, WLAN (Wireless LAN) e infrarrojos. En agosto de 2003, el tren *Altamont* que opera en el área de San Francisco, se convirtió en el primer tren de cercanías en implementar este standard [18]. En teoría, la gama de estos protocolos se puede ampliar utilizando redes móviles ad-hoc o redes Mesh. Un amplio rango de comunicaciones han sido propuestos utilizando redes de infraestructuras WiMax IEEE 802.16 [19], GSM (Global System for Mobile Communications) o 3G. Las comunicaciones en distancias largas utilizando estos

métodos han sido bien establecidas, pero a diferencias de los protocolos e implementaciones en rangos de comunicaciones cortas, éstos requieren el despliegue de una mayor infraestructura y, por lo tanto, un mayor coste. Hasta el momento existe una gran incertidumbre en torno al modelo de negocio a ser implementado en infraestructuras ITS. Los recientes avances en tecnología inalámbrica de comunicaciones de V2V y V2I se han convertido en la piedra angular de los ITS. Las comunicaciones inalámbricas vehiculares en los ITS es uno de los temas de investigación más activos. Según el ETSI [20], los ITS incluyen sistemas telemáticos y todo tipo de comunicaciones intra-vehiculares, inter-vehiculares (V2V) y entre vehículos e infraestructuras fijas (V2I). Sin embargo, los ITS no están restringidos solamente a sistemas de transportes en carretera. Estos también incluyen el uso de las TICs en redes de ferrocarriles, transportes marítimos y aéreos, incluyendo sistemas de navegación Figura 2.

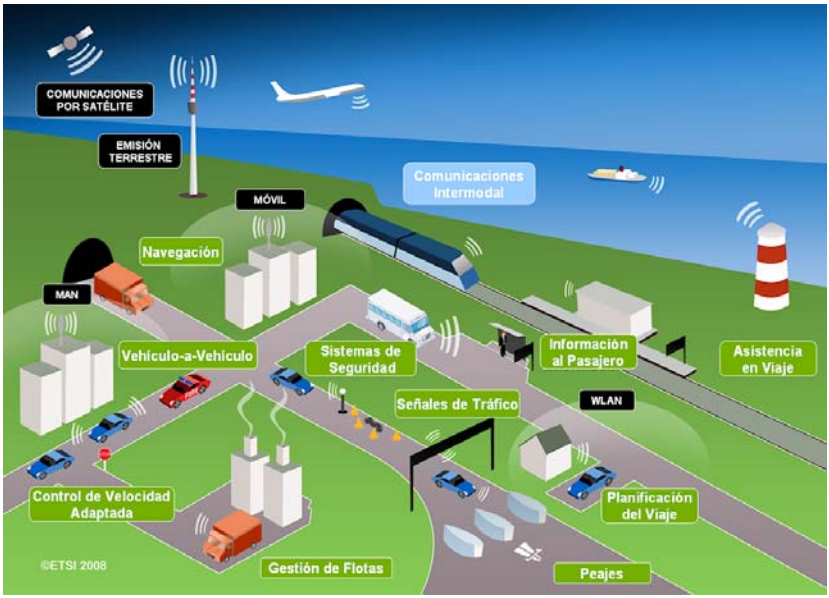


Figura 2. Escenario de comunicación ITS según la ETSI

Dependiendo del escenario de implementación, todos los actores intervinientes dentro del esquema expuesto por el ETSI pueden ser compuestos arbitrariamente para formar una cooperación entre ITS. Estas entidades pueden comunicarse unas con otras utilizando varias redes de comunicaciones diferentes. Las estaciones fijas de ITS ofrecen básicamente la capacidad de comunicación así como la implementación de la infraestructura. Un vehículo equipado con un hardware de comunicación es capaz de establecer contacto con otros vehículos o

infraestructuras. Los componentes de la infraestructura vial como los VMS o luces de tráfico también están equipados con estos dispositivos de comunicación. Los dispositivos personales no quedan exentos dentro de esta gran red. Los teléfonos móviles y otros dispositivos de navegación personal también proveen numerosas aplicaciones ITS, y pueden soportar aplicaciones ITS cooperativas basadas en comunicaciones con otros usuarios o infraestructuras dentro del escenario. De esta manera todo el entorno está conectado. En el campo de las ITS, las tecnologías cableadas e inalámbricas son utilizadas ambas en conjunto. Las cableadas son comúnmente utilizadas en infraestructuras de cableado internos mientras que las inalámbricas son más utilizadas en sistemas externos. Desde el punto de vista de las entidades de comunicación, la arquitectura de comunicación ITS consta de cuatro entidades principales: Vehículos, Equipamiento de Carreteras, Equipamientos Centrales y Dispositivos Personales. Cada una de las cuatro entidades contiene una estación ITS y por lo general una puerta de enlace de conexión a la estación ITS. Una estación ITS comprende una serie de funciones específicas y un conjunto de dispositivos de aplicaciones. Dependiendo del escenario de implementación, estas cuatro entidades se pueden componer arbitrariamente para formar ITS cooperativos [21].

1.2.1. Arquitectura de Comunicación

Desde la perspectiva de comunicación, una estación ITS está basada en las capas de referencias mostradas en la Figura 3 [22].

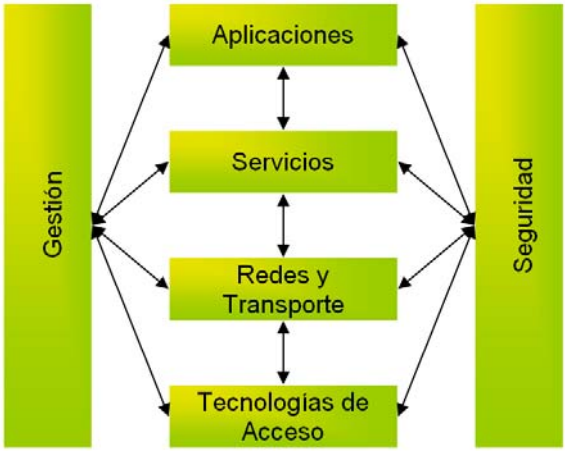


Figura 3. Arquitectura de comunicación ITS

Esta arquitectura de comunicación consiste en cuatro capas horizontales: las tecnologías de acceso, redes y transporte, servicios y aplicaciones. Estas capas horizontales están flanqueadas por una capa de gestión y una capa de seguridad. La capa de aplicaciones es una parte fundamental en los ITS cooperativos. Es importante asegurarse de que cada aplicación que se encuentra ejecutándose en la misma estación ITS utiliza los mismos datos e información para garantizar la consistencia y cierta QoS. Para ello, la capa de Servicios integra un middleware y un repositorio donde se almacena la información (Base de Datos o Ficheros). Esta capa se sitúa entre la capa de aplicaciones y la capa de red y transporte. El objetivo de la capa de red y transporte es el transporte de datos entre estaciones ITS. En la capa de tecnología de acceso se incluyen tanto las tecnologías cableadas (utilizados principalmente en estaciones internas) como inalámbricas (CEN DSRC European 5.9 GHz, WLAN, Infrarrojos, WiFi, WiMax, GSM/GPRS, UMTS) así como sistemas broadcast digitales como DAB (Digital Audio Broadcasting), DMB (Digital Multimedia Broadcasting), DVB-T (Digital Video Broadcasting-Terrestrial), DVB-H (DVB-Handheld) y GPS (Global Positioning System). La capa de Seguridad habilita un amplio rango de funcionalidades para dotar aplicaciones seguras. Con la nueva era de los ITS los requerimientos de seguridad son indispensables y deben ser considerados en todo su entorno para prevenir ataques ilegales al sistema. Algunos de los sistemas de seguridad implementados podrían ser firewalls, mecanismos de gestión de intrusiones, servicios de autenticación y autorización, servicios de privacidad, gestión de claves, servicios de seguridad hardware, etc. La capa de gestión es la responsable de todas las aplicaciones, redes e interfaces en implementaciones específicas. Las tareas comunes de la capa de gestión son las de gestionar las políticas, las interfaces dinámicas, los criterios de decisiones, requerimientos, medir el rendimiento, gestionar los permisos y las sincronizaciones de acceso en base a prioridades, gestionar la seguridad y privacidad dependiendo de las aplicaciones, servicios, etc. [21]. Msadaa y otros [23] realizaron una evaluación de rendimientos comparando el rendimiento entre dos tecnologías móviles, 802.11p y 802.16e en un contexto V2I. En el estudio se presentaron las principales limitaciones de ambas tecnologías de comunicación comparándolas bajo diferentes criterios, midiendo los rendimientos bajo diferentes velocidades así como la información de las tasas en el tráfico de datos teniendo en cuenta la infraestructura de red. Los resultados de las simulaciones presentadas por Msadaa muestran por un lado la gran competitividad de la tecnología móvil WiMAX en el contexto de las comunicaciones V2I. En particular, esta tecnología ofrece no sólo una gran cobertura de radio, sino también retardos muy bajos. Por otro lado, la tecnología 802.11p se adapta mejor en transmisiones de datos con baja carga, donde presenta latencias muy bajas, incluso con altas velocidades del vehículo.

1.3. ITS en sistemas distribuidos

Los servicios distribuidos en los ITS están empezando a ser incorporados dentro de los procesos de planificación donde sólo se tenían en cuenta servicios con capacidades tradicionales. Varias estrategias descentralizadas multi-agentes han atraído considerablemente la atención de varias investigaciones. Daneshfar y otros [24] presentaron un enfoque multi-agente para el control de señales difusas de tráfico descentralizado en donde plantean la gestión eficiente del tráfico de acuerdo con el estado del mismo en intersecciones y las condiciones generales del tráfico. Brickley y otros [25] presentaron una estrategia en la diseminación de datos para sistemas cooperativos entre vehículos, exponiendo que las políticas de difusión cumplen un rol determinante en las ITS para la propagación eficiente de la información. Rockl y otros [26] exponen que el éxito de las aplicaciones en los ITS cooperativos se encuentra principalmente en el intercambio de información entre los nodos distribuidos. Según Rockl, la transmisión de grandes flujos de información contrasta con el ancho de banda limitado de los canales que tiende a ser compartido por todos los nodos participantes en los ITS. Proponen que la solución radica principalmente en el compromiso cooperativo para seleccionar piezas relevantes de información para su difusión de acuerdo a su valor. Estos también sostienen que la prioridad de difusión de la información se puede conseguir mediante una evaluación basada en la entropía de un sistema con filtros probabilísticos dinámicos.

Dado el amplio rango de tecnologías utilizadas para implementar estos servicios y los diferentes tipos de implementación, en muchos casos resulta difícil predecir los potenciales impactos de los servicios ITS planeados para áreas particulares. Mientras más distribuidos se tornen los servicios, más difícil será medir el impacto por separado y las capacidades individuales de cada uno. Cada vez más es necesaria la incorporación de agentes inteligentes en esta área. Un ambiente de transportes inteligentes puede definirse como aquel donde el entorno del tráfico es equipado con las facilidades necesarias para interoperar y adaptarse a las necesidades de los usuarios en base a eventos. Sin embargo, en la actualidad, este entorno aún se ve afectado por la heterogeneidad de la información y la inconsistencia de los datos. Las aplicaciones y dispositivos se encuentran inmersos en sistemas distribuidos muy complejos. Los sensores y las arquitecturas de redes implementados con el entorno de los ITS son insuficientes para proveer servicios autónomos y proactivos [27]. Los dispositivos y aplicaciones de diferentes proveedores y vendedores necesitan una forma flexible de trabajar de forma colaborativa entre sí. Deben ser capaces de descubrir servicios de manera autónoma, coordinar el trabajo, resolver conflictos en base a situaciones críticas, negociar con usuarios y adaptarse a cambios del propio entorno. La implementación de tecnologías inteligentes mejoraría de manera significativa las características y

funcionalidades del entorno cooperativo urbano. El descubrimiento del conocimiento y la explotación de la información recabada podrían ser organizados estableciendo servicios distribuidos que procesen los datos de clientes/servidores para analizar los procesos de acción y el intercambio de eventos o datos. Los ITS deben estar preparados para aprender del entorno y modificar sus características en base a eventos. Deben ser capaces de acceder a información de recursos externos, por ejemplo, localizados en Internet. A través de la globalización de los datos actuales, los servicios deben ser capaces de compartir información de tráfico reutilizando conocimientos y combinarlos si fuera necesario para obtener una información más detallada. La comunicación colaborativa entre conductores en las carreteras debe ser automatizada para ofrecer mayor rendimiento. Los servicios deben poder combinarse en situaciones críticas interactuando con el entorno y los usuarios para tomar decisiones efectivas.

1.4. Problemática de los Sistemas de Transporte actuales

A medida que el número de vehículos y los kilómetros de carreteras crecen, van adquiriendo un grupo heterogéneo mayor de ordenadores, dispositivos, aplicaciones, servicios, redes de trabajo, bases de datos, etc. Para tener éxito en un entorno que cambia a gran velocidad, la información debe presentarse de forma rápida y fiable mediante aplicaciones que se extiendan por una red de estaciones de trabajo, servidores y servicios implementados. El congestionamiento es un problema común para miles de conductores. Cada vez más, los usuarios acuden a Internet para informarse antes de emprender algún viaje, o a sistemas de navegación para evitar atascos. En este sentido, es necesario un mecanismo que ofrezca al usuario información fluida en tiempo real de todo el entorno, y que el mismo entorno sea capaz de informar al usuario de cualquier evento cuando éste se genere. Es necesaria la adaptabilidad de la información. Este tipo de sistema no se podría lograr con los actuales sistemas centralizados que sólo hacen uso de los datos para ofrecer informaciones estáticas y programadas. La construcción de información en tiempo real y la adaptabilidad del entorno sobre sistemas distribuidos en base a las condiciones del tráfico pueden resultar de gran utilidad para los usuarios para evitar accidentes, retenciones, puntos negros, minimizar tiempos de trayecto, etc. Por ejemplo, un sistema de gestión de incidentes en entornos urbanos puede influir directamente sobre respuestas de emergencias, proporcionando información oportuna y precisa sobre la información del incidente y la gravedad del caso. La mayoría de la información en los ITS se clasifica por áreas específicas y la infraestructura que la soporta. Es necesario desarrollar alternativas con adaptabilidad heterogénea, suficientemente eficientes, e incrementar la capacidad efectiva de los ITS para mejorar la eficacia de los sistemas de transporte. En la mayoría de los casos, la información es heterogénea, y no provee los datos

necesarios para cumplir con determinadas tareas. El principal reto en el diseño y operación de los ITS es que se requiere proveer a los usuarios de una completa y formal transparencia del entorno sin importar las características y capacidades de los componentes conectados al sistema. En las arquitecturas distribuidas la información heterogénea se pueden encontrar tanto en las redes, los dispositivos hardware como en las aplicaciones software. En este contexto, podemos clasificar las debilidades de la siguiente forma:

1. Heterogeneidad en dispositivos hardware
   - Incompatibilidad en la representación de los datos
   - Problemas en la sincronización
   - Controladores de dispositivos diferentes
   - Incompatibilidad en la representación de la información
   - Protocolos de comunicación e interfaz de trabajo

2. Heterogeneidad en aplicaciones y servicios software
   - Problemas de compatibilidad
   - Diferentes lenguajes de programación
   - Diferentes versiones de una misma aplicación o servicio
   - Competitividad entre software propietario y libre
   - Problemas de entendimiento entre software centralizados
   - Bases de datos distribuidas

3. Heterogeneidad en redes
   - Diferencias entre los medios de comunicación
   - Diferentes protocolos de red
   - Diversas arquitecturas de implementación
   - Dificultad en el momento de trabajar en redes distribuidas

Distintas aplicaciones software, desarrolladas en diferentes lenguajes de programación y ejecutadas sobre cualquier plataforma hardware, deben ser capaces de utilizar servicios para intercambiar información en diferentes arquitecturas y protocolos en las redes. Uno de los factores críticos detrás de la heterogeneidad de los sistemas de software diseñados y desarrollados aisladamente es la diversidad en los modelos básicos implementados. Los principales obstáculos a vencer en la interconexión de sistemas heterogéneos es el intercambio de la información, lo cual resulta una tarea difícil en sistemas altamente distribuidos donde la complejidad aumenta. La interoperabilidad de toda esta infraestructura se consigue mediante la adopción de estándares abiertos y políticas adecuadas en el intercambio inteligente de la información.

En esta tesis se propone la incorporación de tecnologías distribuidas middleware que actúen de soporte a la heterogeneidad de la información y que se apoyen en ontologías como forma de solucionar la problemática de los sistemas de transporte actuales y la comunicación entre dispositivos en el ámbito de los ITS.

1.5. Sistemas Embebidos en entornos Urbanos

Los modernos equipos de transporte instalados en la actualidad tienen el potencial de proporcionar aplicaciones ITS como sistemas avanzados de gestión del tráfico, información al viajero, operación de vehículos comerciales o sistemas de gestión de emergencias [28]. Por lo general se basan en plataformas embebidas que se ejecutan en sistemas operativos embebidos [29], [30]. Este sistema operativo embebido puede ser visto como la interfaz entre la aplicación y el hardware embebido, y debe ser adaptado para las funciones del dominio específico. Sin embargo, una vez instalado, se facilita el uso de redes y capacidades de multitarea [31]. Muchos sistemas operativos de propósito general han sido modificados para disminuir su tamaño y adaptarse a arquitecturas de procesadores embebidos. Entre las diferentes posibilidades de los sistemas operativos disponibles, Linux está firmemente en el primer lugar como el sistema operativo de elección para sistemas embebidos gracias a su escalabilidad, bajo costo y las grandes capacidades de red [30], [32]. En particular, la pila del Linux IP, ha demostrado ser estable, eficaz y muy adecuado para aplicaciones de redes y servicios que requieren alta fiabilidad y disponibilidad [33]. Estas funciones se pueden utilizar para transformar el entorno urbano en una red distribuida de sistemas embebidos, ofreciendo servicios avanzados a los administradores de tráfico, vehículos, usuarios o también a otros equipos embebidos conectados a la misma red urbana. Los sistemas distribuidos aplicados en entornos urbanos incluyendo el incremento en la capacidad de las redes actuales, reducen el congestionamiento y la polución, acortando la duración en los viajes, mejorando la seguridad vial a través de logísticas más eficientes, optimizando la gestión y el control de la red de carreteras (tanto urbanas como interurbanas), aumentando la eficacia de los sistemas de transporte publico y ofreciendo respuestas eficientes frente a potenciales peligros, incidentes y accidentes [34], [35].

Capítulo 2 – Tecnologías Middleware y Ontologías en los ITS

2.1. Tecnologías Middleware

En los últimos años, los sistemas de desarrollo para microprocesadores, microcontroladores, FPGAs entre otros, proporcionan extensas librerías y la posibilidad de trabajar con sistemas operativos embebidos, facilitando en gran medida el desarrollo de aplicaciones. La flexibilidad de los sistemas embebidos permite asimismo que puedan adaptarse a futuras necesidades con un costo mínimo [36]. Además, sus posibilidades de comunicación a través de redes digitales plantean nuevos retos en el ámbito de los sistemas distribuidos embebidos. La tendencia de los ITS hacia el desarrollo de sistemas DRE (Distributed Real-time and Embedded) ofrece muchas ventajas. Los componentes DRE proporcionan un alto nivel de abstracción en sistemas operativos, lenguajes de programación y la integración del middleware como objeto de computación distribuida en todo tipo de entornos ITS. Las caídas del sistema, la necesidad de detección de fallos y la correcta adaptación de los sistemas en tiempo real requieren la adaptación constante de los sistemas DRE en entornos ITS y sus recursos para lograr una continua operación en misiones críticas. Por otra parte, las redes de sensores han madurado bastante en la industria automotriz. Un vehículo moderno tiene alrededor de 70 ordenadores a bordo interactuando entre sí, lo que ha dado lugar a las llamadas redes de sensores vehiculares (VSN). En la actualidad, estas redes se empiezan a extender para ofrecer servicios de comunicación entre vehículos, y entre entornos externos para dar soporte a los ITS. La integración de esta tecnología sería imposible sin la incorporación de los middleware, que se presentan como una metaestructura de futuro para los ITS y los SOA [27].

Los Middlewares representan en esencia un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Permiten abstraer la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando unas APIs (Application Programmer Interface) para la fácil programación y manejo de aplicaciones distribuidas. Dependiendo del problema a resolver y de las funciones necesarias, serán útiles diferentes tipos de servicios middleware. En arquitecturas C/S, el middleware está implementado por el SO (Sistema Operativo) subyacente, que posee las librerías que implementan todas las funcionalidades para la comunicación a través de la red. La capa middleware debe ser capaz de traducir la información de una aplicación y pasarla a otra de forma transparente al desarrollador. La Figura 4 muestra como el middleware sustituye la capa de sesión, de presentación y de aplicación del modelo de capas OSI. Algunas de sus principales bondades son:

1. Simplifica el proceso de desarrollo de aplicaciones.
2. Es el encargado del acceso a los datos: acepta las consultas y datos recuperados directamente de la aplicación y los transmite por la red. También es responsable de enviar de vuelta a la aplicación los datos de interés y de la generación de códigos de error.
3. El middleware debe ser capaz de manejar todas las posibilidades que posee el sistema operativo. Por eso, muchas veces se pierde potencia con la utilización del middleware en lugar de las `APIs` del sistema operativo directamente, pero a cambio se gana el poder trabajar con sistemas heterogéneos.
4. Soporta las interacciones entre cliente y servidor:

   - Protocolos de transporte:
     `TCP/IP, NetBIOS, IPX/SPX, SNA`
   - NOS - Network Operating Systems:
     `Mensajes, RPC, Seguridad, Ficheros`
   - DSM - Distributed System Management (agentes que envían información desde todos los nodos de la red `C/S` para su recogida y presentación).
     `SNMP, CMIP, NIS, SMS`
   - Soporte a servicios específicos:
     `HTTP, IIOP, SMTP, ODBC`

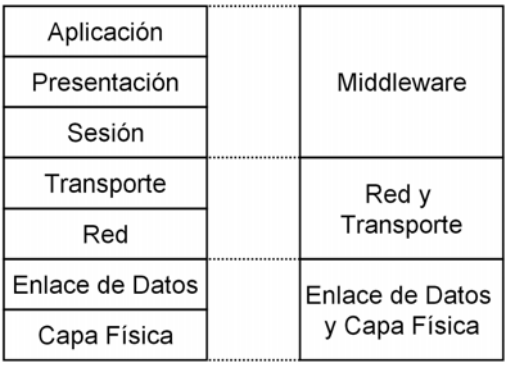


Figura 4. El modelo OSI y el Middleware

Basado en normas significativas o productos en el mercado, el componente middleware puede ser dividido en varias categorías, como el `Socket RPC` (Remote Procedure Call), `RMI` (Remote Method Invocation) de `JAVA`, `DCOM` (Distributed Component Object Model) de `Microsoft` y `CORBA`.

2.1.1. Tipos de Middleware

A continuación se detallan varias de las tecnologías middleware más utilizadas.

2.1.1.1. Remote Procedure Call

RPC o Llamada a Procedimiento Remoto es un tipo de middleware que permite a los clientes utilizar servicios de los servidores llamando a sus procedimientos de llamadas de una manera similar a los se encuentran en programas normales. El RPC es el más básico entre todas las tecnologías middleware. Las RPC son muy utilizadas dentro del paradigma C/S siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función, y envíe éste de vuelta el resultado de dicha operación al cliente. Hay distintos tipos de RPC pero ninguno de estos es compatible entre sí. La mayoría de ellos utilizan un lenguaje de descripción de interfaz IDL (Interface Definition Language) que define el lenguaje utilizado para describir la interfaz del componente del software. Se trata de un lenguaje neutro que permite la comunicación entre componentes de software que no comparten un mismo lenguaje, por ejemplo, entre componentes escritos en C++ y componentes escritos en JAVA. La fuerza de RPC radica en su facilidad de uso y robustez [37], sin embargo su debilidad principal es su poca portabilidad y escasa capacidad para resolver problemas en sistemas heterogéneos.

2.1.1.2. Remote Method Invocation

RMI o Invocación de Método Remoto es un middleware basado en JAVA que permite a los métodos de objetos JAVA, localizados en una Máquina Virtual JAVA (JVM), ser invocados por otro JVM incluso cuando este JVM sea a través de una red. RMI fue introducido por JAVASoft con JDK 1.1 y es esencialmente un JAVA RPC orientado a objetos. Al igual que los RPCs, utiliza IDL para describir la interfaz del software del sistema distribuido (Ej. métodos para RMI y procedimientos para RPC). IDL también es utilizado para producir el stub y el skeleton [38], funciones que permiten que al momento de ser invocadas, éstas puedan ser simuladas localmente. No obstante, RMI tiene algunas diferencias respecto a los RPC. La primera es que el IDL en RPC se basa usualmente en procedimientos C, en tanto que IDL en RMI está basado en JAVA. Sin embargo la naturaleza basada en JAVA juega en contra de RMI ya que resulta poco probable que un sistema que depende de una máquina virtual pueda conseguir rendimientos aceptables en dispositivos embebidos.

2.1.1.3. TP Monitors

Los Monitores de procesamiento de transacciones son los más antiguos y los más conocidos entre los middleware, y están entre las primeras generaciones de aplicaciones para servidores. Son programas que hacen seguimiento a una transacción cuando pasa de una fase de un proceso a otra. EL TP Monitor se asegura que la transacción concluya exitosamente o toma las acciones pertinentes cuando ocurre un error. Son sistemas que proveen procesamiento transaccional a grandes aplicaciones con varios clientes. La función principal de los TPMs es integrar sistemas y administrar recursos. A diferencias de otros modelos, los TPMs son orientados a procedimientos. Proporcionan un mecanismo para facilitar la comunicación entre dos o más aplicaciones. Su mayor beneficio es el rendimiento en función al balanceo de carga, lo que les permite responder a varias transacciones [39]. Sin embargo, presentan dificultades a la hora de intercambiar información ya que todo lo maneja como un paradigma de transacciones, forzando al desarrollador a trabajar con ellos. No importa el tipo de operación o proceso que se esté llevando a cabo, todo deberá ser incluido en una sola transacción [40].

2.1.1.4. Servicios Web

Existen múltiples definiciones sobre lo que son los WS (Web Services), lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican. Podríamos hablar de ellos como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web. Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar [41]. La insatisfacción con diversos aspectos de RPC, RMI y CORBA lleva a los desarrolladores a buscar lo mejor de todas ellas (la multiplataforma de alto nivel, la mejor interpretación entre sus lenguajes, etc.). Los WS se liberan del hardware o de lenguajes de programación utilizando el XML (Extensible Markup Language) para representar datos. Hay analizadores XML disponibles para todo, desde sistemas embebidos hasta los superordenadores, y para casi todos los lenguajes de programación. La ubicuidad de los analizadores XML y la neutralidad de la plataforma estándar XML significa que los WS de los desarrolladores no tienen que preocuparse por las cuestiones de orden de bytes y el tamaño que tendrán los datos,

los cuales son un obstáculo importante para los diseñadores de Sun RPC. Para alejarse de la complejidad de los ORBs, sockets, y todo tipo de problemas de conectividad, los WS se basan en la parte superior del protocolo de transferencia de hipertexto (http). Http también es ubicuo, con servidores Web disponibles para casi todas las plataformas. Un servidor está identificado por una dirección URL (Uniform Resource Locator), y la gestión de la comunicación simplemente se convierte de un problema de mapeo de una llamada de procedimiento a una petición y respuesta http [42]. Los WS son con frecuencia confundidos como la última tecnología en objetos distribuidos. Muchos creen que los WS son sistemas distribuidos que se basan en algún tipo de tecnología de objetos distribuidos. Dentro del mundo de la tecnología distribuida, es más apropiado asociar a los WS con las tecnologías de mensajería debido a que comparten una misma vista de diseño a pesar de que son diferentes tipos de aplicaciones. Dado que los WS se basan en documentos XML y el intercambio de documentos, se podría decir que su base tecnológica es la computación orientada al documento. La tecnología de los WS es bastante simple; está diseñada para mover documentos XML entre los procesos de servicios utilizando protocolos estándar de Internet. Esta simplicidad ayuda a los WS lograr el principal objetivo, la interoperabilidad. La simplicidad también significa que hay que añadir otras tecnologías para construir complejas aplicaciones distribuidas. El objetivo de los WS es proporcionar una plataforma interoperable, por lo que el mercado no toleraría soluciones propietarias [43]. Los proveedores de WS, utilizan los WSDL (Web Services Description Language) [44] para describir los servicios que prestan y la forma de invocarlos. Los proveedores de servicios luego registran sus servicios en un registro de servicios públicos utilizando una descripción universal, descubrimiento e integración (UDDI) [45]. Descubren los servicios de programas de aplicación en el registro y obtienen una URL para el archivo WSDL que describe el servicio. Las aplicaciones pueden invocar los servicios usando XML basados en SOAP (Simple Object Access Protocol) [46], [47]. Los WS son una tecnología middleware adecuada cuando se pretenden desarrollar arquitecturas orientadas a servicio SOA. En estas arquitecturas distintos componentes interactúan intercambiando mensajes para conseguir llevar a cabo una tarea compleja. Gracias a la utilización de los WS, que pueden ser accedidos remotamente, componentes distintos pueden ejecutarse en distintas máquinas distribuidas por toda la red. Actualmente este tipo de arquitectura goza de gran acogida en entornos B2B (Business to Business).

2.1.1.5. SOA (Service Oriented Architecture)

En los últimos años, los SOA han sido ampliamente considerados como un método prometedor en la gestión de redes distribuidas y en redes de comunicaciones heterogéneas a gran escala [48], [49]. Los SOA no representan una

tecnología ni un producto software o una plataforma de WS. Es una forma de construcción de sistemas de información basada en servicios que se integran para satisfacer necesidades de los procesos de negocio [50]. El servicio se proporciona mediante la cooperación de piezas software que realizan una función concreta. La clave de la arquitectura SOA está en la exposición de interfaces abstractas que aíslen la implementación particular de cada pieza de software. Para conseguir este objetivo resulta especialmente útil el uso de WS basados en los estándares SOAP, XML y WSDL. Sin embargo, hay que tener en cuenta que es posible tener WS y no por ello tener SOA, dado que SOA es un método de diseño y no de aplicación de una determinada tecnología. La intención principal de los SOA fue el deseo de pasar de la comunidad de las IT (Information Technology) a aplicaciones estrechamente ligadas al diseño para satisfacer necesidades específicas de funcionamientos más modulares, interoperables y con funciones reutilizables [51], [52]. Algunas de sus principales características son:

- La interacción con los servicios es desacoplada.
- Puede involucrar procesos de negocios que se convierten en servicios interoperables.
- Clientes y otros servicios pueden acceder a servicios locales que se ejecutan en el mismo nivel.
- Clientes y otros servicios acceden a servicios remotos sobre una red.
- Estos servicios pueden usar un rango de protocolos y formatos de datos para comunicar información.
- Ofrecen la posibilidad de desarrollar código en un lenguaje determinado.
- Otorgan todas las capacidades necesarias para construir servicios.
- Dotan de la capacidad necesaria para diseñar procesos de negocio.

Una de las características más resaltantes de los SOA es que está basada en contratos, donde el proveedor establece las reglas de comunicación, el transporte, y los datos de entrada y salida que serán intercambiados por ambas partes.

2.1.1.6. CORBA (Common Object Request Broker Architecture)

Representa un modelo de soporte de programación distribuida al igual que el modelo RPC, RMI entre otros [37]. El grado de soporte de la distribución es muy distinto en estos modelos siendo CORBA el soporte más moderno de programación distribuida junto con RMI de JAVA. Desarrollado por el OMG (Object Management Group), es un software distribuido, orientado a objetos, reutilizable, portable y capaz de interactuar con entornos heterogéneos. Durante los últimos años, se ha vuelto más y más popular como un middleware de referencia para la computación distribuida [53], [54]. Se presenta como un Framework de estándares y conceptos

para sistemas abiertos [55]. Se puede considerar como una infraestructura abierta de computación distribuida que tiene el objetivo de automatizar tareas comunes de programación de red, tales como el registro, ubicación y activación de objetos; peticiones de demultiplexación; gestión y control de errores; empaquetamiento y desempaquetamiento de parámetros; y operaciones de despachado [56]. Un objeto CORBA se define mediante el IDL y se identifica de manera única mediante su IOR (Interoperable Object Reference). En esta arquitectura, los métodos de objetos remotos pueden ser invocados de forma transparente en entornos distribuidos y heterogéneos a través de su ORB, el cual se encarga de todos los mecanismos necesarios para encontrar la implementación del objeto en las peticiones, preparar la implementación del objeto para recibir peticiones y comunicar los datos que son parte de las peticiones [57], [58]. En CORBA el ORB es el canal de comunicación a través del cual los objetos se solicitan servicios con independencia de su ubicación física y representa la parte medular de un sistema CORBA. Las invocaciones del cliente son recibidas en el lado del servidor por un adaptador de objetos POA (Portable Object Adapter), que se encarga de llamar a la implementación del objeto adecuado. Figura 5.

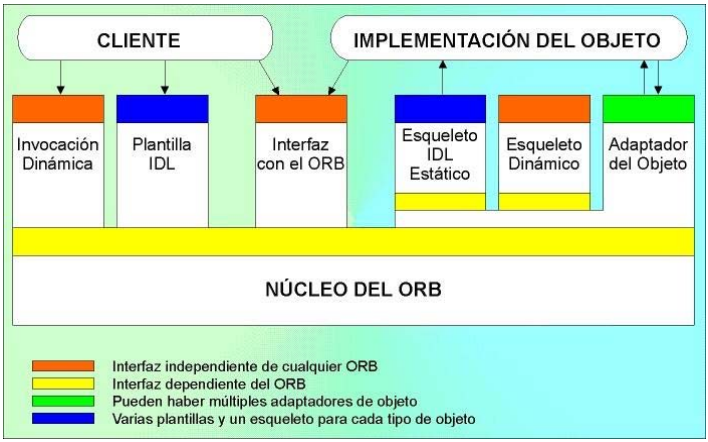


Figura 5. Arquitectura CORBA

Una descripción de las llamadas de procedimiento que esté escrita en este lenguaje permite desarrollar los denominados stub y skeleton. Un stub es un procedimiento del lado del cliente encargado de empaquetar los parámetros de llamadas de procedimiento y desempaquetar los parámetros de sus respuestas. Un skeleton es un procedimiento del lado del servidor que desempaqueta los parámetros de llamadas de procedimiento y empaqueta los parámetros de sus respuestas. Los stubs y skeletons no son obligatorios, también es posible realizar invocaciones dinámicas del lado del cliente DII (Dynamic Invocation Interface) y

del lado del servidor DSI (Dynamic Skeleton Interface). Esto significa que, dinámicamente, es posible crear, descubrir y utilizar interfaces.

Desde el punto de vista del cliente, da igual que el objeto sea local o remoto, ya que la presencia del stub hace que la invocación sea transparente a ese hecho. No obstante, el cliente necesita obtener una referencia al objeto del servidor para realizar la invocación. El proceso del lado del servidor es similar, pero utilizando el skeleton generado por el compilador IDL. Las especificaciones CORBA definen el GIOP (General Inter-ORB Protocol) como el entorno de interoperabilidad básica. GIOP asume que el protocolo de transporte subyacente es orientado a conexión, full-duplex, simétrica y que proporciona la abstracción del protocolo de comunicación entre los ORB's. GIOP sobre TCP/IP es IIOP (Internet-IOP) y para que un ORB sea compatible con CORBA, IIOP debe ser soportado. Las referencias a objetos son los medios de identificación de un objeto y se crean mediante un proceso servidor POA. Las referencias a objetos son transparentes del lado del cliente y las respuestas son encapsuladas totalmente para responder las peticiones. Figura 6.

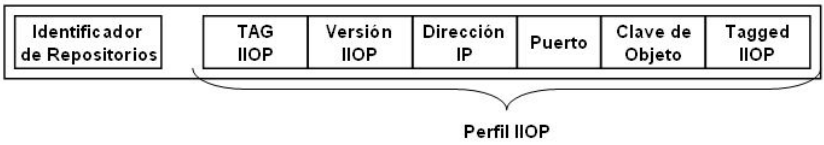


Figura 6. IOR con perfil IIOP

La referencia de objetos consiste en dos campos, el identificador de repositorios y el perfil. El identificador de repositorios identifica la mayoría de los tipos de objetos. El perfil por otro lado, posee toda la información de direccionamiento necesaria para que el mensaje de petición llegue a su destino. La referencia al objeto podría contener más de un perfil si se requiere una duplicación de objetos [53].

CORBA utiliza el lenguaje IDL para definir una serie de interfaces en la comunicación de dos o más aplicaciones. CORBA utiliza este lenguaje para ofrecer la sintaxis necesaria definiendo los procedimientos o métodos a invocar remotamente. Describe una interfaz en un lenguaje neutral, lo cual permite la comunicación entre componentes de software que no comparten el mismo lenguaje.

2.1.1.6.1.Servicios CORBA

Lo que hace de CORBA un estándar poderoso son las colecciones de servicios llamados CORBA Services que proporcionan una funcionalidad útil para el

desarrollo de una amplia variedad de aplicaciones distribuidas. Los Servicios CORBA tienen unas APIs que son definidas en el idl. Muchos de los Servicios CORBA son proporcionados como aplicaciones de servidor precompilados en lugar de librerías que están vinculadas a su propia aplicación. Los servicios CORBA son realmente unas librerías de clases estandarizadas y distribuidas [59]. Algunos de los servicios CORBA más significativos son:

A) Servicio de Nombres "NameService"

Quizás el más importante de los servicios CORBA, permitiendo a los objetos registrarse para un futuro uso. El NameService es la mejor manera de hacer que las aplicaciones CORBA sean portables y completamente distribuidas. Ofrece una correspondencia entre nombres y referencias a objetos: dado un nombre, el servicio devuelve una referencia al objeto almacenado con ese nombre. Este servicio traduce nombres a referencias a objetos. La misma referencia a un objeto se puede almacenar en diferentes momentos con diferentes nombres, pero cada nombre identifica exactamente a una sola referencia. Un contexto de nombrado es un objeto que almacena las asociaciones de los nombres. Cada objeto de contexto implementa una tabla que traduce nombres a referencias a objetos. Un nombre en la tabla puede denotar una referencia a un objeto de la aplicación o a otro objeto de contexto en el Servicio de Nombres. Esto significa que, al igual que un sistema de archivos, los contextos se pueden combinar para formar jerarquías: los contextos se corresponden con los directorios que almacenan nombres de otros directorios (otros contextos) o archivos (objetos de aplicación) [60].

B) Servicio de Trading "TradingService"

Los clientes necesitan un mecanismo dinámico para localizar los objetos. Un cliente puede tener una idea del tipo de objeto que necesita, pero puede no tener toda la información necesaria para hacer una elección precisa. El Servicio de Trading del OMG proporciona la funcionalidad que permite a los clientes localizar objetos con la ayuda de un agente (trader). Al igual que en el Servicio de Nombres, un trader almacena referencias a objetos. Sin embargo, en lugar de almacenar un nombre para cada referencia, el trader almacena una descripción del servicio provisto por cada referencia. Los clientes llevan a cabo una búsqueda dinámica de servicios a través de peticiones sobre las descripciones de dichos servicios. Este mecanismo, conocido como enlace dinámico (dynamic binding), permite una asociación dinámica de los criterios de selección a las referencias de los objetos. A menudo se ven los traders como las guías de páginas amarillas. En lugar de listar los servicios por nombre, las páginas amarillas categorizan las entradas por tema y describen cada entrada con más detalle, como nombre, dirección, rango de productos, precios, etc. Un trader almacena una oferta de servicio (service offer). Esta contiene una descripción del servicio, así como una referencia a objeto para un objeto que proporciona ese servicio. El acto de

colocar una oferta a un servicio se conoce como una operación de exportación (export) y la persona o programa que pone la oferta se denomina exportador (exporter). La referencia de objeto dentro de la oferta de servicio denomina un objeto que proporciona el servicio anunciado. Este objeto se conoce como el proveedor del servicio (service provider). Después de que se ha exportado una oferta de servicio, el proveedor de servicio es inmutable. No se puede cambiar la referencia de objeto dentro de una oferta de servicio sin borrar la oferta y volver a exportarla. Las ofertas a servicios pueden ser retiradas (withdraw), es decir, borradas de un trader. El acto de buscar el trader para un proveedor de servicio que satisfaga ciertos criterios se denomina importación (import). En esencia trader es una base de datos que almacena referencias a objetos que son descritos por sus propiedades [60].

C) Repositorio de Interfaces “InterfaceRepository”

Ofrece objetos persistentes que representan la información IDL de los interfaces disponibles en CORBA de una forma accesible en tiempo de ejecución (run-time). Esta información puede ser utilizada por el ORB para realizar peticiones. El repositorio de interfaces es una base de datos que contiene la información sobre las interfaces IDL implementadas por objetos en una red. Un programa cliente puede consultar esta base de datos en tiempo de ejecución para obtener información sobre esas interfaces. El cliente puede entonces hacer llamadas sobre la función miembro de esos objetos utilizando un componente del ORB denominado Interfaz de Invocación Dinámica (DII). Esta interfaz permite a los clientes invocar operaciones en tiempo de ejecución ofreciendo una construcción dinámica de objetos cuando éste es invocado, en lugar de llamar a una rutina stub. El estándar OMG especifica otra interface llamada Interface Skeleton Dinámica (DSI), que también permite el manejo dinámico de invocación de objetos, solo que en lugar de acceder a través de un skeleton específico de una operación, la implementación de los objetos se realiza a través de una interfaz que provee acceso al nombre de la operación y los parámetros de forma análoga al DII del lado del cliente. La DSI es una forma de manejar peticiones desde el ORB a la implementación de objeto. Tanto DSI como DII tienen varias aplicaciones mas allá de la solución de interoperabilidad. Sus usos incluyen herramientas de desarrollo de software interactivo basado en intérpretes, depuradores y monitores que desean trabajar con objetos dinámicamente y ofrecer soportes a lenguajes de tipo dinámico [61].

D) Servicio de Eventos “Event Service”

Proporciona soporte para comunicaciones desacopladas entre objetos. Permite a los proveedores enviar mensajes a uno o más consumidores con una única llamada. Los proveedores que utilizan la implementación del Servicio de Eventos no necesitan ser concientes de ninguno de los consumidores de sus mensajes; el

Servicio de Eventos actúa como un mediador que desacopla los proveedores de los consumidores. Una implementación del Servicio de Eventos también blinda a los proveedores frente a las excepciones que resulten debido a que cualquiera de los objetos consumidores pueda ser inalcanzable o a que su rendimiento sea muy pobre [60]. El Servicio de Eventos proporciona dos modelos para entregar eventos: el modelo de inyección (`push`) y el modelo de extracción (`pull`). Con el modelo inyección, los proveedores inyectan eventos al canal de eventos y el canal de eventos inyecta eventos a los consumidores. La Figura 7 muestra el estilo de entrega de eventos por inyección. Las flechas indican los papeles tanto del cliente como del servidor e indican cómo se apunta del cliente al servidor.

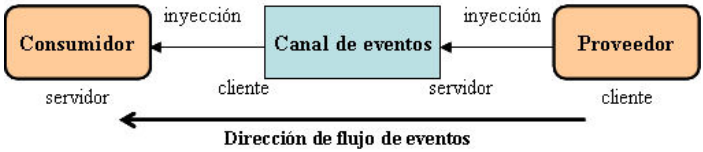


Figura 7. Modelo de entrega de eventos con estilo inyección

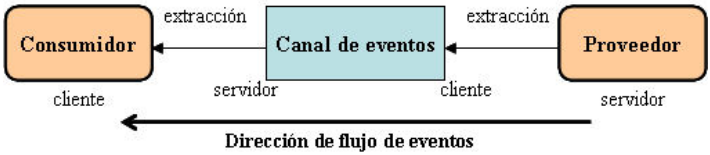


Figura 8. Modelo de entrega de eventos con estilo extracción

Para el modelo de extracción, las acciones que causan la existencia del flujo de eventos ocurren en la dirección opuesta: los consumidores extraen los eventos del canal de eventos y el canal de eventos extrae los eventos de los proveedores [60]. El modelo de extracción se muestra en la Figura 8. El canal de eventos permite conectar múltiples proveedores y consumidores. Los proveedores son los iniciadores activos de eventos mientras que los consumidores esperan pasivamente a recibirlos. Obsérvese que el mismo canal de eventos es en sí mismo un proveedor y un consumidor, Figura 9.

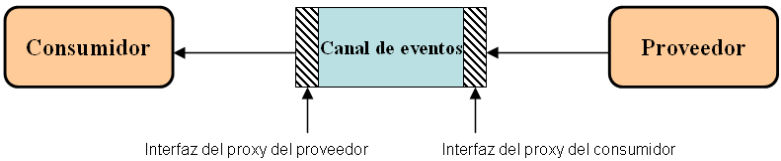


Figura 9. Interface proxy del consumidor y proveedor del canal de eventos

Estas interfaces proporcionan la ilusión a los consumidores y proveedores de que están interactuando con los proveedores reales y los consumidores reales respectivamente, por ello se denominan interfaces Proxy [60]. En el modelo de inyección, los proveedores inyectan datos de eventos a los consumidores invocando la operación push en la interface PushConsumer. Para ponerse de acuerdo tanto el consumidor como el proveedor intercambian referencias de objetos PushConsumer y PushSupplier.

2.1.1.6.2. Servicios Dinámicos CORBA en Sistemas Embebidos

La computación distribuida requiere un eficiente y poderoso mecanismo de comunicación entre aplicaciones ejecutándose en diferentes hosts y redes. Con el creciente impacto de CORBA en sistemas DRE [62] se ha aumentado la necesidad de implementar servicios que puedan adaptarse a entornos variables. Los servicios CORBA basados en stubs y skeletons estáticos son utilizados con éxito en las áreas de telecomunicaciones, aeroespacial, automoción, sistemas embebidos y microprocesadores con pocos recursos. Sin embargo los stubs y skeleton estáticos presentan algunos inconvenientes. Un servicio podría necesitar interactuar con objetos cuyas interfaces ya no existen o que fueron reemplazadas por otras. Con los servicios CORBA estáticos, aplicar este tipo de cambios requeriría tediosas modificaciones y el rediseño o reimplementación de la aplicación existente. Actualmente, las implementaciones CORBA en sistemas DRE están centradas al desarrollo de aplicaciones y servicios estáticos ya que éstas están más enfocadas a dispositivos cuya arquitectura son contadas en kilobytes. Las implementaciones CORBA/e de perfiles compacto y micro eliminan la necesidad de trabajar con sistemas dinámicos para adaptarse a entornos de capacidad reducida. En sistemas embebidos pequeños como sensores o microprocesadores de pocos recursos esta característica de CORBA es una necesidad imperiosa. Pero, gracias al incremento de la capacidad de procesamiento, periféricos y el crecimiento exponencial de la capacidad de memoria, hay una clara tendencia a aumentar las prestaciones de los mismos, lo que conlleva la posibilidad de ejecutar complejas aplicaciones como las de los actuales ordenadores. Para enfrentar las limitaciones de las aplicaciones CORBA estáticas implementadas en sistemas DRE, se hace uso del repositorio de interfaces (IFR) dinámicas CORBA [62] el cual es un servicio que provee cierto tipo de información acerca de las interfaces CORBA y otras entidades definidas en el IDL. Las invocaciones estáticas se realizan a través de stubs previamente generados y cuyas interfaces corresponden a implementaciones y APIs de ORBs propietarios. Las interfaces dinámicas DII/DSI proporcionan el único acceso directo como interfaz para hacer peticiones a la capa de comunicación. Gracias a los avances en HS (Hardware/Software) hoy día, las implementaciones dinámicas de CORBA en sistemas DRE no son un factor limitante. Una de las desventajas más importantes a

tener en cuenta es el rendimiento. Se ha demostrado experimentalmente [63] que las interfaces DII/DSI dinámicas en implementaciones corrientes son mucho más lentas que aquellas desarrolladas estáticamente.

2.1.1.6.3.CORBA en Sistemas Embebidos

Los sistemas embebidos interactúan de muchas maneras de forma cooperativa: en automóviles, tablets, móviles, PDA's, sistemas de vigilancias, cámaras, y otros miles de dispositivos que puedan contener uno o varios procesadores integrados conectados a una red. En una zona urbana, por ejemplo, podría haber cientos de dispositivos, cámaras inteligentes de visión artificial, semáforos, paneles de tráfico, y diferentes módulos de control de proceso los cuales pueden interoperar a su vez con muchas unidades como sensores pequeños o varios servidores de gran tamaño de manera distribuida e integrada entre sí a través de los sistemas cooperativos. La combinación, interoperabilidad y fiabilidad absoluta basada en estándares middleware son necesarios para estas aplicaciones. Para los desarrolladores de sistemas embebidos, CORBA/e ofrece una interoperabilidad de alto rendimiento permitiendo a los sistemas DRE y sistemas basados en microprocesadores de recursos limitados interactuar, no sólo con los demás, sino también con todos los sistemas existentes utilizando el estándar CORBA.

CORBA/e está orientado al desarrollo de entornos middleware abiertos, maduros, robustos y de memoria eficiente (small footprint), ofreciendo alto rendimiento y comportamiento determinístico en el momento de trabajar en entornos de tiempo real [62]. CORBA está diseñado para permitir a los clientes invocar operaciones sobre objetos remotos sin preocuparse de donde reside el objeto [64], lo que lo hace ideal para proporcionar la infraestructura de comunicación básica en sistemas DRE. Las mejoras de los sistemas embebidos basados en procesadores a nivel de procesamiento y soporte de sistemas operativos embebidos hace a los fabricantes pensar en la incorporación de un software middleware de conectividad que ofrezca conjuntos de servicios que interactúen con aplicaciones distribuidas abstrayendo la heterogeneidad y complejidad [65], [66]. Un desafío clave para la investigación en sistemas DRE es determinar la forma de mantener un tamaño reducido del middleware ORB, los stub y skeletons generados por el compilador IDL [67], limitando a los servicios y aplicaciones de todas las características dinámicas del standard CORBA. CORBA/e se enfoca principalmente en desarrollo de sistemas interoperables que serán implementados en pequeños dispositivos de limitada capacidad. CORBA/e mantiene la total interoperabilidad con el IIOP preservando muchas de las opciones del POA. CORBA/e afronta a los sistemas DRE con dos tipos de perfiles, el perfil Compacto y

el perfil Micro, cuyas habilidades son la incorporación adicional de perfiles específicos en sistemas embebidos.

El perfil compacto de CORBA/e, también conocido como minimumCORBA, está enfocado al desarrollo de servicios que puedan ser ejecutados en procesadores embebidos de pocos recursos y en entornos de tiempo real. Los servicios implementados bajo este perfil son asumidos por ser desarrollados con interfaces estáticas. La especificación minimumCORBA [68] de la OMG define que para algunas aplicaciones CORBA resulta excesivamente grande cumplir con las exigencias y requisitos de tamaño y rendimiento. Por esta razón, el perfil compacto define un subconjunto de utilidades CORBA opcionales para ser implementados en sistemas embebidos. El perfil compacto de CORBA soporta toda la especificación IDL de la OMG permitiendo la compatibilidad entre el perfil compacto y aplicaciones desarrolladas bajo el standard full de CORBA. Para preservar los beneficios claves de CORBA el perfil compacto desarrolla la portabilidad entre aplicaciones y la interoperabilidad entre los ORBs. En general el perfil CORBA/e compacto soporta la programación estática en tiempo real. Se presenta como un subconjunto de la especificación CORBA IIOP diseñado para sistemas con limitaciones computacionales o de escasos recursos de memoria. Dada las limitaciones y aspectos omitidos en este perfil, la única forma de ejecutar objetos sirvientes (entidad que implementa uno o varios objetos CORBA) es a través de ORB::run() [69].

El perfil CORBA/e micro es un subconjunto del perfil compacto con características aún más reducidas. Son lo suficientemente pequeños como para caber en microprocesadores o DSPs (Digital Signal Processors) de baja potencia. El perfil CORBA/e micro define ejecutables CORBA que se deberán encajar en sólo decenas de kilobytes de memoria. Pero, incluso en estos pequeños tamaños disponibles, el perfil micro contempla la total interoperabilidad IIOP soportando funcionalidades básicas en sistemas y redes de trabajos pequeños. El perfil micro proporciona integración con aplicaciones que son ejecutadas en CORBA, CORBA/e compacto y CORBA/e micro. Soporta IIOP nativos, todas las versiones a través de GIOP 1.4 e IIOP 1.4. Este perfil, como el compacto, sólo soporta definiciones de interfaces, programación e interacciones estáticas. Del lado del servidor, para mantener su comportamiento reducido y determinístico, soporta exactamente un sólo POA permitiendo sirvientes transitorios, identificadores de sistemas asignados y el manejo de multi-hilos bajo el control y supervisión del ORB.

A pesar de la existencia de perfiles reducidos para sistemas embebidos con recursos limitados, la realidad es que en muchos sistemas de transporte se utilizan sistemas embebidos con grandes capacidades de procesamiento y almacenamiento. La mejora de los prestaciones de los sistemas procesadores y la mayores

capacidades de integración logran conseguir dispositivos capaces no sólo de implementar sistemas operativos embebidos sino de incorporar capas midlleware totalmente funcionales.

2.1.1.6.4.Implementaciones CORBA

Hoy en día existen varias implementaciones CORBA, tanto en el dominio comercial como en sistemas de código abierto. Una comparación detallada de varias implementaciones de CORBA se puede encontrar en [70]. Al elegir una aplicación CORBA, se deben tener en cuenta varias consideraciones:

- *Plataformas soportadas.* Cuando se trabaja con sistemas embebidos heterogéneos es importante elegir una aplicación CORBA capaz de funcionar correctamente en la mayoría de ellos. Esto es particularmente cierto en entornos urbanos, donde los equipos de diferentes fabricantes y con diferentes plataformas han sido instalados en un sistema altamente distribuido. En particular, el soporte a las arquitecturas SoC (System-on-Chip) representa un gran avance en este campo. Los SoC han sido muy populares para procesadores multimedia, utilizados comúnmente en aplicaciones de vigilancia y monitorización.

- *Lenguajes de programación soportados y requisitos previos.* Los lenguajes de programación soportados y los requisitos previos para compilar con éxito la implementación de CORBA deben ser también tenidos en cuenta. Esta particularidad es cierta para equipos que suelen basarse en una gran variedad de procesadores y sistemas operativos. En este caso, la heterogeneidad de los equipos instalados requiere que el software pueda ser adaptado al Sistema Operativo y a la plataforma de hardware existente.

- *Características Soportadas.* En función de la aplicación final, características adicionales como objetos pasados por valor, mensajería asíncrona, modelo de componentes CORBA, etc.

- *Licencias.* Hoy en día se pueden encontrar implementaciones CORBA comerciales y de código abierto. Las licencias bajo las cuales son liberadas estas implementaciones pueden afectar el desarrollo del proyecto. En el caso de entornos urbanos, las licencias de código abierto son las preferidas por las autoridades públicas para lograr la independencia del proveedor.

MICO (www.mico.org) y TAO (http://www.theaceorb.com) son implementaciones CORBA totalmente compatibles entre sí. Ambas satisfacen ampliamente los criterios detallados anteriormente. Están completamente escritos

en C++, y son compatibles con las plataformas más utilizadas, incluyendo arquitecturas SoC basadas en procesadores ARM y MIPS (Microprocessor without Interlocked Pipeline Stages). Están disponibles bajo los modelos de "software de código abierto". Esto significa que sus recursos son de descarga gratuita, abiertos para inspección, revisión y comentarios. Estos incluyen la mayoría de las características novedosas de CORBA [70].

2.1.2. Comparativa de Tecnologías Middleware

Una colección de agentes colaborativos como los MAS (Multi Agent Systems) interactúan unos con otros para alcanzar objetivos comunes e individuales [71]. Los MAS se están convirtiendo en un enfoque muy prometedor para resolver problemas complejos a través de mecanismos de interacción entre agentes [72]. En situaciones heterogéneas, los sistemas distribuidos son bien conocidos por sus dificultades para interoperar entre agentes, lo que hace que crezca el interés sobre plataformas software unificado y protocolos de implementaciones estándar [73].

RPC se caracteriza por su facilidad de uso y robustez [37]. Su facilidad de uso es el resultado del mayor nivel de abstracción que los RPCs proporcionan a los desarrolladores con el procedimiento normal de las llamadas. RPC aísla el nivel inferior de detalles tales como el empaquetamiento y desempaquetamiento de parámetros pero es inflexible a cambios. RPC supone una relación entre el cliente y el servidor en tiempo de ejecución. Esto hace al cliente y al código del servidor estar acoplados fuertemente uno al otro. RPC además se basa en la programación estructurada que ya está obsoleta frente al modelo orientado a objetos. Otro de los problemas del RPC es la incapacidad para manejar la comunicación entre un cliente y múltiples servidores, salvo cuando una conexión es utilizada para cada comunicación. Esto se debe a que el RPC asume que todas las comunicaciones son uno-a-uno, es decir, el cliente habla con un solo servidor a la vez.

En RMI el flujo de una petición no es muy diferente a la del RPC. No obstante, RMI tiene algunas diferencias respecto a los RPC. IDL en RPC se basa usualmente en procedimientos C, en tanto que IDL en RMI está basado en JAVA orientado a objetos. RMI proporciona una mayor transparencia comparado con RPC. Por otra parte, RMI se basa en su plataforma independiente. Esto implica que la portabilidad del código JAVA ayuda a hacer frente a los problemas heterogéneos en sistemas distribuidos. Sin embargo, su naturaleza basada en JAVA juega en contra de RMI.

Los TP Monitors proporcionan colas, enrutamientos, mensajerías y funciones, lo cual permitirá a los desarrolladores de aplicaciones distribuidas pasar por alto las características transaccionales del TP Monitor. Su mayor beneficio es el

rendimiento en función al balanceo de carga, lo que les permite responder a varias transacciones [39]. La principal desventaja de los TP Monitors es que presentan dificultades a la hora de intercambio de información ya que todo lo maneja como un paradigma de transacciones forzando al desarrollador trabajar con ellos. No importa que tipo de operación o proceso se lleve a cabo, todo debe ser incluido en una sola transacción [40].

Los WS son tecnologías comúnmente utilizadas en arquitecturas SOA, haciendo uso de XML para crear conexiones entre aplicaciones a través de la web. En los WS se utiliza SOAP como protocolo de comunicación entre los distintos servicios. SOAP es un protocolo basado en XML. Pero procesar grandes mensajes SOAP reduce significativamente el rendimiento del sistema, provocando cuellos de botellas en comparación con tecnologías como CORBA [74]. Esto representa un problema central, específicamente en redes de comunicación wireless [75] así como en el creciente número de dispositivos que forman parte de los ITS.

SOA se presenta como una alternativa atractiva para permitir la interoperabilidad de sistemas y la reutilización de los recursos. No es una nueva arquitectura, las primeras arquitecturas orientadas a servicios eran DCOM o los ORB basados en CORBA. Puede servir como soporte subyacente para aplicaciones MAS en sistemas distribuidos y heterogéneos [76]. Sin embargo, las aplicaciones SOA enfrentan muchos problemas en materia de seguridad durante el diseño y el desarrollo [77] como la falta de herramientas de soporte. La seguridad es agregada en el momento de la implementación, lo que conlleva problemas con las normas SOA y la incapacidad de controlar la seguridad de los Servicios Web para adoptar nuevas medidas. En la práctica, las aplicaciones basadas en SOA no siempre resultan exitosas ya que la mayoría se realiza de forma ad-hoc basados principalmente en experiencias personales [78]. Extraer la información, representarlo, interpretarlo y mantenerlo también forma parte de los inconvenientes de los SOA actuales. A pesar de que las empresas van incrementando sus dependencias hacia SOA, estos sistemas se encuentran aun en una etapa incipiente e inmadura con grandes problemas de seguridad [79].

Las funciones básicas de los ITS son la recolección, manejo y disposición de la información en tiempo real del transporte. CORBA es capaz de gestionar todo el flujo de comunicación de manera transparente en estos sistemas altamente distribuidos. Una de las ventajas más importantes de la incorporación de CORBA es su capacidad para soportar entornos heterogéneos, productos de diferentes proveedores, y varios de los lenguajes de programación más populares. Esto es precisamente lo que ocurre en entornos urbanos, donde abundan equipamientos y tecnologías de diferentes proveedores. Por lo tanto resulta más lógico utilizar CORBA para sistemas embebidos implementados en un entorno ITS maximizando la movilidad, seguridad

y la utilidad a través de la combinación de sistemas de transporte existentes con las tecnologías de la información, comunicación y control.

2.2. Ontologías en los ITS

La localización de servicios, la gestión de la información de forma inteligente y la interoperabilidad son varios de los problemas más conocidos en la actualidad en la implementación de entornos cooperativos [80]. La combinación de una capa middleware como CORBA junto una ontología en el ámbito de los ITS representa una posible solución a los problemas mencionados. Una ontología puede definirse como el resultado de seleccionar un dominio y aplicar sobre éste un método para obtener una representación formal de conceptos contenidos en él [81]. La evaluación de contenidos ontológicos tuvo su inicio en 1994 [82]. En los últimos años, la incorporación de ontologías ha ido aumentando, especialmente en sitios Web semánticos donde posibilitan la interoperabilidad de datos heterogéneos [83]. Las ontologías a menudo son capturadas en forma de red semántica (gráfica cuyos nodos representan los conceptos u objetos individuales y las relaciones o asociaciones entre ellas) [84]. Las clases o el concepto de un dominio son representados formalmente como objeto (físico o lógico) muy probablemente parecidos a los sustantivos, y las relaciones entre ellos son representadas como verbos en oraciones que describen al dominio. Las ontologías nos brindan un esquema conceptual a explotar a través de esquemas de intercambio de metadatos. En el caso de las tecnologías middleware, el servicio Trading de CORBA, por ejemplo, actúa como una base de datos que almacena referencias a objetos junto con descripciones del servicio que oferta. No obstante, la búsqueda de un servicio en particular debe hacerse de manera más inteligente, con algún mecanismo de diferenciación y ponderación de la información. El servicio de Trading es incapaz de gestionar información semántica. En este sentido, el uso de ontologías en el campo de los ITS es una solución para permitir la reutilización del conocimiento del dominio y generar servicios inteligentes. Los agentes que comparten información sobre bases de datos semánticas podrían utilizar esta información ontológica para responder solicitudes DD (Device-Device), servir como datos de entrada a otros servicios, permitir la reutilización del conocimiento de dominio o incluso trabajar de forma cooperativa con otras ontologías existentes. Una de las principales ventajas de la incorporación de ontologías en los ITS es la localización semántica de servicios de determinadas características y propiedades de forma dinámica e inteligente. Los dispositivos pueden hacer uso de datos y metadatos de distinta naturaleza en tiempo de ejecución sobre eventos generados en el tráfico o sobre algún interés en particular. Los servicios podrán almacenar sus referencias a objeto en ontologías descriptivas. Cuanto más estructurada se encuentre la información de los servicios, mas sencilla, exacta, rápida e inteligente será

encontrarla. Los metadatos podrían dotar de cierta semántica a esta problemática ya que las ontologías ofrecerían un esquema conceptual de intercambio.

Existen numerosos trabajos previos que han hecho uso de metadatos para mejorar la implementación de aplicaciones colaborativas en distintos escenarios. Lee y otros [85] presentan un modelo de contexto basado en una ontología que adopta un enfoque combinado para modelar la información de contexto, utilizados por los servicios de transporte. La información distribuida que se modela está relacionada con un contexto primario sobre la localización, tiempo, identidad y calidad de servicios, pero enfocados únicamente a un servicio de localización de espacios en aparcamientos. Gracias al escenario propuesto demostraron que la información de contexto originario de fuentes autónomas distribuidas se puede representar mediante un modelo de datos comunes, y que puede ser estructurada siguiendo una ontología común, dando como resultado que los datos pueden ser compartidos, asociados, fusionados, o razonados. Kuong-Ho Chen y otros [86] presentan el diseño e implementación de un Framework para transportes públicos. Incluyen un mecanismo de colección de datos a través de WS, así como la incorporación y composición de esquemas para planear rutas. En su estudio incluyen un mecanismo de recopilación de datos, la agregación y composición de esquemas que se utilizan específicamente para la planificación de rutas de tránsito a través de los datos recogidos. Sin embargo, hay tener en cuenta que el núcleo de los WS es el SOAP y el protocolo para integrar estos servicios está basado en XML, lo que incrementaría la demanda de servicios y en consecuencia el consumo de ancho de banda para sistemas más complejos. En [87], Fernández y otros sostienen que el uso de los MAS permite un diseño desacoplado y la implementación de diferentes módulos (agentes), fomentando así la reutilización de dominios ontológicos similares (Ej. tráfico) reduciendo el esfuerzo de desarrollo y aumentando la fiabilidad del sistema (reutilización de servicios existentes). Presentan el estudio sobre una arquitectura multi-agente orientada a servicios para la construcción de avanzados DSSs (Decision Support Systems) para la gestión del transporte. Sin embargo no especifican cómo utilizar la información como herramienta ni como trabajar de forma cooperativa con otras ontologías existentes. En [27], Terziyan y otros presentaron un enfoque sobre los requerimientos y la arquitectura necesaria sobre sistemas de gestión de tráfico, mostrando cómo dicho sistema puede ser beneficioso desde el punto de vista semántico a través de agentes tecnológicos pero cuestionándose a la vez la forma de combinarlos con el procesado de datos y herramientas automáticas. Jun Zhai y otros [88] han presentado un sistema de recuperación de la información basándose en un Framework ontológico difuso. El Framework presentado consta de tres partes: conceptos, propiedades de los conceptos y valores de las propiedades, donde el valor de la propiedad puede ser cualquier tipo de dato estándar o valores lingüísticos de conceptos difusos. El inconveniente en el trabajo presentado por Jun Zhai es que el sistema de

recuperación de la información se enfoca principalmente en información de accidentes en el tráfico, dejando de lado otros puntos clave como la interoperabilidad entre dispositivos o la heterogeneidad de la información.

Un sistema de tráfico cooperativo debe ser capaz de resolver problemas complejos utilizando datos y metadatos del entorno. Deben ser capaces de interactuar unos con otros, formando sistemas multi-agentes para alcanzar objetivos. Los ITS deben estar preparados para aprender del entorno y modificar sus características en base a eventos. Se propone una solución inteligente para la recuperación y gestión de la información heterogénea con la finalidad de construir una ontología utilizando una taxonomía como punto de partida del estudio. La ontología servirá para organizar y ofrecer una base de metadatos de servicios propagados por toda la red de tráfico.

2.2.1. Construcción de Ontologías

Uno de los primeros pasos para la construcción de una ontología es sin duda la IR (Information Recovery). La IR representa un reto importante para los servicios, aplicaciones y dispositivos implementados en los ITS. Construir ontologías desde cero y de forma manual requeriría una gran cantidad de tiempo y esfuerzo, y el resultado quedaría posiblemente incompleto. Construir una ontología de servicios ITS incorporando técnicas de rigor científico en el análisis y selección dinámica de la información permitirá dotar a la gran cantidad de información existente de una estructura lógica basada en soluciones de gestión del conocimiento tanto en el campo de los ITS como en otras áreas de dominio.

Según sostiene Kitchenham [89], la SLR (Systematic Literature Review) se ha convertido en una metodología de investigación importante para la recuperación y recopilación de la información. Las SLR científicas son estudios pormenorizados, selectivos y críticos que tratan de analizar e integrar la información esencial de los estudios primarios de investigación sobre un tema específico, en una perspectiva de síntesis unitaria de conjunto. Las revisiones sistemáticas de la información tienen como objetivo principal la identificación, evaluación e interpretación de todos los estudios de investigación relevantes a una cuestión de investigación en particular, utilizando métodos rigurosos y algoritmos específicos sobre el fenómeno de interés. He Zhang y otros [90] exponen que la rigurosidad en los procesos de búsqueda de información representa un punto crítico que distingue a las revisiones sistemáticas de las revisiones tradicionales de la literatura hechas de forma ad-hoc. Estos autores han desarrollado un enfoque sistemático basado en la evidencia para el desarrollo y ejecución de estrategias óptimas de búsqueda sobre la literatura digital. El enfoque propuesto incorpora el concepto QGS (Quasi-Gold Standard), que consiste en la

recopilación de los estudios conocidos, y la correspondiente “quasi-sensitivity” encargado de evaluar el rendimiento del proceso de búsqueda. Sin embargo, aunque los estudios de casos presentados proporcionan pruebas primarias para apoyar la viabilidad del enfoque de búsqueda sistemática, no proporcionan un proceso sistemático y riguroso para la integración y búsqueda de pares/triples de palabras en su estudio.

Existen otros trabajos sobre las metodologías para el desarrollo de ontologías. Gruninger y otros [91] proponen una metodología para diseñar y evaluar una ontología que consiste en primer lugar en identificar intuitivamente las aplicaciones posibles en las que se utilizará la ontología. Utilizan un conjunto de preguntas llamadas cuestiones de competencia para determinar el ámbito de la ontología y para extraer los conceptos principales, sus propiedades, relaciones y axiomas. Fernández y otros [92] presentaron un enfoque más sistemático para la construcción de una ontología a partir de cero llamado Methontology. Esta es quizás una de las propuestas mas completas, ya que toman el desarrollo de ontologías como un proyecto informático. Abarca actividades para la planificación del proyecto, la calidad del resultado, la documentación, etc., permitiendo construir ontologías nuevas o reutilizar otras existentes. Methontology consta de tres procesos diferentes: la gestión, el desarrollo y el proceso de apoyo. Según Methontology, esos procesos se llevan a cabo simultáneamente y cada uno de ellos se lleva a cabo por varios grupos de desarrolladores (por ejemplo, expertos sobre ciertos dominios, expertos en ontologías, expertos informáticos, etc.). Sin embargo, no explican cómo agrupar a esos grupos con tareas específicas, cómo combinar ni dividir esas tareas una vez terminadas. Methontology no es propiamente una herramienta de extracción ni recuperación de la información. Conceptualiza una ontología usando un conjunto de representaciones tabulares y gráficas, las cuales permiten modelar los principales componentes de una ontología. Nanda y otros [93] aplicaron un análisis de conceptos formales en su metodología para desarrollar una ontología de dominio específico. Utilizaron un análisis de conceptos formales para identificar las similitudes entre un conjunto finito de objetos basados en sus propiedades y proporcionando una agrupación jerárquica conceptual. Sin embargo, ninguna de las metodologías anteriormente mencionadas hace uso de herramientas que permitan, a partir de la IR y la SLR, construir ontologías de manera más sistemática. Es necesario estipular mecanismos más autónomos para llevar a cabo un meta-análisis de los datos y en base a estos confeccionar ontologías.

2.2.2. Elementos de una Ontología

Las ontologías tienden a proporcionar un vocabulario común de un área y definen, a diferentes niveles de formalismos, el significado de los términos y

relaciones entre estos. Según Gruber [94], el conocimiento en ontologías se formaliza principalmente utilizando cinco tipos de componentes: clases, relaciones, funciones, axiomas e instancias.

- **Las clases** o conceptos suelen ser organizados en taxonomías. Representa una descripción formal de una entidad del universo o dominio que se quiere representar. Constituye la pieza básica de estructuración del conocimiento. La decisión sobre qué clase considerar en un dominio no es fácil. Tal decisión debe ser tomada de acuerdo a los objetivos de la ontología (extraídos de las sesiones con los expertos, preguntas relevantes y de estándares existentes en el dominio). Una clase puede tener subclases que representan conceptos que son más específicos que dicha clase.

- **Las relaciones** representan un tipo de interacción entre los conceptos o clases del dominio que se modela. Algunas relaciones semánticas básicas son: `subClassOf`, `isPartOf`, `ako`, `connectedTo`, `is-a`, `areDriverBy`, `isPerformedBy`, `isManagedBy`, etc. Las relaciones más simples se modelan mediante una propiedad de una clase cuyo valor es una instancia de otra clase. Si las relaciones son más complejas (deben tener a su vez propiedades o deben organizarse en jerarquías) se suelen definir mediante clases que las representen. Las relaciones pueden definirse formalmente como cualquier subconjunto de un producto de n conjuntos, donde: `R: C1 x C2 x... x Cn`.

- **Las 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. Por ejemplo, pueden aparecer funciones como `identify-class`, `assign-date`, `check-availability`, etc. El `n-ésimo` elemento de la relación es único para los `n-1` precedentes. Formalmente, definimos las funciones `F` como: `F: C1 x C2 x... x Cn-1` → `Cn`.

- **Los axiomas** son reglas que se añaden a las ontologías y que permiten describir el comportamiento de las clases. Son expresiones que siempre son consideradas ciertas. Se establecen a partir de valores específicos de las propiedades. Por ejemplo: “Para todo `A` que cumpla la condición `C`, entonces `A` es `B`”. Permiten dejar constancia de que ciertos valores de propiedades introducidos son coherentes con las restricciones de la ontología, o bien inferir posteriormente valores de atributos que no se han introducido explícitamente. De esta forma, a través de estos es posible inferir conocimiento no codificado explícitamente en la ontología.

- **Las instancias** representan objetos concretos del dominio pertenecientes a una clase. La colección de instancias constituye la base de hechos (también

denominada base de datos o base de conocimiento) del modelo. Cuando una clase es instancia indirecta de otra quiere decir que es instancia de alguna de sus clases derivadas. En contraposición a instancia directa, donde no existen clases intermedias.

2.2.3. Principales Lenguajes Ontológicos

Existen muchos lenguajes que se utilizan para representar ontologías o el conocimiento de un dominio. Los lenguajes ontológicos son vehículos para expresar ontologías de forma comprensible por las máquinas. Algunos de ellos han emergido en los últimos años en paralelo a la idea de Web Semántica. En las siguientes secciones se presentan algunos de los principales lenguajes ontológicos.

2.2.3.1. RDF (Resource Description Framework)

Representa un modelo estándar para el intercambio de datos en la Web. RDF tiene características que facilitan la fusión de datos incluso si los esquemas subyacentes son diferentes, y se apoya específicamente en la evolución de esquemas en el tiempo sin requerir que todos los consumidores de datos sean cambiados. RDF extiende la estructura de enlace de la Web utilizando URIs para nombrar relaciones entre las cosas, así como dos extremos del enlace (normalmente conocido como "tripleta"). El uso de este modelo simple permite que los datos puedan ser estructurados y semi-estructurados para ser mezclados, expuestos y compartidos a través de diferentes aplicaciones. Esta estructura forma un enlace tipo gráfico de etiquetas, donde los arcos representan el enlace con nombre entre dos recursos, representada por los nodos del gráfico. Este punto de vista gráfico es utilizado a menudo para entender de forma fácil las explicaciones del dominio RDF [95]. El diseño de RDF está enfocado a satisfacer los siguientes objetivos:

1. Gráfico de modelado de datos simple, independiente de cualquier sintaxis de serialización específica.
2. Semántica formal e inferencia demostrable.
3. Vocabulario extensible basado en URI.
4. Soporte de sintaxis basadas en XML utilizando tipo de datos XML-Schema.
5. Permite hacer declaraciones sobre cualquier recurso.

En RDF los datos son representados como set de tripletas (declaraciones) del tipo (Sujeto, Predicado, Objeto). Este set es llamado grafo RDF. Los sujetos y objetos representan los nodos del grafo (en ocasiones nodos en blanco) y los predicados representan el enlace (arco) entre los nodos. Los sujetos y predicados

son identificados a través de URI, y éstos incluyen la URL del recurso. Los objetos pueden actuar como sujetos en otras declaraciones o como literales. Los literales son constantes escritas junto con sus tipos de datos [96].

2.2.3.2. RDFS (RDF Schema)

RDFS representa una extensión de RDF. Provee un mecanismo para describir grupos de recursos relacionados y las relaciones entre ellos. Está escrito en RDF y es utilizado para determinar características de otros recursos como dominios y rangos de propiedades. Fue desarrollado y publicado en febrero de 2004 por la W3C con el objetivo de describir contenidos semánticos, interoperables y basados en XML. RDFS no proporciona vocabularios específicos, sino facilidades para describir clases y propiedades de un dominio específico. Un RDFS permite comprobar si un conjunto de tripletas RDF resulta válido o no para ese esquema, Figura 10. Al disponer de un esquema RDF, se puede comprobar si las propiedades aplicadas a los recursos son correctas (Ej. Una Videocámara no puede tener una propiedad asociada que sea numeroMatriculaVehiculo) y si los valores vinculados a las propiedades tienen sentido. RDFS permite controlar la validez de los recursos y restringir las entidades a las cuales pueden aplicarse ciertas propiedades.

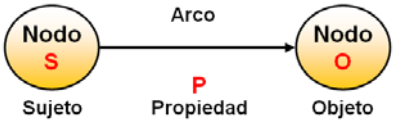


Figura 10. Tripleta RDF

Pese a la similitud entre los términos RDFS y "esquemas XML", sus significados son muy distintos. Los esquemas XML especifican el orden y las combinaciones de etiquetas que son aceptadas en un documento XML. En cambio, los RDFS especifican qué interpretación hay que dar a las sentencias de un modelo de datos RDF dejando libre la representación sintáctica del modelo. Los esquemas XML modelan datos expresados en XML mientras que los RDFS modelan conocimiento del domino. Dicho de otra manera, XML y los esquemas XML representan un lenguaje de modelado de datos mientras que RDF y RDFS son lenguajes de modelado de metadatos. Según los Miembros de la W3C [97], la especificación RDFS incluye seis clases principales:

1. *rdfs:Resource* todas las entidades descritas por expresiones RDF son recursos y son consideradas instancias de la clase rdfs:Resource. Esta representa la clase del todo. Todas las demás clases son subclases de esta clase. rdfs:Resource es una instancia de rdfs:Class.

2. *rdfs:Class* se corresponde con el concepto de clase en POO (lenguajes de programación orientados a objetos). Las clases RDF pueden representar webs, organizaciones, personas, dispositivos, etc. cada clase es miembro de rdfs:Class, clase de todas las clases. Cuando se crea una nueva clase mediante un RDFS, la propiedad rdf:type del recurso representado por la clase, adquiere como valor el recurso rdfs:Class o alguna subclase de rdfs:Class. Cuando un recurso tiene una propiedad rdf:type cuyo valor es una determinada clase, se dice que el recurso es una instancia de esa clase. Todas las clases son recursos.

3. *rdfs:Literal* corresponde al conjunto denominado literals en el modelo formal RDF. Las cadenas de texto son ejemplos de literales. Un literal es en sí una instancia de una clase de tipo de dato. rdfs:Literal en una instancia de rdfs:Class. rdfs:Literal es una subclase de rdfs:Resource.

4. *rdfs:Datatype* es la clase de tipos de datos (datatypes). Todas las instancias de rdfs:Datatype se corresponden con el modelo de tipo de datos descritos en la especificación RDF. rdfs:Datatype es en sí una instancia y una subclase de rdfs:Class. Cada instancia de rdfs:Datatype es una subclase de rdfs:Literal.

5. *rdf:XMLLiteral* es una instancia de rdfs:Datatype y una subclase de rdfs:Literal.

6. *rdf:Property* es una instancia de rdfs:Class. Representa el subconjunto de recursos RDF que son propiedades. El dominio de estos recursos se describe mediante la propiedad rdfs:domain y el rango mediante rdfs:range. Las jerarquías de las propiedades están representadas mediante rdfs:subPropertyOf. rdfs:range es una instancia de rdf:Property que se utiliza para establecer que los valores de una propiedad son instancias de una o más clases. rdfs:domain es una instancia de rdf:Property empleada para establecer que un recurso con una cierta propiedad es instancia de una o más clases. rdf:type representa también una instancia de rdf:Property y es utilizado para indicar que un recurso es una instancia de una clase (Un triple de la forma R rdf:type C donde: C es la instancia de rdfs:Class y R es una instancia de C). Otra de las propiedades relevantes de RDFS es rdfs:subClassOf, que describe una clase como subclase de otra. Sólo las instancias de rdf:Class pueden tener la propiedad rdf:subClassOf. RDFS define también varias propiedades:

- *rdfs:comment* proporciona la descripción en una lengua natural de un recurso. Ej. <rdfs:comment> "El servicio de Control de Tráfico Urbano en sí es la representación formal de las clases que intervienen en un ITS urbano." </rdfs:comment>.
- *rdfs:label* proporciona una versión legible para los humanos del nombre del recurso. Ej. <rdfs:label> Control de Tráfico </rdfs:label>.
- *rdfs:seeAlso* especifica un recurso que proporciona información adicional sobre el recurso principal. Ej. <rdfs:seeAlso> http://edsplab.us.es/ITSresearch </rdfs:seeAlso>.
- *rdfs:isDefinedBy* indica cual es el recurso donde se defina el recurso principal. Esta propiedad es una subpropiedad (rdfs:subPropertyOf) de la propiedad rdfs:seeAlso.

RDFS presenta algunas desventajas. Por ejemplo, no es posible declarar restricciones de rango rdfs:range que sean válidas sólo para algunas clases. rdfs:range define el rango de una propiedad para todas las clases, no siendo posible representar algunas características de las propiedades. Tampoco se puede declarar que una propiedad es transitiva como menorQue, simétrica tocaA o inversa raizCuadradaDe, y no es posible reflejar que unas determinadas clases son disjuntas. Por ejemplo, la clase Persona tiene asociada dos clases disjuntas (Hombre y Mujer). En RDFS puede declararse que tanto Hombre como Mujer son subclases de Persona, pero no puede especificarse que son clases disjuntas y no permite expresar restricciones de cardinalidad.

2.2.3.3. DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer)

Se presenta como un lenguaje de marcas para recursos WEB. Se basa en los estándares de la W3C como RDF y RDFS pero extiende primitivas de modelado comúnmente encontrados en lenguajes basados en frames. Es el lenguaje sucesor de DAML y OIL que combinados presentan características muy similares. Está basado en tecnologías WEB existentes como XML y URI. Introduce varios constructores del lenguaje que incrementan el poder expresivo de DAML. La primera versión de DAML+OIL fue lanzada en diciembre del 2000 introduciendo innovaciones como el manejo de conceptos en ontologías superiores, restricciones, cardinalidades locales y semánticas de restricciones [98]. La semántica y la representación del lenguaje pueden ser formalmente especificadas de múltiples formas. Una de estas formas es la traducción de DAML+OIL en FOL (First Order Logic). La traducción consiste en mapear las declaraciones de DAML+OIL en sentencias FOL acompañadas de un set de axiomas FOL que restringen las interpretaciones de símbolos no lógicos (símbolos de relación, de función y restricciones) [99]. La traducción permite el uso de

teoremas y solucionadores de problemas para responder a consultas y buscar inconsistencias lógicas en teorías representadas por DAML+OIL. A diferencia de las descripciones estándar de la semántica del modelo teórico, las restricciones en el mapeo son representadas como axiomas en un lenguaje en el cual existen herramientas para el razonamiento automático. Estas herramientas pueden ser utilizadas para fiscalizar las restricciones por inconsistencias y redundancias, y son de particular importancia en la especificación semántica de lenguajes de desarrollo (como DAML+OIL), ya que pueden ayudar a los desarrolladores a depurar y comprender las consecuencias de cambios en el lenguaje.

2.2.3.4. OWL (Web Ontology Language)

A partir de RDF han surgido otras tecnologías con mayor expresividad y capacidad de razonamiento para representar los conocimientos que contienen las ontologías, con el fin de poder representar de la forma más potente posible el campo de conocimiento de cada dominio. La más significativa es el lenguaje OWL, que está basado en RDF. El lenguaje OWL es un lenguaje de marcación semántica recomendado por W3C (World Wide Web Consortium) para publicar y compartir ontologías en la Web. Permite describir clases, propiedades e instancias a través de tres especificaciones: OWL Lite, OWL DL y OWL Full que dan tres niveles crecientes de expresividad. OWL Lite está diseñado para aquellos usuarios que necesitan sobre todo una jerarquía de clasificación y restricciones simples ofreciendo menor complejidad formal que OWL DL. OWL DL debe su denominación a la Lógica de Descripción (Description Logic). DL es el nombre de una familia de formalismos de representaciones de conocimientos que representan el campo de conocimiento de un dominio de aplicación: definiendo primeramente los conceptos relevantes del dominio (su terminología), y luego usando esos conceptos para especificar las propiedades de objetos e instancias que toman lugar en el dominio. En DL, la terminología especifica el vocabulario de un dominio que consta de conceptos y roles, donde los conceptos denotan instancias, mientras los roles denotan relaciones binarias entre esas instancias. OWL DL es básicamente una Lógica Descriptiva, excepto que en OWL DL los símbolos para clases, propiedades, etc. son referencias de tipo URI, en lugar de nombres usuales tipo string, y la semántica del modelo teórico incluye semánticas para propiedades de anotación y propiedades de ontologías. OWL DL provee una sintaxis abstracta y una sintaxis RDF/XML, así como un mapeo de la sintaxis abstracta a la sintaxis RDF/XML [100]. OWL Full se caracteriza porque no impone restricciones en el uso de propiedades transitivas, pero a su favor posee una completa integración con RDFS. OWL DL y OWL Full utilizan el mismo vocabulario, aunque OWL DL está sujeto a algunas restricciones. De forma general, OWL DL requiere separación de tipos (una clase no puede ser un individuo o una propiedad, una propiedad no puede ser tampoco un

individuo o una clase). Esto implica que no se pueden aplicar restricciones a elementos del lenguaje OWL (algo que si es permitido en OWL Full). OWL Full es un soporte OWL bastante expresivo pero a la vez puede llevar a razonamientos sin solución. El razonamiento en OWL DL es en sí mucho más tratable, pero a la vez este posee muchas restricciones en instancias y clases. Mayores restricciones implican que las relaciones entre las clases sean restringidas a una pequeña serie de relaciones OWL. Por lo tanto, es necesario conocer cuales serán las instancias y las clases aplicadas para decidir que soporte expresivo se utilizará en una ontología escrita en OWL.

2.2.4. Recuperación de la información Ontológica

Uno de los principales problemas en la recuperación de la información es la variedad de interpretaciones de algunos conceptos, la relevancia de los datos o la exactitud en la consulta. Los metadatos son un enfoque muy importante para dotar de cierta semántica a la recuperación de la información. Hoy en día, las ontologías forman parte importante dentro del ámbito de investigación de la recuperación y organización de la información. La cantidad de información disponible en la red crece desmesuradamente haciendo que la gestión, mantenimiento y recuperación de información se convierta en un problema difícil de resolver. La relevancia en la recuperación resulta cada vez más difícil desde que el conocimiento no está sólo determinado por unidades físicas de información, sino que se ha convertido en un acervo distribuido de textos, servicios, aplicaciones, imágenes, sonidos, publicaciones electrónicas, etc., con formatos heterogéneos que conforman nuevas representaciones de conocimiento. Para evitar estos problemas se impulsaron diferentes soluciones para mejorar la recuperación de información. Una de esas soluciones fue el desarrollo de modelos de metadatos, estructuras de base para describir distintos objetos de información distribuidos de tal forma que la búsqueda basada en esos metadatos disminuyese el problema de la recuperación de información. En este contexto, surge lo que algunos denominan la "*Segunda Generación Web*" propiciada por el desarrollo del XML. Sobre la base de XML se han definido distintos lenguajes de marca para los diferentes tipos de documentos como RDF. Para recuperar la información en base de datos u otro tipo de almacenamientos se utilizan comúnmente lenguajes de consultas. Algunos lenguajes de consultas RDF son:

A) RQL

RQL [101] (RDF Query Language) permite el uso de variables para denotar los nombres de nodos (clases) y los arcos (propiedades). Una de las principales características de RQL, y que lo distinguen de otros lenguajes de consulta RDF, es

su capacidad de consultar esquemas RDF y descripciones RDF (instancias) en una misma consulta. RQL permite realizar consultas sobre esquemas RDF, Ej. class (que recupera todas las clases), property (recupera todas las propiedades) o sobre instancias (de clases y propiedades). Está definido por medio de un conjunto de consultas básicas e iteradores que son utilizados para construir otras consultas a través de una composición funcional. Utiliza tres cláusulas para construir una consulta:

- ***SELECT*** Define la proyección sobre las variables de interés.
- ***FROM*** Especifica una parte del subgrafo RDF.
- ***WHERE*** Una expresión formada por la comparación de variables.

```
SELECT service, $service_name
FROM kb:service{service:service_name}
WHERE $service_name <= kb:Car_Counter
USING = http://edsplab.us.es/ITSresearch/ITS.rdf#
```

Código 1. Sintaxis RQL para recuperar el servicio Car_Counter

RQL provee algunos operadores lógicos estándares como <, <=, =, >=, !, =, "like" o "in" como se puede ver en el Código 1. Todos estos operadores se pueden utilizar en valores literales (Ej. string, integer, real, date...) o recursos (URIs). Permite además los operadores lógicos "and", "or" y "not"

B) RDQL

La sintaxis RDQL [102] (RDF Data Query Language) es similar a la de SQL y se relaciona de cerca con SquishQL [103], el cual a su vez está basado en RdfQL, un lenguaje de bases de datos escalable creado para trabajar con Servicios Web Semánticos. RDQL utiliza 5 cláusulas para construir una consulta:

SELECT - SOURCE - WHERE - AND - USING

Por ejemplo, la sintaxis básica de una sentencia RDQL para recuperar un servicio activo encargado de determinar el número de coches en una coordenada específica de la ciudad de Sevilla sería:

```
SELECT ?Car_Counter, ?City, ?Coordinate, ?TrafficCamera
FROM <ITS.rdf>
WHERE (?TrafficCamera, <service:status>, ?active),
      (?TrafficCamera, <service:name>, ?Car_Counter),
      (?TrafficCamera, <service:city>, ?Seville)
AND ?Coordinate = "37.10N 5.550"
USING service FOR <http://edsplab.us.es/ITSresearch>
```

Código 2. Sintaxis RDQL

RDQL se presenta como una evolución de varios lenguajes. Consiste en un patrón gráfico expresado como una lista de patrones de triples. Cada patrón se compone de variables de nombres y valores RDF (URI y literales). Una consulta RDQL puede tener un conjunto de restricciones sobre los valores de las variables.

C) SPARQL

En 2008, el RDF Data Access Working Group (Parte de Semantic Web Activity), publicaron un lenguaje estándar de consultas para RDF llamado SPARQL [104] (SPARQL Protocol and RDF Query Language) que se ocupa de las necesidades básicas en consultas RDF dejando abiertas varias cuestiones a futuro: inclusión de vocabularios RDFS, rutas de localización, anidamientos, premisas, etc. [105]. Este lenguaje esta inspirado en SQL y adopta varias de sus palabras claves como SELECT, DISTINCT, ORDER BY, FROM, WHERE, PREFIX, OPTIONAL, UNION y GRAPH. Esta definida por BGPs (Basic Graph Patterns), que es un set de patrones de triples donde un patrón triple es una estructura de tres componentes que pueden ser fijos o variables. Los tres componentes que forman un patrón triple son respectivamente llamados sujeto, predicado y objeto. Los sets de patrones triples (ej. BGPs) son fundamentales para las consultas SPARQL ya que éstos especifican el acceso a los datos RDF [106]. BGPs es una conjunción de patrones de accesos simples (SAPs). Un SAP es un triple cuyos elementos representan cualquier combinación atómica o variable, donde las variables son precedidas por "?". Las combinaciones atómicas en los SAPs son llamados patrones ligados mientras que las variables son patrones independientes. Un SAP selecciona triples donde coinciden todos los patrones (ligados o independientes) [107]. El lenguaje SPARQL incluye IRIs, un subset de referencias RDF URI que omite espacios. Los IRIs incluyen URIs y URLs, respectivamente. El protocolo SPARQL contiene una interfaz SparqlQuery que a su vez contiene una sola operación "query". Este protocolo es descrito de manera abstracta en WSDL 2.0 [WSDL2] [108] como un servicio Web que implementa la interfaz, tipos, fallas y operaciones como los enlaces HTTP y SOAP. La operación query es descrita como un patrón de intercambio de mensajes In/Out. Las restricciones de estos mensajes de consultas consisten exactamente de dos mensajes, en el siguiente orden [109]:

1. Mensaje:
   - Indicado por la referencia de la interfaz de mensajes, componente cuya {etiqueta de mensaje} es "In" y la {dirección} es "In".
   - Recibido desde algún nodo N.

2. Mensaje:
   - Indicado por la referencia de la interfaz de mensajes, componente cuya {etiqueta de mensaje} es "Out" y la {dirección} es "Out".
   - Enviado al nodo N.

Algunas de las principales cláusulas de SPARQL [104] podrían ser:

SELECT - Es una cláusula requerida en toda consulta, similar a la utilización indicada en SQL. Define las variables a retornar como resultados cuyos nombres deben empezar con "?" y separadas por ",".

CONSTRUCT - Esta cláusula retorna un simple grafo RDF especificado por una plantilla. Construye un grafo de respuesta en vez de devolver una típica tabla de variables y valores.

DESCRIBE - Representa una cláusula que devuelve un grafo RDF describiendo los recursos encontrados. Toma cada recurso identificándolo y nombrándolo por una IRI que luego ensamblará en un único grafo RDF tomando una descripción proveniente de cualquier información disponible en el set de datos RDF. Ej. La cláusula DESCRIBE puede adoptar IRIs para identificar recursos.

ASK - Devuelve una variable booleana indicando si la combinación SPO (Sujeto, Predicado, Objeto) de una consulta existe en la ontología RDF consultada. Es utilizado para comprobar si un patrón de consulta tiene la solución. Si no existe solución, no se retorna la información.

Otros recursos del set de datos SPARQL:

FROM - Identifica los datos sobre los que se ejecutará la consulta. En caso de no estar en la consulta, el motor de proceso toma como fuente de datos todos los grafos disponibles. Si existieran múltiples FROM, la consulta se realiza sobre la fusión de todos los grafos RDF nombrados.

WHERE - Representa el patrón de la consulta con una o mas tripletas encerradas entre "{}". Esta cláusula es requerida en el caso de utilizar las cláusulas QUERY, RESULT, FROM y ASK.

FILTER - Impone constricciones adicionales al patrón de búsqueda. Restringe soluciones a aquellos donde la expresión FILTER evalúa como TRUE.

SPARQL también presenta, así como SQL algunas funcionalidades de ordenamiento y selección de resultados:

ORDER BY - Tiene la misma funcionalidad que en SQL, establecer la solución en cierto orden. Es una secuencia de comparadores de orden compuestas de una expresión y un modificador de orden opcional (ASC() o DESC()). Cada comparador de ordenamiento puede ser ascendente o descendente.

LIMIT n - Restringe el numero de soluciones devueltas. Si el límite se pone a "0" daría lugar a que no se retornen resultados. El límite tampoco puede ser un número negativo.

SPARQL se presenta como el lenguaje más estable de consultas, suficientemente estable como establecerlo en la práctica de forma generalizada. El lenguaje RDQL es anterior a SPARQL, de hecho el diseño de RDQL es anterior a las actuales especificaciones RDF. SPARQL agrega información opcional de resultados en la consulta, disyunción en patrones gráficos y mayor expresión en pruebas (ej. date-time). La importancia de SPARQL radica en gran parte en su capacidad de expresar sus consultas en un lenguaje de alto nivel sin importar el formato en que estos datos han sido almacenados. El utilizar el lenguaje de consultas SPARQL nos permitirá realizar consultas orientadas puramente a contenidos semánticos, y no basándonos en la teoría de conjuntos como el modelo relacional.

2.2.5. Ontologías en el Campo de los ITS: Elecciones Tecnológicas

La creación de ontologías requiere un compromiso entre uso específico y reuso. Esto es, cuanto más ligada esté una ontología a un dominio específico, sus elementos terminológicos se pueden generalizar y reutilizar menos en otros dominios y tareas. Una ontología de clase A en un dominio específico solo puede ser compatible con otra que ofrezca prácticamente el mismo dominio. Si B desea compartir alguna información de A, la integración entre ambos podría acarrear inconsistencias de datos, redundancias y perdería atomicidad. Es necesario que los datos permanezcan aislados unos de otros. Si se pretende incorporar una ontología en el campo de los ITS, debe haber un gestor que se encargue de coordinar el trabajo cooperativo entre ontologías y dispositivos gestionando la interoperabilidad de datos de formato común. El uso de un servicio que gestione la interoperabilidad entre ontologías de diferentes proveedores ahorraría espacio de almacenamiento y tiempo de rediseño.

Existen una infinidad de gestores y herramientas ontológicas para diferentes tipos de lenguajes, OWL, RDF, RDFS, etc. Estos gestores a su vez se pueden encontrar en distintos lenguajes de programación como C, C++, Java, Python, Perl, etc. Implementar y gestionar una ontología no es tarea sencilla, más aún teniendo en cuenta que la mayoría de los dispositivos implementados en el campo de los ITS se encuentran diseminados a través de redes distribuidas y comúnmente estos carecen de compatibilidad entre sí. Otro problema a tener en cuenta es que estos dispositivos son en su mayoría embebidos. En el caso de las herramientas y gestores que utilizan Java como base, estas deberán ser portadas a equipos heterogéneos que usan procesadores sobre los que no siempre es posible implementar una máquina virtual. Al momento de decidir qué lenguaje ontológico utilizar no queda duda que el mejor de éstos actualmente es OWL ya que se presenta como un estándar ontológico de facto dentro del W3C y es el lenguaje más utilizado por los razonadores lógicos [110]. El problema de utilizar OWL en dispositivos desplegados en el campo de los ITS es que las herramientas de gestión actuales están basadas en Java. En este sentido, RDFS como lenguaje ontológico y SPARQL como lenguaje de consultas se presentan como las mejores opciones, ya que existen herramientas de gestión para éstas tecnologías implementadas en C y C++, que pueden ser fácilmente portados a plataformas embebidas.

2.3. Servicios Colaborativos Inteligentes en ITS

El descubrimiento de servicios es un reto bien conocido en entornos distribuidos. Los ITS basados en tecnologías MAS están empezando a ser un enfoque importante para resolver problemas complejos en el sistema de transporte. Cuando se invoca a un servicio, se espera que este resuelva la petición y produzca los mejores resultados con un rendimiento aceptable. Sin embargo, descubrir y localizar servicios no es tarea fácil. Las búsquedas carecen de inteligencia creando retardos y múltiples inconvenientes en entornos críticos del transporte urbano, rural y metropolitano. Un agente que realiza una petición podría obtener gran cantidad de información sin sentido o recibir información que no corresponda con la búsqueda. En el campo de los ITS, donde la exactitud y rapidez de la información enviada/recibida es sumamente importante, urgen mecanismos que se adapten mejor a los cambios futuros y adopte cierta semántica en sus operaciones.

2.3.1. Ontologías en los Sistemas de Control de Tráfico

El control del tráfico urbano es extremadamente complejo y dinámico. Los UTCS (Urban Trafic Control System) están usualmente basados en redes

centralizadas. No obstante, los sistemas de control actuales deben ser capaces de gestionar los cambios indeseables en el flujo del tráfico de forma dinámica. Por ejemplo, en caso de accidentes, deben ser capaces de optimizar el flujo mediante ajustes de las señales de tráfico y coordinar las operaciones para cada señal [111]. La incorporación de semánticas en la localización, invocación, descubrimiento, composición, ejecución y monitorización de servicios, se muestra como una solución factible. El objetivo de aplicar ontologías en el campo de los ITS es la localización de servicios y la búsqueda de información de forma inteligente, así como posibilidad de reutilizar las KB's (Knowledge Bases) existentes. Los dispositivos que cuenten con servicios que hagan uso de ontologías o interactúen en función a técnicas de conocimiento podrán utilizar nuevas ontologías favoreciendo la ágil transferencia del conocimiento. El desarrollador tendrá que preocuparse más de qué información podrá ser consumida por los dispositivos que enfocarse a desarrollar complejos sistemas de tráfico urbano C/S. Incluso podría utilizarse, por ejemplo, una ontología del estándar IEEE-1451.4-2004 [112] que albergue información (Datasheets) de sensores y actuadores, proporcionando la re-calibración y el ajuste automático a éstos dispositivos. Dada la creciente densidad vehicular en áreas donde la implementación de los ITS va en aumento, los importadores y/o dispositivos necesitarán tomar decisiones inteligentes para encontrar servicios e informaciones más adecuadas, de forma automática, minimizando retardos y aumentando el rendimiento del sistema. Deberán ser capaces de tomar decisiones de rendimiento, dejando de lado, de forma automática, servicios o mecanismos que le resulte de lenta ejecución, para tomar medidas que le ofrezcan mejores resultados. La incorporación de la inteligencia y la toma de decisiones autónomas DD en entornos ITS pueden evitar gran parte de los actuales caos vehiculares y ayudará en gran medida a la creación e interacción de un entorno amigable entre el humano y la máquina.

2.3.1.1. ¿Por qué ontologías y no Trading para localizar servicios?

Un importador puede tener una idea del tipo de objeto que necesita, pero puede no tener toda la información necesaria para hacer una elección precisa. Las ventajas que ofrece el servicio de Trading Service de CORBA son evidentes desde el punto de vista de descubrir servicios de tipo particular. Sin embargo, aplicar mecanismos clásicos de búsqueda, en base a propiedades básicas, puede acarrear resultados ineficientes. Por otra parte, los importadores que realizan peticiones sobre algún servicio en particular necesitan estar capacitados de adquirir un cierto nivel de verdad sobre un servicio en particular. Los servicios Trading son incapaces de proveer este nivel de búsqueda. El uso de importadores inteligentes con mecanismos de verdades reducirían los problemas de selección de un servicio en particular. Se podría hacer, por ejemplo, uso de un servicio mediador (gestor de

ontologías) que trabajara como un Trading pero interoperando con ontologías. Gracias a la implementación de este mediador sería posible extraer automáticamente datos de ontologías ubicadas en URI's, procesarlas y sacar conclusiones de ellas. Este medidor podría tomar decisiones y negociar con otros servicios, dispositivos y ontologías en diferentes localizaciones. Implementar este tipo de mediador en entornos ITS podría ser utilizado para hacer búsquedas específicas de información en servicios como meteorológicos, tráfico, accidentes, eventos especiales, etc., que satisfaga las preferencias de búsqueda. Por ejemplo, consideremos un dispositivo que consulte: *La temperatura actual en Sevilla Capital*. El mediador utilizará una ontología de datos meteorológicos, digamos meteorology_sev.rdfs, y empleará los datos para mostrarlo en un panel de tráfico o a algún dispositivo que lo solicite. El uso de ontologías en sistemas distribuidos, específicamente en cuanto a elección de servicios por parte del importador se refiere, puede reducir sustancialmente la tasa de fallo y más aún en un entorno donde la información recibida es de vital importancia como el entorno de los ITS. Las ontologías proveerán el nivel semántico y los medios adecuados para especificar los mejores mecanismos de interoperación basados en la representación lógica de la información [113].

2.3.2. Arquitectura de un Servicio Semántico Inteligente

Una arquitectura semántica se presenta como un entramado de componentes funcionales aprovechando diferentes estándares, APIs, reglas y procesos. Un Servicio Semántico o SS, permitirá integrar una amplia gama de productos y servicios informáticos de manera que puedan ser utilizados eficazmente por dispositivos ITS en un entorno distribuido. El SS propuesto haría uso principalmente de ontologías basándose en teorías del conocimiento y el procesamiento del lenguaje natural. Los diferentes servicios de tráfico, implementados en su mayoría en dispositivos con arquitecturas embebidas, interoperarían con el SS quien sería el encargado de proveer y gestionar toda la información de los C/S intermediando entre ellos de manera dinámica e inteligente. El SS cooperativo haría uso de los datos de operaciones desde el momento de la recopilación de la información, automatizando la toma de decisiones ante situaciones que requieran intervención inteligente en el entorno de los ITS. Uno de los problemas a tener en cuenta en la semántica ontológica es como superar los problemas de homonimia. En este sentido, se encontraron los mismos problemas que a nivel de lenguaje de modelado, pero esta vez asociados a los términos del lenguaje natural que van a utilizarse para representar conceptos. Un constructo puede tener un significado inequívoco, por ejemplo, ser una clase, pero el término con el se anota ese constructo es un término que pertenece al lenguaje natural, y por tanto está sometido a las ambigüedades del mismo, tales como la homonimia. Una

de las características referente al descubrimiento semántico de un servicio es que se basa en un juego de palabras clave. Estas palabras clave pueden ser sintácticamente equivalentes pero semánticamente diferentes (homonimia). En un SS, los problemas de homonimia podrían ser superados con el uso de ontologías en la que se especificarían conceptos de recursos como sujetos únicos pero a la vez sus literales podrían ser iguales. La declaración de espacios de nombres (namespace) permite eliminar las ambigüedades y solucionar los problemas de homonimia. Los espacios de nombres son colecciones de nombres identificados mediante una referencia URI (Uniform Resource Identifier), que se utiliza como tipos de elementos y nombres de atributos. El URI combina URNs (Uniform Resource Names) y URLs (nombres/direcciones) para identificar de forma universal los recursos. Gracias a la diferenciación por espacios de nombres, los sujetos, predicados u objetos pueden ser homonómicamente iguales entre otras declaraciones dentro de un mismo esquema. Diferentes servicios en diferentes dispositivos podrían por ejemplo tener la misma propiedad serv_name, el mismo objeto Car_Counter, pero ofrecer otro tipo de información Código 3.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:lab="http://www.lab.es/lab#"
         xmlns:kb="http://www.edsplab.us.es/kb#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://www.lab.es/lab#Car_Counter">
    <lab:serv_name>Car_Counter</lab:serv_name>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.edsplab.us.es/kb#Car_Counter">
    <kb:serv_name>Car_Counter</kb:serv_name>
  </rdf:Description>


</rdf:RDF>
```

Código 3. Homonimia en la propiedad serv_name y en el objeto Car_Counter

Los repositorios de servicios y aplicaciones en los ITS actuales no comparten en casi ningún sentido información semántica. Hay una baja disponibilidad de Frameworks que facilitan el descubrimiento, composición y ejecución de servicios distribuidos. Algunas ventajas que de la incorporación de un Servicio Semántico en éste ámbito serían:

A) Mayor expresividad en la representación del conocimiento y el soporte a ontologías. Cuanto más expresiva se encuentre la información se representará de manera más compacta.

B) Soporte con razonamiento sin ambigüedades y soporte de razonadores/motores de inferencia lógica.
C) Compatibilidad Web gracias a su capacidad de trabajar con lenguajes semánticos utilizados en las Webs actuales.
D) Utilización de metadatos en aplicaciones distribuidas y recuperación de la información ofreciendo un modelo de programación más sencillo y eliminando la necesidad de tener que intercambiar archivos `IDL`, archivos de encabezado o cualquier método externo de referencia a componentes y objetos.
E) Cooperación de la información Web y la información de aplicaciones distribuidas.
F) Estandarización de las recomendaciones `W3C` existentes Ej. `XML`, `RDF`, `RDFS`.
G) Incorporación de nuevos niveles de funcionalidad y cooperación entre aplicaciones middleware y la Web.

Gracias a las capacidades semánticas y a la utilización de ontologías para el almacenaje e intercambio de metadatos entre dispositivos, aplicaciones y servicios, las características fundamentales que deberá cumplir un Servicio Semántico pueden clasificarse como:

A) ***Publicación*:** Hacer disponible la descripción de las capacidades de un servicio.
B) ***Descubrimiento*:** Localizar diferentes servicios que cumplan con los requerimientos para efectuar una tarea concreta.
C) ***Selección*:** Elegir el servicio más apropiado a través de una recopilación exhaustiva de valores de subcriterio.
D) ***Composición*:** Combinar servicios para alcanzar un objetivo.
E) ***Mediación*:** Resolver desajustes en el procesado de los datos y servicios combinados ofreciendo la intermediación de la comunicación entre el cliente y el servidor semántico.
F) ***Ejecución*:** Invocar servicios siguiendo convenios pragmalingüísticos en el contexto de la interpretación del significado.
G) ***Acción*:** Actuar en base a eventos producidos entre clientes y servidores semánticos.

Las principales características que debe cumplir un Servicio Semántico en ejecución son:

A) ***Monitorización*:** Control de la ejecución de los procesos.
B) ***Compensación*:** Proveer soporte de transacciones deshaciendo o migrando efectos no deseados.
C) ***Reemplazamiento*:** Facilitar la substitución de servicios similares y duplicados.

D) ***Inspección*:** Verificar que la ejecución de los servicios se comporten de la manera esperada.
E) ***Consulta*:** Ofrecer mecanismos de consultas a través de lenguajes como RDQL o SPARQL.
F) ***Intermediación*:** Arbitrar en la comunicación entre clientes y servidores.

Las principales funciones de un Servicio Semántico desarrollado e implementado en la arquitectura CORBA son:

A) Almacenar referencias a objetos IOR CORBA de los ITS en literales haciéndolas accesible de manera semántica a clientes y servidores semánticos.
B) Proveer una biblioteca digital inteligente que facilite el intercambio del conocimiento entre dispositivos y aplicaciones C/S.
C) Proporcionar la funcionalidad que le permita a los clientes localizar objetos o información de interés de manera más descriptiva e inteligente a través de literales.
D) Organizar las categorías en jerarquías y proporcionar extensibilidad a través de un refinamiento de subcategorías.
E) Ofrecer un mecanismo de razonamiento de software independiente a la ontología y definida sobre un conjunto de proposiciones simples RDF y RDFS.
F) Representar los recursos formalmente utilizando tripletas (sujeto, predicado, objeto). Donde S representará un recurso o sujeto (miembro de resources), P representará la propiedad o predicado (miembro de properties) y O podría ser tanto un recurso como un literal o un objeto (miembro de literals).
G) Representar declaraciones simples sobre recursos como grafos de nodos y arcos que representen los recursos, propiedades y valores.
H) Proporcionar un marco abstracto y conceptual para definir y describir detalladamente un servicio ofertado dentro del dominio de los ITS.
I) Ofrecer una interoperabilidad semántica aprovechando las bondades de comunicación CORBA.
J) Trabajar de manera independiente a la plataforma (mac, pc, unix, etc.), y reducir en gran medida la interoperabilidad heterogénea entre dispositivos.

Capítulo 3 - Middlewares Inteligentes basados en ITS - Propuesta para la Creación de Entornos Inteligentes Cooperativos en el Ámbito de los ITS

3.1. Introducción

En este capítulo se detallan los fundamentos teóricos que sirven de base al diseño de la investigación, así como los métodos empleados. En primer lugar, se presenta el método empleado en la medición de la carga computacional de la capa middleware en entornos embebidos. El objetivo del método consistirá en analizar y cuantificar el rendimiento de la capa middleware en sistemas embebidos en términos de transacciones procesadas por segundo. El resultado obtenido permitirá evaluar en qué medida la capa middleware puede afectar a la funcionalidad principal de los equipos instalados en las vías de transporte. En segundo lugar, se propone una metodología para la construcción de una ontología en el ámbito ITS, partiendo de una taxonomía y una colección de documentos. El objetivo principal del método propuesto consistirá en descubrir servicios ITS en base a patrones comunes entre los datos que permitan separar las muestras en jerarquías de clases para construir la ontología. En tercer lugar, se presenta un Servicio Semántico desarrollado en CORBA y enfocado principalmente a interactuar e intermediar en la comunicación de clientes y servidores semánticos en un sistema distribuido ITS. El objetivo principal de la arquitectura semántica propuesta será el de gestionar la información de la ontología desarrollada como resultado de los métodos anteriormente propuestos e interoperar principalmente con servicios implementados en dispositivos embebidos de entorno urbano tales como cámaras, semáforos, paneles de tráfico, limitadores de velocidad, etc. La incorporación del Servicio Semántico en entornos de tráfico le dará un enfoque inteligente a la comunicación e interpretación de servicios de tráficos urbanos/metropolitanos/rurales, diseminados actualmente en un entorno distribuido de difícil interoperabilidad.

3.2. Método para la Evaluación de la Carga Computacional de CORBA en los ITS

Los middleware proveen la reusabilidad de servicios que pueden ser compuestos, configurados y desarrollados para crear aplicaciones DRE distribuidas en tiempo real y ser portados en una gran variedad de sistemas embebidos de forma rápida y robusta [114]. Asimismo, la capa middleware ofrece capacidades críticas a los sistemas distribuidos como el encapsulamiento en las comunicaciones y mecanismos de concurrencias, evitando dependencias de hardware y software o permitiendo la gestión de servicios distribuidos. No obstante, añadir una capa middleware también supone añadir una mayor carga computacional a los dispositivos embebidos que forman parte de los equipos ITS. El proyecto MTE (Middleware Technology Evaluation) ha definido el procedimiento para evaluar los componentes y las tecnologías middleware [115]. Este proyecto considera el testeo de productos genéricos basados en laboratorio, evaluando el balanceo de carga en peticiones clientes como un componente clave del middleware.

3.2.1. Evaluación de Rendimiento en sistemas Embebidos

El siguiente método es utilizado para cuantificar los efectos de un servicio middleware en un sistema embebido que ejecuta una aplicación pesada que requiere un uso intensivo del procesador. El método utiliza los archivos `proc` disponibles los sistemas `GNU/Linux` para recuperar datos sobre el tiempo de `CPU` asignado a los procesos por el planificador (`scheduler`). Este método para medir el impacto de la capa middleware sobre plataformas embebidas fue publicada y sirve de base como aportación al desarrollo de esta tesis [116].

A primera vista, la solución podría ser la de medir la carga de la `CPU` con el comando `top` de `GNU/Linux`. Esta alternativa podría ser suficiente en algunos casos, pero en muchos otros podría distorsionar las mediciones como consecuencia de sus características y la propia carga de la `CPU`. Algunos ejemplos de situaciones en las que el comando `top` no ofrece mediciones exactas se describen a continuación:

- Cuando es necesario sincronizar el principio y el final del período de medición. Esto es muy importante en pruebas donde las peticiones de los clientes son testeadas con ráfagas para medir la escalabilidad del sistema. El comando `top` no puede ser sincronizado con estas ráfagas. Por lo tanto, un período de medición más amplio o más estrecho conduciría a resultados falsos.

- Algunos servidores suelen replicarse a sí mismos con el fin de procesar múltiples solicitudes de clientes al mismo tiempo. Estas replicaciones se llevan a cabo mediante la creación de procesos hijos que por supuesto, tienen sus propios `PIDs` y su propio archivo asociado en `/proc`. Si el número de procesos hijos es alto, `top` debe leer muchos archivos cada vez que se actualiza y la información que aparecería sería difícil de analizar, ya que la carga de `CPU` actual es la suma de todas las cargas de `CPU` de los procesos hijos. Hay otros casos en que este problema aparece aunque los servidores no creen procesos hijos. En versiones antiguas de sistemas `GNU/Linux`, los programas multi-hilo a menudo aparecían como procesos diferentes, ya que dicho núcleo asignaba un `PID` para cada hilo, aunque los trate de una manera diferente. El servidor debe ser considerado como una aplicación multi-proceso. En versiones recientes del kernel `Linux`, este problema está resuelto ya que un servidor multi-hilo puede ser visto como un solo proceso.

- Cuando la velocidad de actualización es alta o cuando hay muchos procesos monitorizados, `top` puede consumir una cantidad significativa de tiempo de `CPU` y produciría medidas poco realistas.

Cuando el uso de top no es posible debido a sus limitaciones, se requiere un método más ligero y selectivo. Un procedimiento adecuado para medir la carga de CPU promedio de servicios middleware debería cumplir estas condiciones:

- El servidor de middleware puede crear muchos procesos hijos.
- La versión del núcleo Linux considera a los programas multi-hilos como procesos.
- Las peticiones del cliente se puede modelar como ráfagas.
- El período de medición puede ser configurado y se debe realizar durante la ráfaga de peticiones.

Para medir los efectos del middleware de una aplicación en particular que se ejecuta en un sistema embebido se debe comparar la carga promedio de CPU en dos casos: sin y con el servidor middleware.

Las mediciones en el primer caso, se puede hacer directamente con el comando top, observando el porcentaje de tiempo de CPU que el scheduler asigna a la aplicación principal. Los efectos indeseables de carga del comando top puede ser minimizado forzando a top a recuperar sólo la información del tiempo de CPU de la aplicación principal. La Figura 11 muestra el promedio y el tiempo de CPU acumulado requerido por la aplicación principal (proceso epi.out) sin el middleware. Esta aplicación principal utiliza normalmente más recursos de procesador, por lo que su curva está muy cerca del tiempo de CPU acumulado total. Se debe tener en cuenta que todas estas curvas tienen siempre una pendiente entre 0º y 45º. Cuanto mayor sea la curva, mayor será el porcentaje de uso de CPU.

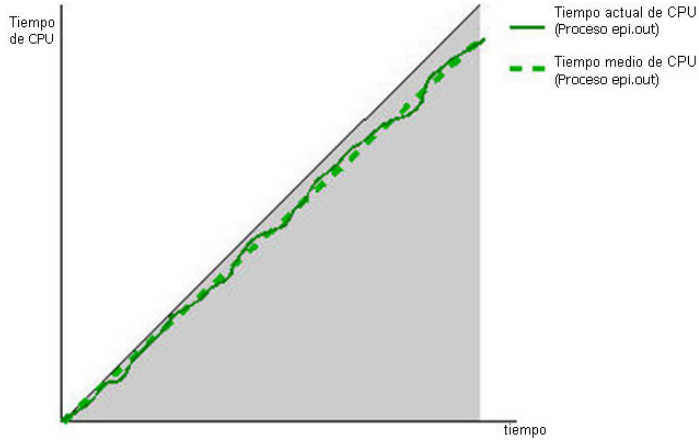


Figura 11. Tiempo de CPU promedio y acumulado sin el servidor middleware

En el segundo caso, el comando `top` no puede ser utilizado debido a los problemas mencionados anteriormente, pero ya que toda la información recopilada por el comando `top` se encuentra disponible en los archivos especiales dentro de la carpeta `/proc`, es posible insertar unas pocas líneas de código en el servidor middleware para leer estos archivos y extraer sólo la información relevante en ciertos momentos. Cuando existen solamente unos pocos procesos a considerar, esta solución puede ser adecuada y sus efectos en la medición (en términos de carga de `CPU` adicional) son insignificantes. Por ejemplo, si tanto la aplicación principal y el servidor middleware son ejecutados por simples procesos, el código de medición podría abrir sólo tres ficheros: uno con respecto a la aplicación principal, otro con respecto a la capa de middleware y otro fichero que contiene el tiempo total de `CPU`. Desafortunadamente, esto no siempre es cierto ya que algunos servidores middleware como los de `MICO` crean múltiples hilos sobre demanda y el número de `PIDs` puede ser bastante grande, por lo que tomaría un tiempo considerable leer todos los archivos asociados a éste.

Para solucionar éste problema se debe aprovechar el hecho de que el servidor middleware apenas consume tiempo de `CPU` cuando este no recibe peticiones de los clientes. Se puede asumir entonces que la curva de tiempo de `CPU` es plana en estos periodos, como se muestra en la Figura 12. Debido a que los valores contenidos en los ficheros `proc` asociados con el servidor no cambian durante los periodos de inactividad (antes y después de la ráfaga de peticiones del cliente) es posible realizar las mediciones dentro de estos períodos (`tm1`, `tm2`) a pesar de sus largas demoras. Los valores obtenidos deben ser los mismos que se obtendrían si se hubieran tomado las mediciones al principio y al final de la ráfaga.

$$TCPU_{srv}(t_{m1}) = TCPU_{srv}(t_s)$$
$$TCPU_{srv}(t_{m2}) = TCPU_{srv}(t_e) \qquad \text{Ec. (1)}$$

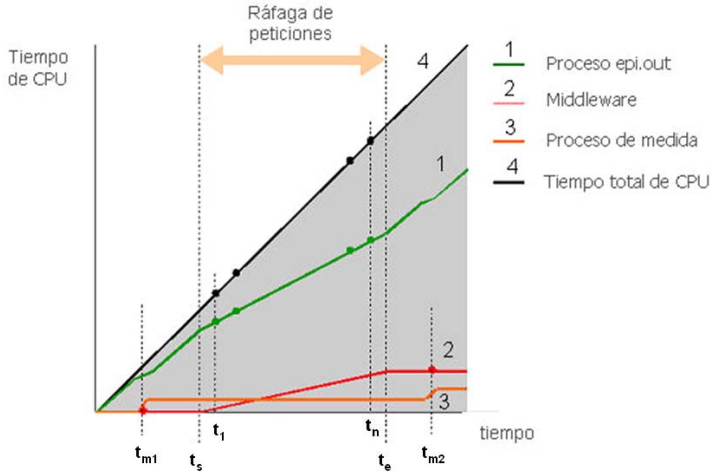


Figura 12. Tiempo de CPU acumulado con el servidor middleware

Estas mediciones prolongadas se deben tomar en cualquier momento antes y después de la ráfaga por una aplicación simple. Esta aplicación debe obtener primero la lista de los `PIDs` relacionados con el servidor y luego recuperar los tiempos de `CPU` acumulados para cada fichero `proc` relacionado con la lista de los `PID`. El tiempo de `CPU` acumulado total del servidor es la suma de todos estos valores. Esta tarea se puede lograr fácilmente mediante un `script bash`.

Con el fin de calcular la carga del proceso principal, se necesita tomar las mediciones al principio y al final de la ráfaga. Esto se puede hacer mediante la inserción de unas pequeñas líneas de código en la rutina del servicio, de modo que recupere la información de los ficheros `proc` relacionadas con el proceso principal y el sistema cuando el servidor recibe la primera y la última petición. No obstante, esta solución tiene dos problemas:

1. Como determinar la primera y la última petición.
2. A pesar de que el código podría determinar la primera y la última petición, el proceso del servidor puede estar trabajando antes y después de ejecutar el código de medición, por lo tanto, esto podría consumir un poco más tiempo de lo estimado.

Una forma sencilla de resolver el primer problema es la de marcar la primera y la última petición por parte del cliente, pero esto puede fallar en algunas circunstancias. Esto es particularmente cierto cuando la tasa de peticiones (peticiones por minuto) es alta y el servidor inicia varios procesos hijo o hilos que no comparten ningún espacio de memoria. El `scheduler` en el lado servidor podría tratar el hilo sirviendo la primera petición de manera que las peticiones siguientes entrantes queden en espera hasta que la petición anterior haya sido tratada o haya sido marcada con 1. Una solución sencilla cuando la comunicación entre `procesos/hilos` no es posible es marcar el primero y las últimas n peticiones, siendo n un número pequeño. Después de terminar la ráfaga, mantener sólo los valores menores y mayores.

El segundo problema es difícil de evitar, pero puede ser mitigado si la ráfaga es lo suficientemente larga. Dado que estamos interesados en el comportamiento promedio, podemos hacer la ráfaga de peticiones tan grande como la necesitemos. Cuanto más larga es la ráfaga, menor será el efecto. Las expresiones siguientes muestran esto desde un punto de vista matemático. La carga promedio de `CPU` del proceso principal (en %) se define como:

$$L_{MAIN} = 100\left(\frac{TCPU_{MAIN}(t_e) - TCPU_{MAIN}(t_s)}{TCPU_{SYS}(t_e) - TCPU_{SYS}(t_s)}\right) = \\ = 100\left(\frac{TCPU_{MAIN}(t_e) - TCPU_{MAIN}(t_s)}{t_e - t_s}\right) \qquad \text{Ec. (2)}$$

Asumiendo t_1 y t_n como el tiempo de la primer y última medición, respectivamente, la carga media de la `CPU` del proceso principal de acuerdo con el método propuesto es:

$$L'_{MAIN} = 100\left(\frac{TCPU_{MAIN}(t_n) - TCPU_{MAIN}(t_1)}{TCPU_{SYS}(t_n) - TCPU_{SYS}(t_1)}\right) = \\ = 100\left(\frac{TCPU_{MAIN}(t_n) - TCPU_{MAIN}(t_1)}{t_n - t_1}\right) \qquad \text{Ec. (3)}$$

Las diferencias entre los tiempos de mediciones esperados y los actuales son:

$$\Delta t_s = t_1 - t_s \; ; \Delta t_e = t_e - t_n \qquad \text{Ec. (4)}$$

Los errores en las mediciones dados los problemas explicados anteriormente pueden ser definidos como;

$$e_{Ms} = TCPU_{MAIN}(t_1) - TCPU_{MAIN}(t_s)$$

$$e_{Me} = TCPU_{MAIN}(t_n) - TCPU_{MAIN}(t_e) \qquad \text{Ec. (5)}$$

Sustituyendo las definiciones anteriores en la expresión del promedio de carga del CPU estimado, el resultado está dado por la ecuación (6).

$$L'_{MAIN} = 100\left(\frac{TCPU_{MAIN}(t_e) + e_{Me} - TCPU_{MAIN}(t_s) - e_{Ms}}{t_e - t_s - (\Delta t_s + \Delta t_e)}\right) \qquad \text{Ec. (6)}$$

Δt_e y Δt_s no dependen de la longitud de la ráfaga de peticiones, así que si la ráfaga es lo suficientemente larga se pueden asumir las siguientes hipótesis:

$$t_e - t_s >> \Delta t_s + \Delta t_e \qquad \text{Ec. (7)}$$

$$L'_{MAIN} \approx 100\left(\frac{TCPU_{MAIN}(t_e) + e_{Me} - TCPU_{MAIN}(t_s) - e_{Ms}}{t_e - t_s}\right)$$

Puesto que la carga de la CPU de cualquier proceso que se ejecuta, siempre se encuentra por debajo de 100%, e_{Me} y e_{Ms} son menores o igual a Δt_e y Δt_s, respectivamente. Estos errores son independientes de la longitud de la ráfaga, así que:

$$L'_{MAIN} \approx 100\left(\frac{TCPU_{MAIN}(t_e) - TCPU_{MAIN}(t_s)}{t_e - t_s}\right) = L_{MAIN} \qquad \text{Ec. (8)}$$

ya que $t_e - t_s >> e_{Me} - e_{Ms}$.

La carga de la CPU del servidor middleware se puede estimar utilizando la ecuación (9).

$$L_{SRN} = 100\left(\frac{TCPU_{SRV}(t_e) - TCPU_{SRV}(t_s)}{TCPU_{SYS}(t_e) - TCPU_{SYS}(t_s)}\right) = \left(\frac{TCPU_{SRV}(t_{m2}) - TCPU_{SRV}(t_{m1})}{t_e - t_s}\right) \quad \text{Ec. (9)}$$

Además de las mediciones de carga de la CPU, cierta información útil, tal como el consumo de memoria, también puede ser extraída de los archivos proc sin que esto sea una carga adicional significativa.

El método propuesto ayudará a optimizar servicios middleware y minimizar su impacto sobre otras aplicaciones [116].

3.3. Método para la Creación de una Ontología en el Ámbito de los ITS

El problema común con las tecnologías mencionadas en el capítulo anterior surge principalmente en la interoperabilidad de información entre servicios y dispositivos que son parte de los ITS. El uso de ontologías en este campo combinado con un servicio que la gestione representa una posible solución a estos problemas.

Para desarrollar una ontología no existe un único modo correcto de modelar un dominio; al contrario, siempre hay alternativas posibles ya que son comúnmente hechas a mano. La mejor solución casi siempre depende del propósito u objetivo final de la ontología, o de las aplicaciones en las que vaya a emplearse. No obstante, es importante desarrollar técnicas que ayuden a realizar de manera automática estas labores para ahorrar tiempo de diseño y estandarizar en cierta forma el proceso y los mecanismos de construcción.

A partir de aquí se implementan técnicas de identificación de información relevante dentro de los textos de partida. A continuación se aplican técnicas de análisis semántico que permiten reducir la dimensionalidad de la matriz términos-documentos y técnicas de análisis semántico que proporcionarán las relaciones entre los términos.

3.3.1. Mecanismos y Estrategias para la Construcción de la Ontología

La Figura 13 muestra una representación gráfica en forma de diagrama de bloques la metodología general propuesta para el desarrollo de la ontología. El objetivo principal de la metodología es descubrir servicios ITS basados en patrones

comunes de datos, con el objetivo final de obtener jerarquías de clases en el bloque "*Construir la Ontología*".

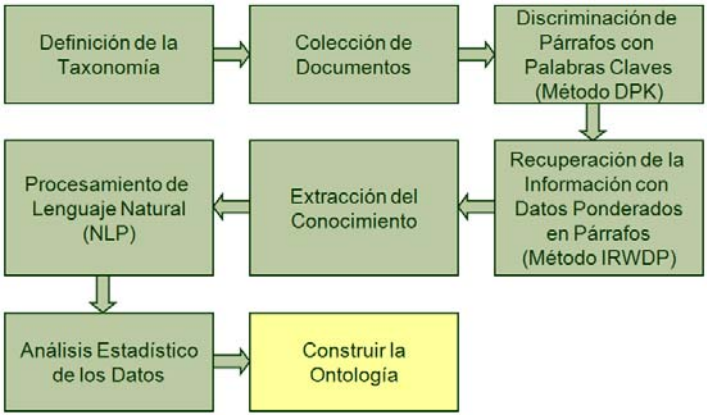


Figura 13. Diagrama de Bloques: Metodología Propuesta

En esta metodología se propone la incorporación de métodos automáticos que nos permitan desarrollar técnicas de meta-análisis sobre documentos. Los meta-análisis serán aplicados únicamente cuando los documentos incluidos en la revisión a juicio "*se puedan combinar razonablemente y pertenezcan a un área de dominio especifico*" efectuando una estadística cuantitativa de sus resultados. En base a los métodos propuestos, se construirá una ontología completa de servicios y contenedores de servicios en el dominio de los ITS. La ontología nos brindará un esquema conceptual a explotar a través de esquemas de intercambio de metadatos entre dispositivos y aplicaciones inmersas en el sistema distribuido urbano. En los siguientes apartados se describe en detalle cada uno de los pasos contemplados en el esquema general de la metodología.

3.3.1.1. Definición de la Taxonomía

El dominio específico sobre el cual se trabajará en esta tesis es el de los ITS. Para desarrollar la ontología, es necesario partir de unos conceptos o clases base. Por ello se utilizará como plantilla o contenedor de servicios ITS parte de la taxonomía propuesta por la U.S. DOT (Deptartment of Transportation) y RITA (The Research and Innovative Technology Administration) [117], [118], [119]. En la Figura 14 se muestra una representación gráfica de parte de esta taxonomía, que se categoriza por niveles de detalle o LoD (Level of Detail).

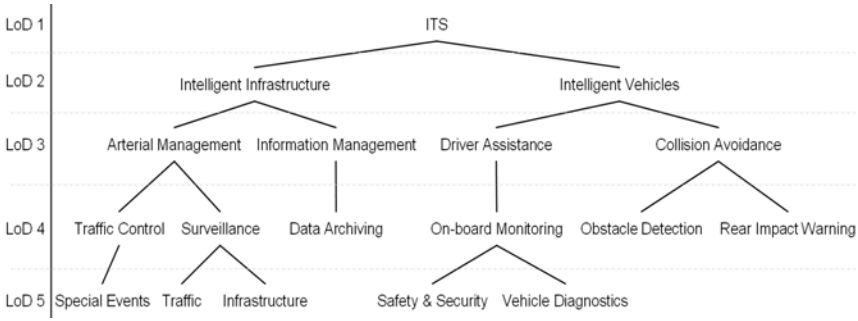


Figura 14. Parte de la Taxonomía de RITA U.S. DoT con LoD 5

La parte que se utilizará será hasta el LoD 4, que servirá como contenedor de servicios. Esta taxonomía dispone de una colección de 77 contenedores de servicios en el LoD 4. Por tanto, la investigación se enfocará principalmente en el LoD 5 donde, en base a los métodos propuestos, se buscarán servicios ITS más relevantes de la documentación recopilada en pares y triples de palabras.

3.3.1.2. Colección de Documentos

El siguiente paso es el de identificar la información relevante y el tamaño total de la colección de datos. Se opta por seleccionar las 10 revistas científicas con mayores índices de impacto en el campo de los ITS, y hacer una colección de las 30 publicaciones más relevantes de los últimos 10 años (2002 al 2012) por cada revista, totalizando 300 publicaciones, Tabla 1

ID	Revistas Científicas Consideradas	Nro. de publicaciones
A	Intelligent Transport Systems, IEE Proceedings	30
B	Intelligent Transport Systems, IET	30
C	Intelligent Transportation Systems Magazine, IEEE	30
D	Intelligent Transportation Systems, IEEE Transactions on	30
E	Vehicular Technology, IEEE Transactions on	30
F	Accident Analysis & Prevention	30
G	European Journal of Operational Research	30
H	Transportation Research Part A - Policy and Practice	30
I	Transportation Research Part B - Methodological	30
J	Transportation Research Part C - Emerging Technogies	30
	TOTAL	**300 publicaciones**

Tabla 1. Sitios para Recuperar la Información y Tamaño Total de la Colección

Uno de los inconvenientes cuando se utiliza una colección de documentos es que el peso que tiene cada palabra clave en cada uno de ellos cambia. Para solucionar este problema, en el último paso del proceso de obtención de los documentos se realiza la extracción y clasificación de éstos mediante un modelo de espacio vectorial para la clasificación temática de los documentos relevantes, así como una técnica parecida a la de feedback por relevancia para perfeccionar el proceso de IR. La técnica IR de feedback por relevancia es una de las estrategias más populares [120]. La idea principal en esta técnica consiste en que una vez identificados ciertos documentos previamente recuperados como relevantes o irrelevantes por el usuario, se utiliza la información que proporcionan para adaptar la consulta, de forma que se recuperen más documentos relevantes y menos irrelevantes en una búsqueda posterior [121]. El proceso de alterar una consulta en dirección a documentos relevantes es una técnica eficaz en la recuperación de la información de documentos enteros pero no así para partes del mismo. No obstante, la técnica feedback por relevancia se enfoca más a la extracción de documentos y no a trabajar con sus contenidos. En esta tesis, se propone una metodología alternativa. Conociendo el dominio de interés (ITS) y los contenedores de servicios de la taxonomía del apartado anterior, en los siguientes apartados serán analizados los documentos en estudio aplicando técnicas de discriminación de contenido, ahorrando tiempo y disminuyendo significativamente el tamaño de la muestra.

3.3.1.3. Discriminación de Párrafos con Palabras Claves (Método DPK)

El objetivo principal del método DPK (Discrimination of Paragraphs with Keywords) propuesto es la de recuperar sólo la información más relevante de la colección de documentos en base a discriminación de párrafos. El DPK es un método para seleccionar y discriminar párrafos, filtrar los resultados y organizar las palabras con sus respectivas frecuencias absolutas.

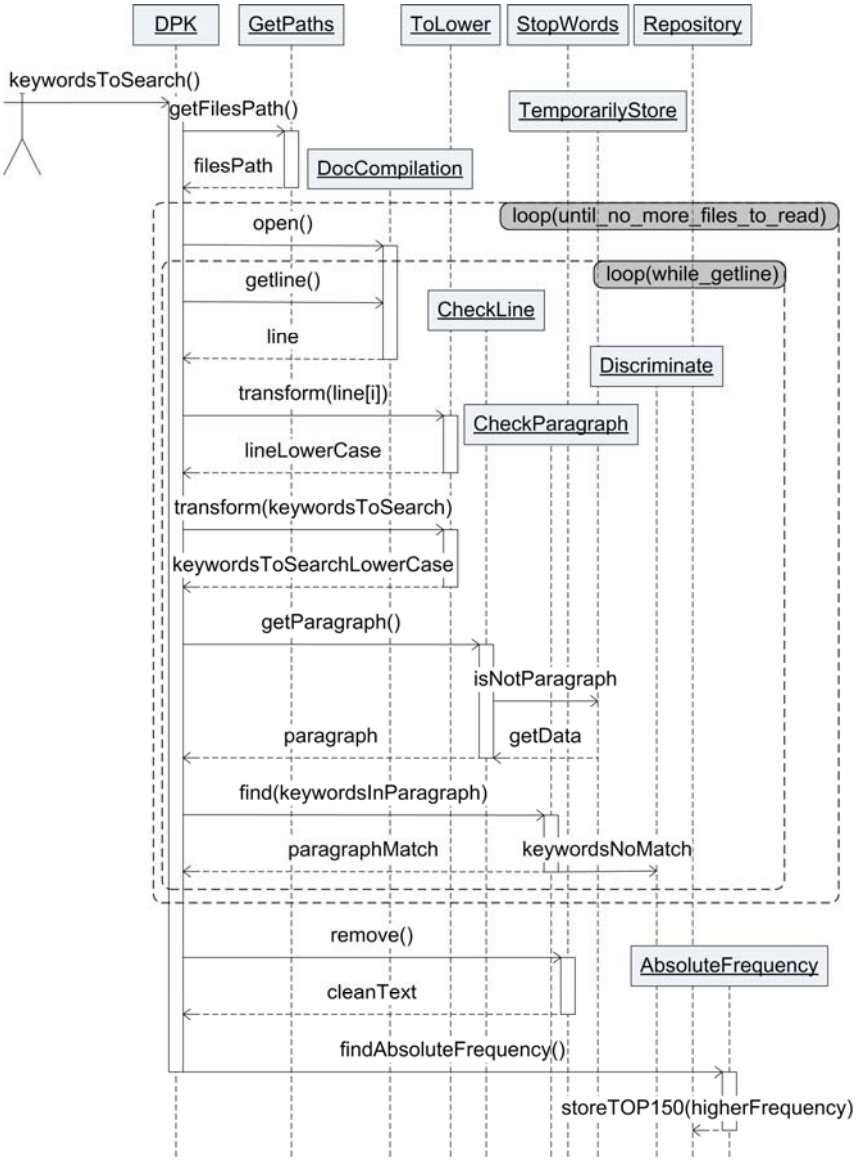


Diagrama de Secuencia 1. Método DPK

En el Diagrama de Secuencia 1 se exponen las principales operaciones del método `DPK`. Básicamente, el método almacena sólo los párrafos que cumplan con un criterio de búsqueda (aquellos párrafos que contengan exactamente los `keywordsToSearch()`). El usuario es el encargado de ingresar la palabra clave

por consola. Como resultado se obtienen las rutas de los ficheros que se analizarán. Lo siguiente es comprobar uno a uno los ficheros encontrados en la ruta especificada y parsearlos línea por línea almacenando en una variable temporal los párrafos. Un párrafo es un componente del texto que en su aspecto externo se inicia con una mayúscula y termina en un punto y aparte. Si la línea no representa un párrafo, se almacena temporalmente hasta encontrar un párrafo. Todas las palabras y caracteres son transformados a minúsculas para una mejor gestión de la información. Una vez que un párrafo entero se encuentra en la variable temporal, se comprueba si en alguna de sus líneas existe concordancia con los `keywordsToSearch()`; si así fuera, se pasa el resultado de los párrafos encontrados a la función que se encargará de eliminar los `stopWords`, devolviendo el texto limpio. El último proceso del método `DPK` propuesto es hallar la frecuencia absoluta de las 150 palabras más frecuentes y almacenar el resultado `palabra/frecuenciaAbsoluta` en un repositorio que luego será utilizado por el método `IRWDP`. Por otro lado, si no se cumple con el criterio de búsqueda, el párrafo entero es descartado.

3.3.1.4. Recuperación de la Información con Datos Ponderados en Párrafos (Método IRWDP)

Una vez realizado el método `DPK`, el siguiente paso de la metodología propuesta es la aplicación del método `IRWDP` (Information Retrieval with Weighted Data in Paragraphs), que se comporta de manera similar al sistema feedback por relevancia para obtener las frecuencias relativas ponderadas en base a una búsqueda específica. El Diagrama de Secuencia 2 muestra el método `IRWDP`. Al principio del procedimiento, entra en un bucle que es el encargado de leer el repositorio de `palabra/frequenciaAbsoluta` almacenado anteriormente con el método `DPK`. Otro bucle es el encargado de gestionar, como en el caso del método `DPK`, la ruta de las colecciones de documentos de la Tabla 1. Si se trabaja sobre la base de la palabra clave buscada anteriormente con el método `DPK`, Ej. “Surveillance”, éste deberá ser la palabra clave a buscar en el método `IRWDP`, que representa un contenedor de servicio `LoD4` de la taxonomía propuesta por `RITA`. El método `IRWDP` busca en el repositorio almacenado por el método `DPK` el top de 150 palabras con sus respectivas frecuencias absolutas. Por cada `palabra/frequenciaAbsoluta`, se realiza toda la operación de búsqueda en la colección de documentos. Nótese que aquí, la frecuencia absoluta de cada palabra representa el tamaño de muestra específico. Los siguientes procedimientos son muy similares a los procedimientos del método `DPK` para obtener la frecuencia absoluta. En este caso, la frecuencia absoluta de la palabra a buscar es la que se extrae del repositorio. El `keywordToSearch()` proporcionado por el usuario representa el parámetro de control que permitirá sólo examinar y trabajar con los párrafos específicos

descartando, como el caso de DPK, aquellos párrafos que no son relevantes o no representan parte del estudio.

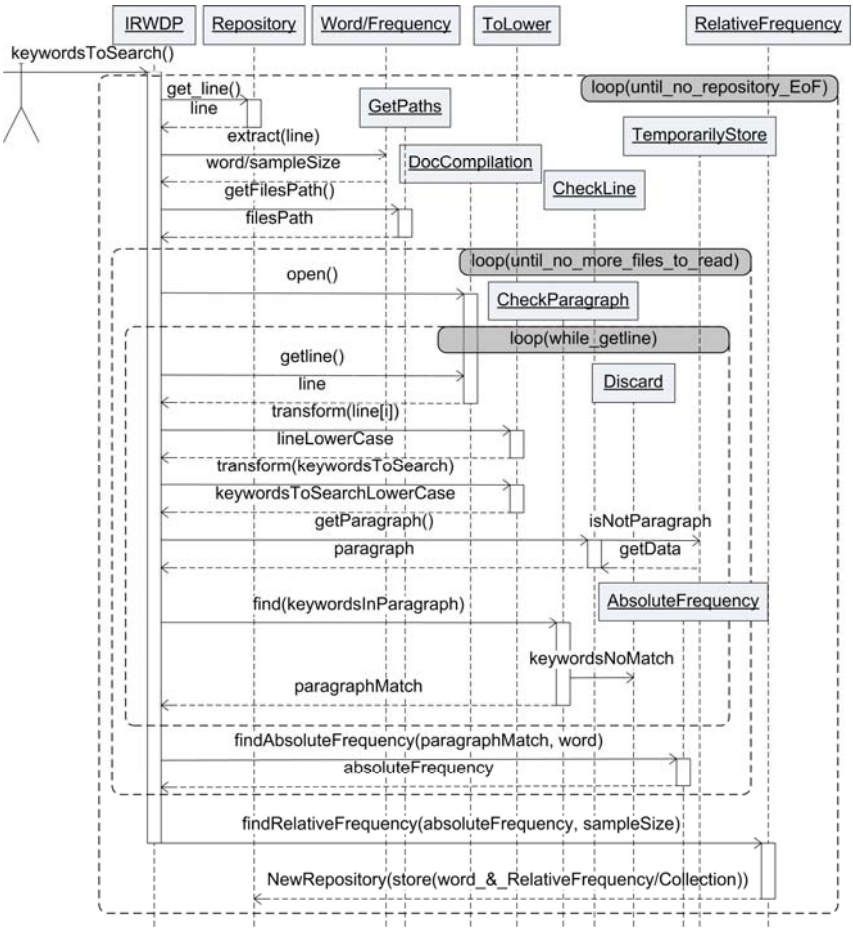


Diagrama de Secuencia 2. Método IRWDP

Una vez obtenida la frecuencia absoluta de la palabra en todas las colecciones de documentos, lo siguiente es realizar los cálculos para hallar la frecuencia relativa proporcionando la matriz de términos/colección de documentos.

La Tabla 2 muestra un extracto de las frecuencias relativas obtenidas para el caso de "*Surveillance*" como palabra clave a buscar.

Contenedor de Servicios

Surveillance	Colección de Revistas Científicas Consideradas										
palabras	A	B	C	D	E	F	G	H	I	J	
traffic	0.32	0.09	0.01	0.16	0.07	0.00	0.18	0.04	0.12	0.01	1
surveillance	0.36	0.06	0.08	0.10	0.23	0.01	0.07	0.02	0.06	0.02	2
time	0.41	0.05	0.02	0.03	0.18	0.00	0.06	0.00	0.25	0.00	3
data	0.45	0.07	0.00	0.06	0.24	0.00	0.08	0.00	0.10	0.00	4
system	0.31	0.06	0.01	0.02	0.37	0.00	0.14	0.02	0.05	0.00	5
vehicle	0.15	0.10	0.10	0.26	0.25	0.00	0.09	0.00	0.04	0.00	6
real	0.38	0.08	0.05	0.03	0.29	0.00	0.06	0.00	0.11	0.00	7
information	0.54	0.03	0.00	0.13	0.07	0.00	0.13	0.00	0.10	0.00	8
estimation	0.63	0.08	0.00	0.00	0.00	0.00	0.12	0.00	0.18	0.00	9
technology	0.17	0.07	0.07	0.05	0.62	0.00	0.02	0.00	0.00	0.00	10
⋮	…	…	…	…	…	…	…	…	…	…	…
palabra 150	…	…	…	…	…	…	…	…	…	…	150
	1	2	3	4	5	6	7	8	9	10	

Tabla 2. Matriz término/colección 150x10

3.3.1.5. Extracción del Conocimiento

La mayoría de los métodos IR dependen de búsquedas exactas entre palabras, consultas o documentos específicos. Estos métodos de IR (booleano, vectoriales estándar, probabilísticos), comúnmente tratan las palabras como si fueran independientes, aun cuando evidentemente no lo son. Si se quisiera obtener pares o triples de palabras esto no sería posible con técnicas básicas. Será necesario implementar métodos capaces de hallar las relaciones que puedan existir entre los términos.

En primer lugar se plantea la aplicación de un método de análisis semántico de espacio vectorial, donde los documentos que son base del estudio están representados por vectores de columnas en una matriz término/colección de documentos que llamaremos en adelante matriz término/colección. Para una matriz de términos/colección $m \times n$, $A = (a_{ij})$ el enésimo término, a_{ij}, representa la frecuencia ponderada del término i en la colección j. Una de las ventajas de

utilizar el método del espacio de vectores es que la alta dimensionalidad de la matriz término/colección puede ser reducida preservando la estructura del espacio vectorial original [122].

A continuación, se realiza una búsqueda de palabras clave utilizando teorías de co-ocurrencia sobre la matriz de términos/colección obtenida en el apartado anterior. El método de palabras co-ocurrentes considera que el contenido de un documento, o en el caso del planteamiento en esta tesis, colección de documentos, viene definido por sus descriptores o palabras clave, que están dadas por la taxonomía del paso 1. La lista de palabras a analizar puede ser muy extensa, del orden de varios miles, por lo que las dimensiones de una matriz de ocurrencias resultan muy complejas. Dos palabras estarán más ligadas o asociadas entre sí cuanto mayor sea la co-ocurrencia entre ellas. Por tanto, la medida del enlace entre pares o triples de palabras será proporcional a la co-ocurrencia de ellas en la colección que se tome como muestra.

3.3.1.6. Procesamiento de Lenguaje Natural (NLP)

Un método que puede automáticamente construir un espacio semántico es LSI (Latent Semantic Indexing) [123], [124], [125]. LSI, también conocido como LSA (Latent Semantic Analysis), es un método de recuperación e indexación que examina la similaridad de los contextos en el que aparecen las palabras, creando una dimensión reducida donde las características de las palabras más similares son las que están más cerca unas de otras. Esta técnica está basada en el principio de que las palabras o términos que son utilizados en el mismo contexto tienden a tener cierta similaridad. LSA es utilizado para predecir la coherencia textual, comprensión, desambiguación contextual de homonimias y generación del significado central inferido de un párrafo. La habilidad del método LSA de derivar simultáneamente representaciones de estos dos tipos de interrelaciones de significado depende de un tratamiento matemático. LSA asume que la elección de la dimensionalidad en el que todas las relaciones del contexto de palabras locales se representan simultáneamente puede ser de gran importancia, y que reduciendo la dimensionalidad de la matriz de los datos observados y del número de contextos iniciales a uno mucho menor se producirán a menudo mejores aproximaciones a las relaciones cognoscitivas humanas [126]. LSA/LSI utiliza un método basado en un teorema de álgebra lineal llamado SVD (Singular Value Decomposition) [127], [128] para reducir la matriz de datos e identificar patrones en el relacionamiento entre los términos y conceptos contenidos en una colección de texto no estructurada. No es necesario utilizar ningún diccionario externo, tesauro o bases de conocimientos para determinar estas asociaciones entre palabras ya que son derivadas de un análisis numérico de textos existentes. La asociación descubierta es específica al dominio de

interés. SVD dice que la matriz rectangular A se puede descomponer en el producto de tres matrices ortogonales, una matriz ortogonal U, una matriz diagonal S y la transpuesta de una matriz ortogonal V. El teorema está usualmente expresado por:

$$A_{mn} = U_{mm} S_{mn} V^T_{nn} \qquad \text{Ec. (10)}$$

Donde en $U^T U = I, V^T V = I$ las columnas de U son vectores propios ortonormales de AA^T, las columnas de V son vectores propios ortonormales de $A^T A$ y S es una matriz diagonal que contiene las raíces cuadradas de los valores propios de U o V en orden descendente.

LSA utilizará la matriz término/colección 150x10 de la Tabla 2 para construir el espacio semántico. En la matriz, cada fila corresponde a una única palabra en el corpus de las publicaciones y cada columna representa la colección de documentos relevantes de la Tabla 1.

3.3.1.7. Análisis Estadístico de los Datos

Paso 1: Aplicación del Método Cluster Jerárquico Aglomerativo

A partir de la reducción de dimensionalidad se pueden extraer caracterizaciones que permitirán predecir o deducir relaciones útiles mediante la utilización de técnicas de clustering (agrupación). Los clusters agruparán una serie de vectores de acuerdo al criterio de cercanía, definida en términos de una determinada función de distancia. Generalmente, los vectores de un mismo cluster comparten propiedades comunes. Conociendo estos grupos podremos describir y construir servicios dentro del conjunto de datos multidimensional y expresarlos como dendrogramas o árboles ultramétricos, donde mediante una jerarquía indexada los pares y triples de palabras podrán ser visualizadas mediante un gráfico sencillo e intuitivo. El clustering jerárquico construye una jerarquía de clusters tipo `top-down` (divisivo) o `bottom-up` (aglomerativo), ya sea a partir de un cluster y recursivamente dividiendo los clusters más apropiados en relación con alguna similitud métrica, o a partir de un punto en el cluster y recursivamente ir fusionándolos con clusters similares. La división/fusión continúa hasta que se cumpla un criterio `stopping` (Ej. número de clusters) [129]. El criterio fundamental en las técnicas HACM (Hierarchical Agglomerative Clustering Methods) es la identificación de clústeres que deben ser enlazados o combinados. Los métodos de enlaces utilizados comúnmente son: `single-linkage`, `complete-linkage`, `average-linkage` [130] y el método de `ward` [131]. Este último es uno de los más populares junto con el método `average-linkage` [132]. Existen varios algoritmos de aglomeración como los nombrados anteriormente, que sólo difieren en la definición de disimilitud entre clusters. En esta tesis y para el desarrollo de la

metodología, se utilizará el algoritmo `average-linkage` por ser uno de los más robustos [132] al formar clusters de mejor calidad [129] y de mayor rendimiento [130]. Este método es originario de Sokal y Michener [133] donde la distancia entre dos clusters esta definida por:

$$D_{KL} = \frac{1}{N_K N_L} \sum_{i \in C_K} \sum_{j \in C_L} d(x_i, x_j)$$

Si $d(x, y) = \| x - y \|^2$, entonces

$$D_{KL} = \| \bar{x}_K - \bar{x}_L \|^2 + \frac{W_K}{N_K} + \frac{W_L}{N_L}$$

La formula combinatoria sería,

$$D_{JM} = \frac{N_K D_{JK} + N_L D_{JL}}{N_M} \qquad \text{Ec. (11)}$$

En el `average-linkage`, la distancia entre dos clusters es la distancia promedio entre los pares de observación, uno en cada cluster. El `average-linkage` comúnmente une clusters con pequeñas variaciones y tiende ligeramente a producir clusters con la misma varianza. El cluster jerárquico puede ser representado gráficamente por un diagrama tipo árbol o por un dendrograma, que corresponde a la jerarquía de particiones o la clasificación terminológica en sí.

Paso 2: Aplicación del Método UPGMA y Construcción del Árbol Ultramétrico para extracción de Pares/Triples de Palabras

Uno de los retos computacionales de esta tesis es la obtención de una estructura de datos que nos ayude a la representación final de una ontología. Una solución propuesta por Gibas y Jambeck [134] fue la implementación de árboles filogenéticos. En las ciencias computacionales, existe una estructura de datos que posee las propiedades de los árboles filogenéticos llamada árbol ultramétrico. Para describir estas propiedades en mayor detalle, es necesaria la definición formal de métrica.

Definición 1.

Sea A un conjunto de taxones (unidades en una clasificación jerárquica agrupada). Sea $d : A \times A \rightarrow Q^{\geq 0}$ una función. Entonces, d es la métrica en A si satisface las siguientes propiedades:

(i) Para todo $a,b \in A, d(a,b) = 0$ se cumple si y sólo si $a = b$

(ii) Pata todo $a,b \in A, d(a,b) = d(b,a)$ (simetría)

(iii) Para todo $a,b,c \in A, d(a,b) \leq d(a,c) + d(c,b)$ (desigualdad triangular)

Otra definición con una mayor restricción en la medida de distancia se detalla a continuación.

Definición 2.

Sea A un conjunto de taxones. Sea $d : A \times A \to Q^{\geq 0}$ una métrica en A. Entonces d es ultramétrica si además satisface la siguiente condición de tres puntos:

Para todo $a,b,c \in A$, dos de las distancias $d(a,b), d(a,c), d(b,c)$ son iguales y no más pequeños que el tercero. Esta condición dice que, por la elección de tres taxones arbitrarios $a,b,c \in A$, se cumple una de las siguientes tres condiciones

$$d(a,b) \leq d(a,c) = d(b,c),$$
$$d(a,c) \leq d(a,b) = d(b,c),$$
$$d(b,c) \leq d(a,b) = d(a,c)$$

Esto significa que la condición de tres puntos es una restricción más fuerte que la de desigualdad triangular.

Una vez conocida en detalle la definición de métrica, se explicará la definición formal de árbol ultramétrico,

Definición 3.

Sea $A = \{a_1, a_2, ..., a_n\}$ un conjunto de taxones. Un árbol ponderado $T = (V, E, d)$ con raíz r y una función para la ponderación de aristas $d : E \to Q^{\geq 0}$ es un árbol ultramétrico para el conjunto A, si satisface las siguientes condiciones:

(i) T es un árbol binario, Ej. cada vértice interior de T posee exactamente dos sucesores.

(ii) T tiene exactamente n hojas, etiquetadas cada una con los taxones de A.

(iii) La suma de los pesos de las aristas de cualquier camino desde la raíz hasta cualquier hoja es la misma.

La distancia entre dos vértices arbitrarios x e y de T es la suma de los pesos de las aristas en el camino desde x hasta y; esta distancia es denotada por la

$distT(x, y)$ [135]. Dada una matriz de n taxones (sujetos u objetos), se sugieren dos métodos simples para construir árboles ultramétricos. El primero de ellos es `UPGMA` (Unweighted Pair Group Method with Arithmetic) y el segundo es `WPGMA` (Weighted Pair Group Method with Arithmetic), ambos métodos jerárquicos aglomerativos (`average-linkage`) [136]. El `UPGMA` es muy utilizado en bioinformática para desarrollar taxonomías con datos numéricos obtenidos de un conjunto de taxones [137]. Este método construye el árbol filogenético tipo bottom-up desde sus hojas (conjunto de taxones). Considerando un árbol ultramétrico T, si un subconjunto de taxones S forma un subárbol de T, lo llamamos cluster, donde cada taxón forma un cluster por sí mismo. Para un nodo u, se define que $height(u)$ será la longitud del camino desde u hasta cualquiera de las hojas descendientes (Como el árbol T es ultramétrico, todos los caminos deberían tener la misma longitud). Entonces, sean i y j las hojas descendientes de u en dos diferentes subárboles. Para asegurarnos de que la distancia desde la raíz hasta ambos descendientes i y j son iguales, se debe cumplir que $height(u) = M_{ij}/2$, donde M representa la matriz de distancia métrica desde i hasta j.

Para cualquier par de clusters $C1$ y $C2$ del árbol T, se define,

$$dist(C_1, C_2) = \frac{\sum i \in C_1 j \in C_2 M_{ij}}{|C_1| \times |C_2|}$$

Nótese que la $dist(C_1, C_2) = M_{ij}$ para todos los $i \in C_1$ y $j \in C_2$. Por ejemplo, sea u el nodo antecesor común más bajo de i y j, la $dist(C_1, C_2) = 2\ height(u)$. Entonces tenemos que para cualquier nodo C_x cuyo antecesor no es u, se entiende que,

$$dist(C_1 \cup C_2, C_x) = \frac{dist(C_1, C_x) + dist(C_2, C_x)}{2} \qquad \text{Ec. (12)}$$

De acuerdo con este análisis, para un conjunto C de clusters, C_i y C_j son dos clusters que pertenecen al conjunto C. Si C_k es un árbol formado por la unión de C_i y C_j con una raíz, C_k es un cluster (subárbol) del árbol ultramétrico T. En el método `UPGMA`, las distancias son calculadas con un promedio aritmético que depende del número de elementos en cada cluster. Otro método, similar al `UPGMA`, es el método `WPGMA` [136]. Básicamente ambos métodos trabajan de la misma forma, con la diferencia en la función de distancia utilizada en el último paso. En `WPGMA`, en lugar de buscar encontrar simplemente el promedio, se busca encontrar el

promedio ponderado, lo que asegura que cada taxón interviene de forma igualitaria en el resultado final. En WPGMA, la función utilizada para calcular la distancia desde el cluster recientemente creado a todos los demás clusters es,

$$dist(C_{(i\cup j)}, C_x) = dist(C_k, C_x) = \\ = \frac{n_i \times dist(C_i, C_x) + n_j \times dist(C_j, C_x)}{n_i + n_j}$$

Donde C_i y C_j representan los clusters que serán fusionados y C_k es el cluster tal que $C_k = C_i \cup C_j$. C_x es un cluster en C pero no en C_k, mientras que n_i y n_j representan los respectivos tamaños de C_i y C_j. Debajo, el lema muestra que esta función de distancia es una definición acorde con la definición original de distancia entre clusters y que cada taxón contribuye igualitariamente al resultado final.

$$dist(C_i, C_j) = \frac{1}{n_i \times n_j} \sum_{c_i \in C_i} \sum_{c_j \in C_j} dist(c_i, c_j)$$

Esto puede ser demostrado con la siguiente inducción matemática,

$$dist(C_{i\cup j}, C_x) = \frac{n_i dist(C_i, C_x) + n_j dist(C_j, C_x)}{n_i + n_j}$$

Entonces, por definición tenemos que,

$$dist(C_{i\cup j}, C_x) = \frac{n_i}{n_i n_x} \sum_{c_i \in C_i} \sum_{c_x \in C_x} dist(c_i, c_x) + \\ \frac{n_j}{n_j n_x} \sum_{c_j \in C_j} \sum_{c_x \in C_x} dist(c_j, c_x) \Big/ n_i + n_j$$

$$= \frac{\frac{1}{n_x}\left(\sum_{c_i \in C_i} \sum_{c_x \in C_x} dist(c_i, c_x) + \sum_{c_j \in C_j} \sum_{c_x \in C_x} dist(c_j, c_x) \right)}{n_i + n_j}$$

$$= \frac{\frac{1}{n_x} \sum_{c_k \in C_i \cup C_j} \sum_{c_x \in C_x} dist(c_k, c_x)}{n_i + n_j}$$

Sea $C_i \cup C_j = C_k$, lo cual significa que $n_i + n_j = n_k$ entonces,

$$dist(C_k, C_x) = \frac{\sum_{c_k \in C_k} \sum_{c_x \in C_x} dist(c_k, c_x)}{n_k n_x} \qquad \text{Ec. (13)}$$

De acuerdo con este lema, se puede concluir que con la función de distancia utilizada por WPGMA, cada taxón contribuye igualmente al resultado final. WPGMA y UPGMA se diferencian en el resultado final y no en el mecanismo matemático para lograrlo. Para el desarrollo de la metodología propuesta en esta tesis, se utilizará el UPGMA por ser un método más sencillo, rápido y que ha sido ampliamente utilizado en la literatura.

3.3.2. Contribuciones de la metodología propuesta

Los métodos propuestos en éste apartado permitirán dotar a la gran cantidad de información de una estructura lógica, basada en soluciones de gestión del conocimiento a partir de la IR y la SLR. Se presentaron mecanismos autónomos de meta-análisis de datos y en base a éstos se confeccionó la ontología. Aunque muchas de las técnicas presentadas ya existen individualmente en la literatura, su uso conjunto para la elaboración de una ontología constituye una novedad de este trabajo. También tienen carácter novedoso los métodos DPK e IRWDP propuestos, con los que se logró disminuir sustancialmente la cantidad de párrafos discriminando aquellos que no representan parte importante del estudio en cuestión. Esta discriminación hace que la extracción de datos relevantes sea más exacta.

3.4. Propuesta de un Servicio Semántico Cooperativo en el Ámbito de los ITS

Aunque el SS (Semantic Service) propuesto no está limitado a trabajar con un dominio ontológico en particular, en esta tesis trabajará con la ontología desarrollada en el apartado anterior ITS.rdfs, que incluye las clases, superclases, subclases, instancias y propiedades más importantes que intervienen en el dominio seleccionado. Esta posee toda la información relacionada con los servicios implementados referentes a la gestión de servicios en los ITS. Si un dispositivo

desea hacer uso de alguno de los servicios, (ej. El Servicio de Vigilancia en el Tráfico o El Servicio de Prioridad en el Control del Tráfico, Servicio de Gestión de Semáforos, Servicio de Avisos en Paneles, etc.), el SS ofrecerá la manera de acceder a ellos a través de los IOR, la interfaz que busca, más toda la información relativa, de manera descriptiva, lógica e inteligente. Se pretende que el SS evite la información redundante y excesiva entre dispositivos y ahorre tiempo de latencia innecesaria. Los sujetos servirán para tomar decisiones basándose en las propiedades de la estructura. En consecuencia, la estructura del sujeto y las relaciones entre éstas representan el esqueleto ontológico de los servicios ofertados.

3.4.1. Descripción formal del Servicio Semántico

El SS es un servicio de comunicación semántico desarrollado en TAO de CORBA y encargado de dar soporte de comunicación distribuida a los exportadores/importadores de información. CORBA no ofrece lo mismo que SOA promete, pero se presenta como el mejor subconjunto de herramientas para apoyar lo que SOA "intenta ser". Junto con esta arquitectura distribuida se propone la utilización de un set de librerías base. En un principio, se realizarán exhaustivas pruebas sobre una ontología ITS.rdfs de menor tamaño, ampliando gradualmente el tamaño de la muestra hasta 12kT para poder simular de esta forma dificultades críticas de comunicación e intercambio de datos. Redland será utilizado para interactuar con la ontología que previamente será portada a RDFS. Se utilizará un analizador sintáctico Raptor para analizar secuencias de símbolos a fin de determinar la estructura gramatical y un lenguaje de sintaxis tipo consulta Rasqal que se utilizará para construir y ejecutar consultas.

Se propone un framework inteligente que permita dar una precisión semántica en la búsqueda de servicios permitiendo la adaptación y combinación de la información sobre un sistema distribuido. Antes que nada, es importante hacer hincapié sobre algunos detalles de diseño. Empecemos por distinguir las diferencias entre declaración e información. Las declaraciones representan un set de triples $D \subseteq S \times P \times O$, donde S representa un set de sujetos, P un set de predicados y O un set de objetos. Entonces,

Definición 1 (Declaración) Las declaraciones D representan un set finito de tripletas donde,

$$D = \{t1, t2, ..., tn\}$$

Por información se refiere a la propiedad ϕ que puede ser derivada del set de declaraciones con respecto a lógica formal L con una relación vinculada $\vdash L$.

Definición 2 (Información) Las declaraciones D poseen la información ϕ si y sólo si ϕ se encuentra implícito en el conjunto finito de triples con respecto a la relación de implicación, entonces,

$$D \models_L \phi$$

Proposición 1. Para un set D de declaraciones `RDF` y una lógica formal del tipo $\mathcal{L}$ = `RDF/RDFS`, cualquier triple $t \in D$ puede ser considerado como una propiedad del set de declaraciones D donde,

$$t \in D \Rightarrow D \models_{RDF} t$$

Cada parte de las declaraciones D (excepto los literales) pueden ser identificadas por una `URI` permitiendo que las declaraciones puedan ser escritas sobre cualquier recurso con `URI`. Los predicados también son definidos de forma descriptiva por una `URI`. Aunque una colección de declaraciones representa la descripción formal, también puede ser representada como un gráfico de nodos (sujetos) y arcos (predicados) que apuntan a otros nodos (objetos) o literales [138], Figura 10.

Para trabajar con la información semántica se utilizará un set de librerías (free software / open source) `RDF Redland` [138], herramienta diseñada para cubrir las cuatro capas inferiores del bloque de construcción semántico, Figura 15.

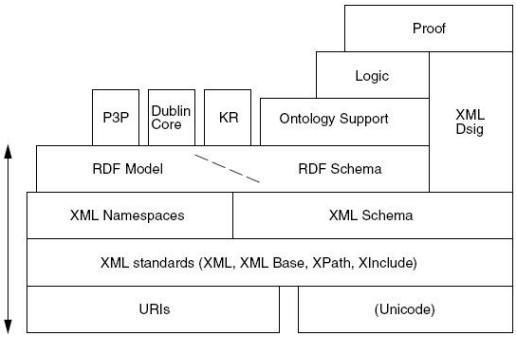


Figura 15. Bloque de construcción semántico Redland RDF

Redland proporciona el soporte necesario para soportar clases tipo cadenas de secuencias, declaraciones, modelos, parseadores, consultas, almacenamientos y serializadores Figura 16. En el diagrama de clases de Redland, las clases son utilizadas y asociadas unas con otras. Las Support Classes son utilizadas en todo el resto de las clases, según sea necesario. Las clases Stream son utilizadas cada vez que una declaración de secuencia es aceptada o generada por el Model, Storage o por la clase Parser. La clase Model utiliza la clase Stream para realizar la serialización/de-serialización del Model y retorna una lista de declaraciones por consultas. La clase Parser es utilizada solamente para proporcionar una secuencia de declaraciones como resultado del análisis. A un nivel más simple, cada objeto Model tiene un mapping one-to-one al objeto Storage que representa. La funcionalidad de agregar modelos se encuentra presente en la misma clase Model. Así, este alto nivel de modelado puede tener sub-modelos y en ese caso el mapping one-to-one a Storage no es aplicado. Este alto nivel de modelado podría tener un conjunto de sub-modelos u otro tipo de relación [139].

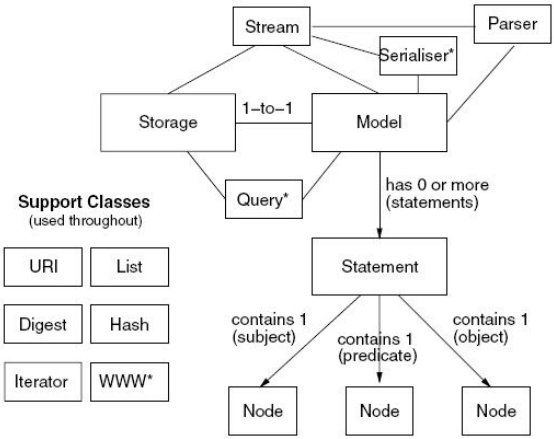


Figura 16. Diagrama de clases de Redland

Una de las ventajas principales en la utilización de la API de Redland es que está implementada en C y puede ser portada fácilmente a C++ o ser compilada para sistemas embebidos, que es nuestro caso. Existen otras APIs desarrolladas para otros tipos de lenguajes como la API de OWL-Protegé o Jena, implementados en Java. En el momento de decidir qué tipo de herramientas utilizar para trabajar con sistemas embebidos y dada la utilidad de portarlo a una arquitectura como CORBA, se opta por Redland como un flexible y eficiente complemento. Redland provee interfaces de alto nivel que permiten tener instancias, modelos, almacenamientos en diferentes estructuras, capacidad de consultar y manipular en diferentes lenguajes como C, Perl, Python, etc. muy útil en el momento de desarrollar sistemas

distribuidos complementados con CORBA. Otra de las ventajas visibles en la utilización de esta librería es que utiliza una API orientada a objetos, proporcionando numerosas implementaciones de clases como módulos que pueden ser agregados, eliminados o reemplazados para permitir diferentes funcionalidades y optimizaciones específicas en las aplicaciones. El Framework provee además un núcleo para desarrollar nuevas aplicaciones RDF experimentando con varias técnicas de implementación [139].

Por otra parte, las declaraciones D necesitan ser almacenadas de alguna manera eficiente. El almacenamiento necesita utilizar sistemas existentes como base de datos relacionales u otro tipo de mecanismos que soporten la escalabilidad de la información. Se espera que la información pueda ser manipulada y esté disponible en tiempo de ejecución en entornos distribuidos. Para las pruebas de rendimiento, se prevé utilizar almacenamientos persistentes sobre un motor de base de datos (free software / open source) embebido de propósito general, BDB (BerkeleyDB db-5.1.19). BDB está diseñado para trabajar con aplicaciones de alto rendimiento y en misiones críticas. Soporta también varios lenguajes como C, C++, Java, Perl, Python, PHP, Tcl, etc. Esta base de datos almacena cada valor de dato con una clave asociada, aunque también es posible una clave para múltiples valores de mapeo. Ha sido implementado en muchas plataformas como UNIX, Linux, Windows y otros sistemas operativos de tiempo real [140].

Finalmente, se utilizará una librería (free software / open source publicado bajo licencias LGPL (GPL)) Raptor RDF Syntax Library [141] que provee un set de parseadores y serializadores para generar tripletas RDF parseando sintaxis o serializando las tripletas dentro de la sintaxis. Soporta parseadores de sintaxis RDF/XML, N-Quads, N-Triples, TRiG, Turtle, etiquetas RDD incluidas todas las versiones de RSS, Atom 1.0 y 0.3, GRDDL y microformatos para HTML, XHTML/XML y RDFa. El serializador de sintaxis soporta RDF/XML (regular y abreviado), Atom 1.0, GraphViz, JSON, N-Quads, N-Triples, RSS 1.0 y XMP. Esta herramienta ha sido diseñada para trabajar con Redland pero se presenta como independiente. Es una librería portable que trabaja a través de múltiples sistemas POSIX (Unix, GNU/Linux, BSDs, OSX, cygwin, win32). Otra librería que se utilizará (free software / open source) es Rasqal RDF Query Library [142], encargado de la construcción y ejecución de consultas. Esta herramienta será implementada del lado del cliente retornando un enlace booleano, sintaxis o gráficos de tripletas RDF. Soporta lenguajes de consultas como SPARQL 1.0, RDQL, Draft SPARQL Query 1.1, Update 1.1 Syntax y extensiones experimentales de SPARQL (LAQRS). Rasqal es capaz escribir resultados de consultas de enlace en SPARQL XML, SPARQL JSON, CSV, TSV, HTML, tablas ASCII, RDF/XML y Turtle/N3 y leerlos en SPARQL XML, RDF/XML y Turtle/N3. Esta herramienta ha sido diseñada para trabajar con Redland y Raptor pero también se presenta como una librería independiente.

3.4.1.1. Arquitectura Distribuida del Servicio Semántico

La clave de la interoperabilidad y reusabilidad de las ontologías por parte del SS radicará en la posibilidad de componer de forma automática la información en base a búsquedas inteligentes. Se propone entonces un SS que se encargará de parsear y almacenar esquemas ontológicos en diferentes proveedores de servicios de almacenamiento como ficheros, BerkeleyDB, PostgreSQL, gobernando y coordinando el trabajo cooperativo y el flujo de la información entre dispositivos C/S. El SS será el encargado de gestionar la interoperabilidad de metadatos. Los exportadores (servidores) e importadores (clientes) que cuenten con metadatos relevantes podrán interactuar cooperativamente consultando otros dispositivos en tiempo de ejecución. El exportador que desee exportar su información al esquema principal gestionado por el SS debe pertenecer al dominio de datos de la ontología principal, en caso contrario será rechazado.

En el Código 4 se puede apreciar el IDL del SemanticManager, parte medular del SS para establecer las comunicaciones C/S. Se define el módulo Semantic que aísla el espacio entre nombres y actúa como contenedor para las interfaces y declaraciones.

```
module Semantic
{
    interface Manager
    {
        exception KbNotFound {};

        exception KbNotMatch {};

        void AddToSchema (in string NewInfo)
        raises (KbNotFound);

        void ClientQuery (inout string QueryDetail)
        raises (KbNotFound, KbNotMatch);

        void destroy ();
    };
}; /* end module Semantic */
```

Código 4. IDL del Semantic Manager

La interfase Manager declara la API que utilizarán los objetos para comunicarse. Tanto el importador como el exportador que deseen comunicarse con el SS podrán realizar llamadas a los métodos remotos especificados en esta interfase. Se establecen también las excepciones que los importadores y

exportadores recibirán en caso de que la Base del Conocimiento no sea encontrada o no concuerde con la información buscada.

En el Diagrama de Secuencia 3 se exponen los métodos de inicialización del `ORB.init()`. En un principio, el enlace del `SS` se establece con el Servicio de Nombres de `CORBA`.

```
-ORBInitRef NameService=corbaloc:iiop:172.16.74.3:12345
```

De esta forma se hace accesible para los importadores/exportadores con el nombre `SemanticService` haciendo la llamada al `NameService` sobre el protocolo `IIOP`.

```
-ORBInitRef SemanticService=corbaloc:iiop:172.16.74.3:12345
```

Esta opción puede omitirse deshabilitando al Servicio de Nombres por línea de comando "`-x`". El enlace entonces debería realizarse directamente sobre el `IOR` del `SS`. El almacenamiento por defecto del esquema principal que gestionará el `SS` será en fichero, pero podría especificarse que se realizará sobre `BerkeleyDB` con la opción "`-s`" y los parámetros "`bdb`" o "`BDB`". Otra opción importante de destacar es que si en la ejecución del `SS` no se especifica la `URI` del esquema a gestionar, no se puede continuar. Es obligatoria la inclusión de la `URI` del esquema principal que gobernará el `SS`.

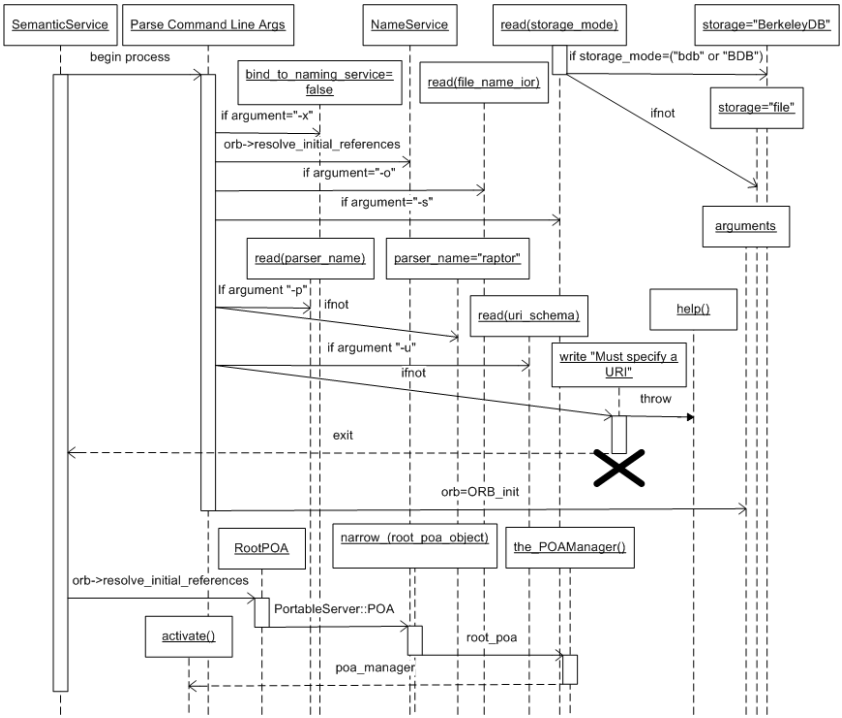


Diagrama de Secuencia 3. Inicialización del ORB y parsear argumentos

Otra de las opciones, pero no obligatorias, son las opciones `file_name_pid` y `parser_name`. En el primero se establece en un fichero el `pid` (Identificador del Proceso) del `SS`, que podrá ser utilizado por otros agentes para localizar al `SS` en el sistema en situaciones de control, monitoreo, o para enviar señales `SIGHUP` para terminar el proceso. La opción `parser_name` se establece para cambiar el nombre del parseador semántico que, en este caso, por defecto es `raptor`. Lo siguiente al comprobar los argumentos es la inicialización y la activación del `POAManager`. El `POAManager` será el responsable de controlar el estado de los procesos del propio `POA`. En el Diagrama de Secuencia 4 se muestra como se inicializa el propio `SS` y se enlaza al Servicio de Nombres. El esquema principal pasado en forma de `URI` como argumento por consola es procesado y parseado en un nuevo modelo. Aquí se establece la copia del esquema principal sobre el cual trabajará el `SS`. Una vez procesado el/los esquemas se ejecuta el `ORB`. Esta operación se bloquea a la espera de peticiones tanto del lado importador como del exportador o hasta que se llame un proceso `shutdown`.

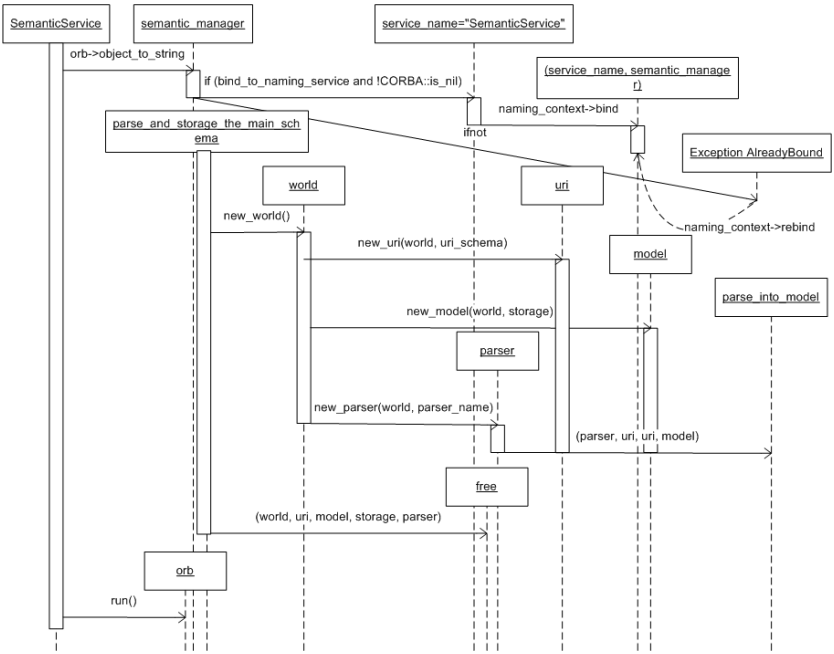


Diagrama de Secuencia 4. Ejecución del Servicio Semántico

El siguiente proceso es el que corresponde a la comunicación del exportador con el ss. En el Diagrama de Secuencia 5 se muestra como el ss trabaja con las peticiones de inclusión de nuevas informaciones de exportadores que quieren publicarse.

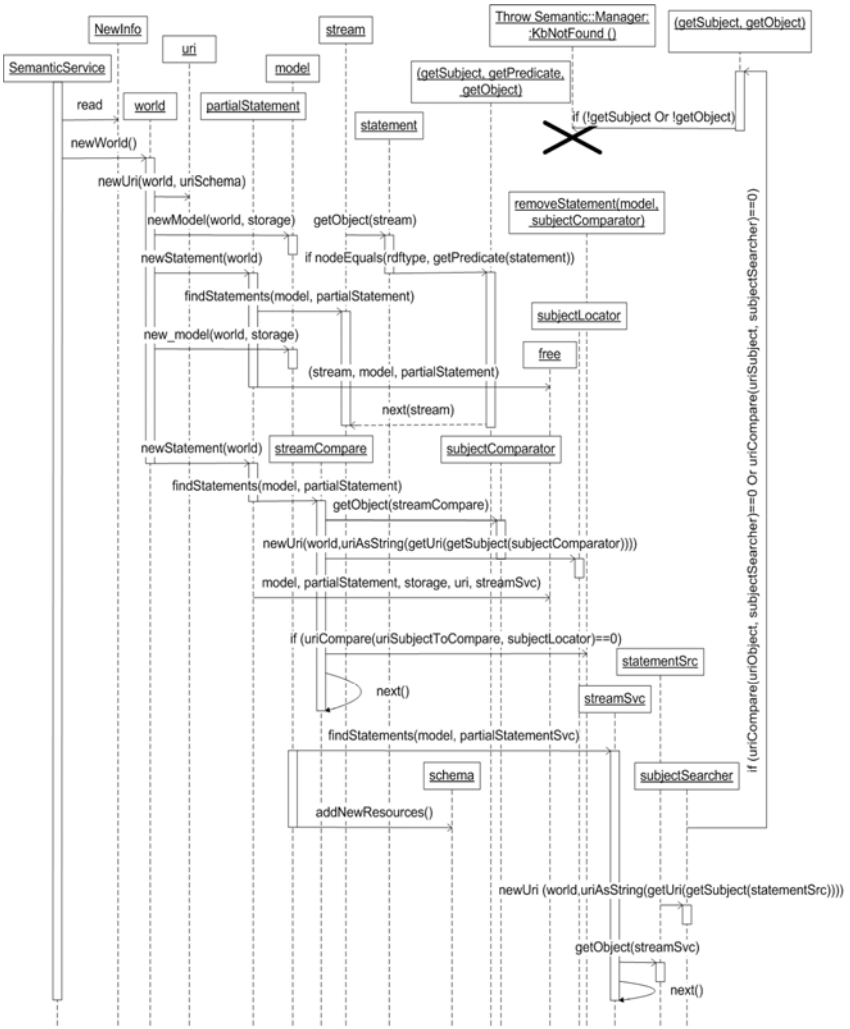


Diagrama de Secuencia 5. Agregar nuevas declaraciones al esquema principal

Se almacena el `string` recibido del exportador en un `RDF` temporal tipo fichero que luego será eliminado por el mismo `SS`. El `SS` lee y parsea la información recibida buscando la propiedad `<rdf:type>` instancia de `rdf:Property`, que es utilizada para indicar que el recurso es una instancia de una clase. Se utiliza para asegurarse de que al menos uno de los recursos que posee el `string` recibido desde el exportador es realmente una instancia de al menos una de las clases del esquema que el `SS` gobierna. Dicho de otra manera, indica que el sujeto es una instancia de la clase. Si al parsear la información recibida no se encontrara esta etiqueta, el proceso

de inclusión no podría realizarse. En este sentido, si se encontraran nodos redundantes o duplicados, estas serán reemplazadas para evitar la replicación de datos. Luego se aplica una función disyuntiva entre los nodos sujeto y objeto del modelo recibido y el esquema gobernado. Si una de las afirmaciones resultara verdadera la expresión `P V Q` será verdadera. Si resultara verdadera en alguno de los nodos, el modelo recibido puede ser almacenado junto a la base del conocimiento del `SS`. Si resultara falso, esto significaría que el exportador que intenta exportar su información no pertenece al dominio sobre el cual el `SS` trabaja. En este caso, una excepción `Semantic::Manager::KbNotFound();` es enviada al exportador. El importador por otra parte hace una petición en busca de un servicio en particular al `SS`. En el Diagrama de Secuencia 6 se describe como el `SS` procesa las consultas recibidas sobre peticiones del importador.

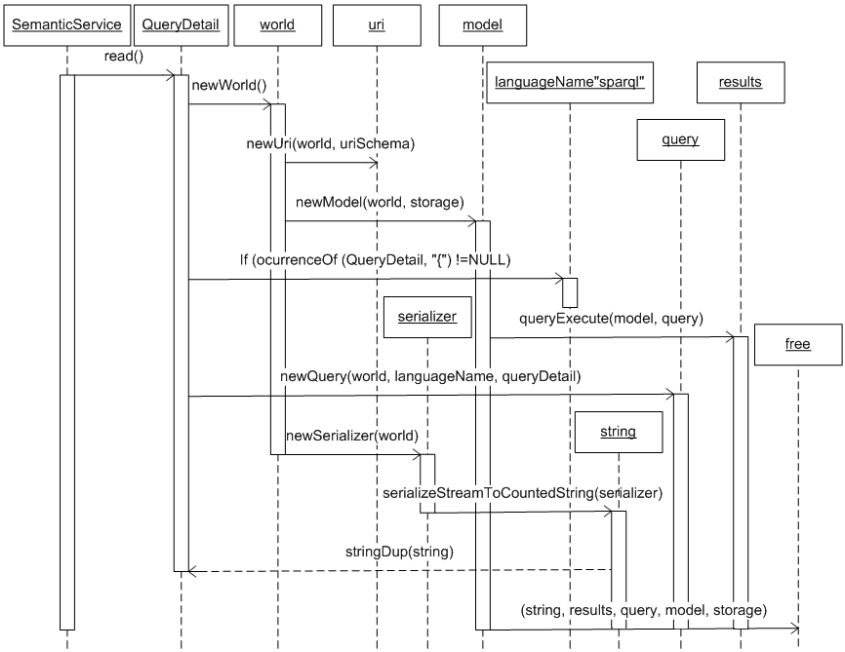


Diagrama de Secuencia 6. Realizar consulta sobre un servicio al SS

El importador es capaz de realizar consultas `RDQL` o `SPARQL` al `SS`. El `SS` lee el `string` recibido, construye la consulta y comprueba el tipo de lenguaje. Una vez definido, el `SS` comprueba y parsea todos los nodos del esquema principal en busca de resultados. Este será capaz de entregar al importador, enlaces, declaraciones booleanas, gráficos `RDF`, triples o `string` en forma de sintaxis si el importador así lo prefiere. El `SS` utiliza `RASQAL` para analizar y consultar el esquema almacenado. La respuesta enviada al importador (en la mayoría de los casos tipo `string`),

contendrá toda la información necesaria para que pueda hacer las peticiones al exportador. Dentro del `string` enviado se incluirá el `IOR` del exportador, ya que en principio es una premisa necesaria para establecer la comunicación con el exportador y se incluye dentro de la consulta del importador al `SS`. Si la consulta recibida no encuentra ninguna concordancia con el esquema principal, el `SS` envía al importador una excepción `Semantic::Manager::KbNotMatch();`. Por otro lado, si la consulta hace peticiones sobre una base del conocimiento con el cual el `SS` no trabaja, recibirá una excepción `Semantic::Manager::KbNotFound();`. En el Diagrama de Secuencia 7 se muestra la secuencia completa de la arquitectura propuesta.

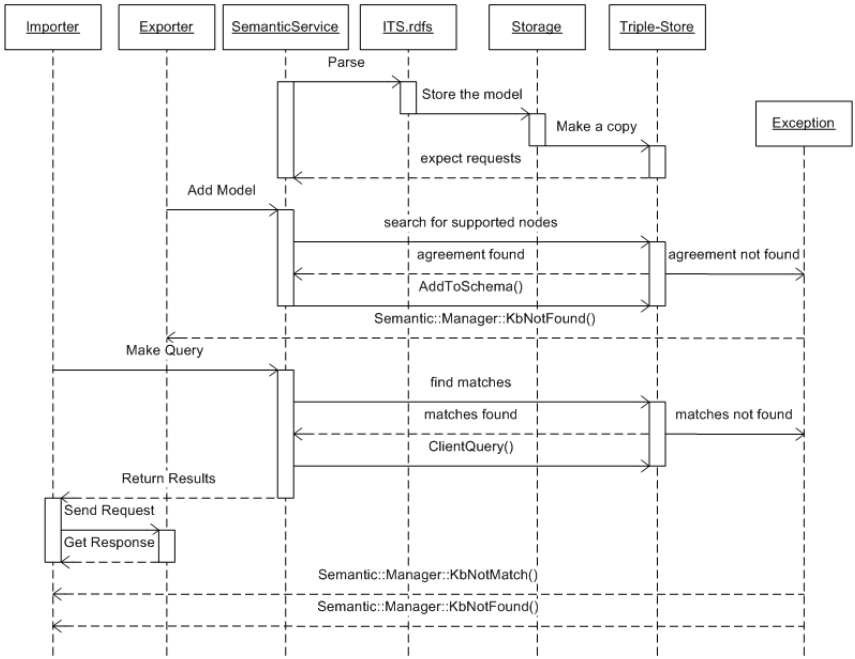


Diagrama de Secuencia 7. Arquitectura Propuesta

Capítulo 4 - Resultados Obtenidos

4.1. Descripción de la plataforma de prueba

Los equipamientos de tráfico se basan generalmente en las plataformas informáticas embebidas que se ejecutan en sistemas operativos embebidos [143], [144]. Entre las diferentes posibilidades de sistemas operativos disponibles, Linux está firmemente en el primer lugar como el sistema operativo de elección para equipamientos inteligentes y sistemas embebidos [30], [145]. Varios equipos ITS comerciales han sido utilizados para implementar una capa de middleware capaz de ofrecer servicios a los ciudadanos y los centros de control de tráfico (www.visioway.com). Estos equipos son utilizados para la estimación de parámetros de tráfico, mediante visión artificial [146], [147]. La Figura 17 es la imagen de una placa base. Incluye un vídeo procesador Freescale i.MX21 basado en un núcleo ARM926EJ-S, tarjetas de memoria y los controladores USB, sensor de interfaces CMOS y una interfaz Ethernet.

Figura 17. Placa base de un equipamiento ITS de visión artificial

El procesador ejecuta un sistema operativo ARM Linux (kernel 2.4.20) que proporciona acceso a una amplia variedad de módulos de software de código abierto, así como el soporte de una gran comunidad de desarrolladores. Se realiza la compilación cruzada de MICO y TAO utilizando arm-linux gcc-3.3.2. El tamaño de los ficheros de la librería resultante, después de compilación cruzada, está por debajo de 7,2 MB. Aunque esto podría ser de gran tamaño para ciertos sistemas embebidos, este no es el caso de los equipos ITS relacionados con visión artificial, ya que necesitan una gran cantidad de memoria para hacer frente a este tipo de aplicaciones.

La idea de un entorno urbano inteligente se ilustra en la Figura 18, donde se muestran varios sistemas embebidos interconectados a través de una red Ethernet que emularían a varios equipos ITS repartidos por la ciudad. Cada uno de ellos desarrolla una tarea en particular, por ejemplo, la detección de longitud de la cola delante de un semáforo (escenario 1), contador de vehículos utilizando técnicas de

visión artificial (escenario 2) o simplemente controlando un panel de tráfico (escenario 3). Las dos implementaciones CORBA seleccionadas son, MICO y TAO.

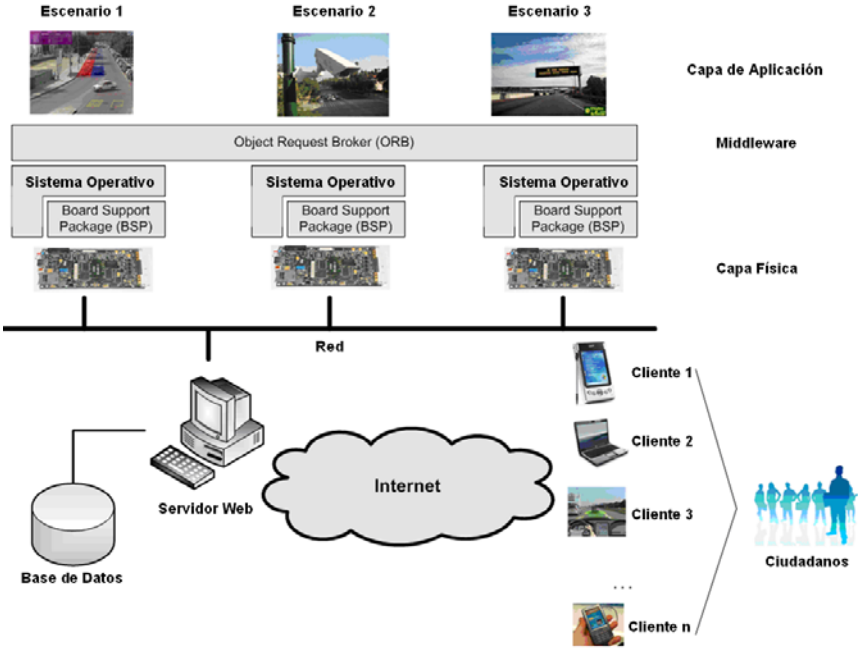


Figura 18. Esquema de un entorno inteligente urbano

El ORB implementado es el responsable de interconectar los equipos, y a su vez éstos con los usuarios locales (ciudadanos) o el centro de control de tráfico. El cliente, para los servicios disponibles, es un servidor Web que contiene un directorio actualizado continuamente ya que diferentes equipos inician o terminan sus operaciones. Los ciudadanos pueden acceder a este servicio Web de forma remota o por equipos con sistemas de asistencia al conductor. La lista detallada de los servicios incluidos en la plataforma propuesta es la siguiente:

- *Detector de Cola de Vehículos*. El equipo lleva a cabo la detección de parámetros de cola de vehículos delante de un semáforo tales como longitud instantánea, longitud media, porcentaje de ocupación de cola, fallos en ciclos semafóricos. También incluye alarmas en el caso de que la cola supere un umbral prefijado. Los límites de la región de cola se pueden configurar de forma remota mediante el dibujo de detectores virtuales en la imagen de la escena, como se muestra en la Figura 19. Las regiones azules y rojas frente al semáforo delimitan las regiones de detección de cola. El servicio consiste en la

entrega de la situación actual de cola de vehículos en un semáforo determinado [148].

- *Estimador de Densidad de Tráfico*. El equipo se basa en un sistema detector de vehículos utilizando técnicas de visión artificial (Figura 19). El servicio proporciona información sobre el flujo del tráfico y detección de incidentes [146], [148].

- *Mensajes en Paneles*. Los paneles pueden ser actualizados automáticamente mediante servicios proveídos por otros equipos conectados a la red urbana. Por ejemplo, un panel de tráfico puede señalizar automáticamente una detección de incidentes o demoras a partir de los servicios de tránsito proporcionados por los equipos anteriores.

Tanto la detección de colas de vehículos como el estimador de densidad de tráfico requieren el uso de técnicas de visión artificial basadas en el modelo de fondo. Esta técnica permite la extracción de un objeto en movimiento a partir de una secuencia de imágenes obtenidas usando una cámara estática. El modelo de fondo de la escena se utiliza para obtener una imagen de referencia con la que se compara con cada imagen de video capturada. Por consiguiente, el modelo de fondo debe ser una representación de la escena después de eliminar todos los elementos no estacionarios, y debe estar permanentemente actualizado para tener en cuenta las condiciones de luz cambiantes o cualquier cambio en la textura del fondo [145], [146]. Un estudio y comparación de diferentes algoritmos de extracción de fondo se pueden encontrar en [149]. La eliminación de sombras es una preocupación principal cuando se trata de imágenes en entornos al aire libre o imagen no estructurada, como escenas de tráfico donde las sombras de los vehículos se emiten sobre regiones vecinas. Las sombras tienen algunas propiedades particulares que pueden ser explotadas, con el fin de eliminar o al menos reducir su presencia en la imagen [150], [151]. La solución adoptada para la detección y eliminación de sombras está inspirada en el trabajo descrito en [152]. Tanto las técnicas de modelado de fondo como los de eliminación de sombras son algoritmos computacionalmente intensivos que demandan un gran uso del procesador.

La Figura 19 muestra una pequeña aplicación cooperativa entre los equipos de tráfico urbano mencionados anteriormente. La capa de middleware implementada permite a los equipos trabajar de manera cooperativa con el Servicio de Eventos de TAO CORBA. En la aplicación desarrollada, los consumidores y los proveedores se conectan entre sí mediante el canal de eventos. El Servicio de Eventos utiliza el modelo de inyección (push) para entregar eventos. Los proveedores inyectan eventos al canal de eventos y el canal de eventos inyecta eventos a los consumidores [111]. En el caso presentado, el estimador de parámetros de tráfico y

el estimador de cola en el tráfico son los proveedores de eventos, mientras que el panel de tráfico es un consumidor de eventos. Cuando el canal de evento es establecido, el mensaje del panel de tráfico se actualiza automáticamente en función de los eventos detectados por las cámaras. La Figura 19 ilustra el centro de control de tráfico, donde equipos pueden ser gestionados individualmente. En particular, las cámaras pueden ser configuradas de forma remota utilizando la página de configuración de la Figura 20, y los mensajes de los paneles de tráfico pueden ser actualizados manualmente, utilizando el botón correspondiente “Insertar nuevo aviso”.

Figura 19. Centro de Control de Tráfico

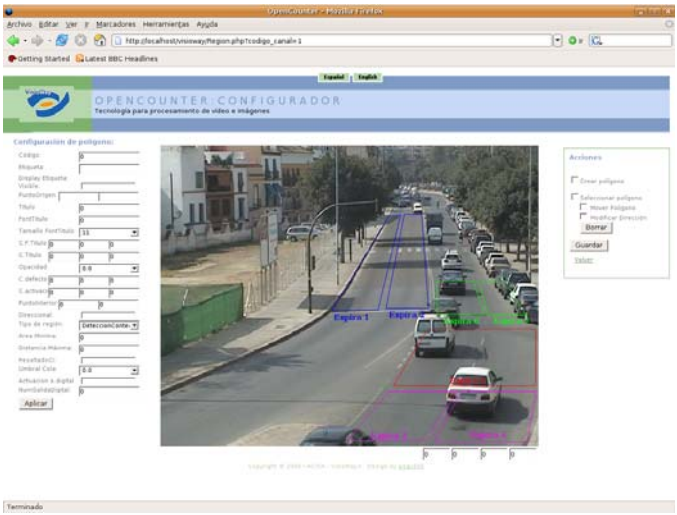


Figura 20. Sitio Web de configuración de equipamientos

4.1.1. Coste Computacional de la Capa Middleware

En el apartado anterior se ha detallado una aplicación cooperativa entre varios equipos ITS que utilizan un canal de eventos sobre CORBA. Dos de lo equipos utilizan técnicas de visión artificial que tienen un gran coste computacional, por lo que el hecho de implementar CORBA y algunos de sus servicios puede tener efectos colaterales en la aplicación principal que ejecutan. Este apartado pretende evaluar en qué medida una capa middleware llega a cargar al capacidad de computación del procesador. Para ello, se ha aplicado el método propuesto en el capítulo 3 considerando una capa de middleware como MICO y TAO desde el punto de vista del procesado de la transacción por segundo. En particular, se ha empleado el estimador de densidad de tráfico de la Figura 18, escenario 2. La Figura 21 detalla los procesos que se ejecutan en el equipo, el tiempo de CPU y los requerimientos de memoria.

```
 12:02am  up 2 min,  0 users,  load average: 1.80, 0.73, 0.27
29 processes: 25 sleeping, 3 running, 1 zombie, 0 stopped
CPU states: 70.0% user, 29.9% system,  0.0% nice,  0.0% idle
Mem:    62992K av,   19048K used,   43944K free,       0K shrd,       0K buff
Swap:       0K av,       0K used,       0K free                   7960K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
  156 root      25   0  2860 2600   952 R    98.8  4.1   1:48 epi.out
  154 root      15   0   948  948   776 R     0.9  1.5   0:01 top
  141 root      15   0  1300  536   456 S     0.1  0.8   0:00 mpegServer.out
    1 root      16   0   528  528   492 S     0.0  0.8   0:06 init
    2 root      0K   0     0    0     0 SW    0.0  0.0   0:00 keventd
    3 root      34  19     0    0     0 SWN   0.0  0.0   0:00 ksoftirqd_CPU0
    4 root      25   0     0    0     0 SW    0.0  0.0   0:00 events/0
    5 root      25   0     0    0     0 SW    0.0  0.0   0:00 kswapd
    6 root      25   0     0    0     0 SW    0.0  0.0   0:00 bdflush
    7 root      15   0     0    0     0 SW    0.0  0.0   0:00 kupdated
    9 root      25   0     0    0     0 SW    0.0  0.0   0:00 mtdblockd
   10 root      25   0     0    0     0 SW    0.0  0.0   0:00 khubd
   11 root      25   0     0    0     0 DW    0.0  0.0   0:00 mx2otg_id
   12 root      35  10     0    0     0 Z N   0.0  0.0   0:00 jffs2_gcd_mtd2 <defunct>
   29 root      35  10     0    0     0 SWN   0.0  0.0   0:03 jffs2_gcd_mtd2
   45 root      15   0  1152 1152  1060 S     0.0  1.8   0:00 sshd
  110 root      15   0   236  236   204 S     0.0  0.3   0:00 EPIWatchdog.out
  138 root      15   0  2244 2240  2136 S     0.0  3.5   0:00 httpd
  143 root      15   0  1300  536   456 S     0.0  0.8   0:00 mpegServer.out
  144 root      17   0   528  528   492 S     0.0  0.8   0:00 init
  145 root      17   0  1300  536   456 S     0.0  0.8   0:00 mpegServer.out
  146 daemon    24   0  2256 2252  2148 S     0.0  3.5   0:00 httpd
  147 daemon    24   0  2256 2252  2148 S     0.0  3.5   0:00 httpd
  148 daemon    24   0  2256 2252  2148 S     0.0  3.5   0:00 httpd
  149 daemon    24   0  2256 2252  2148 S     0.0  3.5   0:00 httpd
  150 daemon    24   0  2256 2252  2148 S     0.0  3.5   0:00 httpd
  151 root      15   0  1548 1548  1240 R     0.0  2.4   0:00 sshd
  153 root      15   0  1024 1024   824 S     0.0  1.6   0:00 sh
  158 root      15   0   468  464   408 S     0.0  0.7   0:00 webCommandServe
```

Figura 21. Tiempo de CPU y consumo de memoria del equipo

Los procesos en ejecución corresponden a la aplicación principal de procesamiento de vídeo (`epi.out`) y el servidor SSH para el registro remoto. En particular, el primero consume la mayor parte del tiempo de CPU (98,8%), pero sólo una pequeña cantidad de memoria (4,1%). Esto se debe a que el equipo cuenta con una gran cantidad de memoria (64 MB) para hacer frente a las aplicaciones de vídeo.

```
 12:37am  up 29 min,  0 users,  load average: 2.00, 1.97, 1.68
50 processes: 46 sleeping, 3 running, 1 zombie, 0 stopped
CPU states: 71.7% user, 28.2% system,  0.0% nice,  0.0% idle
Mem:    62992K av,   27068K used,   35924K free,       0K shrd,       0K buff
Swap:       0K av,       0K used,       0K free                   14092K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
  154 root      25   0  2988 2728  1064 R    97.0  4.3  28:59 epi.out
  367 root      16   0   960  960   776 R     2.1  1.5   0:00 top
  109 root      15   0   240  240   208 S     0.5  0.3   0:03 EPIWatchdog.out
  138 root      15   0  1300  536   456 S     0.3  0.8   0:05 mpegServer.out
    1 root      15   0   528  528   492 S     0.0  0.8   0:06 init
    2 root      0K   0     0    0     0 SW    0.0  0.0   0:00 keventd
    3 root      34  19     0    0     0 SWN   0.0  0.0   0:00 ksoftirqd_CPU0
    4 root      25   0     0    0     0 SW    0.0  0.0   0:00 events/0
    5 root      25   0     0    0     0 SW    0.0  0.0   0:00 kswapd
    6 root      25   0     0    0     0 SW    0.0  0.0   0:00 bdflush
    7 root      15   0     0    0     0 SW    0.0  0.0   0:00 kupdated
    9 root      25   0     0    0     0 SW    0.0  0.0   0:00 mtdblockd
   10 root      25   0     0    0     0 SW    0.0  0.0   0:00 khubd
   11 root      25   0     0    0     0 DW    0.0  0.0   0:00 mx2otg_id
   12 root      35  10     0    0     0 Z N   0.0  0.0   0:00 jffs2_gcd_mtd2 <defun
   29 root      35  10     0    0     0 SWN   0.0  0.0   0:03 jffs2_gcd_mtd2
   44 root      15   0  1152 1152  1060 S     0.0  1.8   0:00 sshd
  139 root      15   0  1300  536   456 S     0.0  0.8   0:00 mpegServer.out
  141 root      17   0  1300  536   456 S     0.0  0.8   0:00 mpegServer.out
  143 root      16   0   528  528   492 S     0.0  0.8   0:00 init
  144 root      15   0  2244 2240  2132 S     0.0  3.5   0:00 httpd
  145 daemon    24   0  2244 2240  2136 S     0.0  3.5   0:00 httpd
  146 daemon    24   0  2244 2240  2136 S     0.0  3.5   0:00 httpd
  147 daemon    24   0  2244 2240  2136 S     0.0  3.5   0:00 httpd
  148 daemon    24   0  2244 2240  2136 S     0.0  3.5   0:00 httpd
  149 daemon    24   0  2244 2240  2136 S     0.0  3.5   0:00 httpd
  150 root      15   0  1544 1544  1240 S     0.0  2.4   0:01 sshd
  152 root      15   0  1060 1060   852 S     0.0  1.6   0:00 sh
  156 root      15   0  1548 1548  1240 R     0.0  2.4   0:00 sshd
  158 root      15   0  1068 1068   856 S     0.0  1.6   0:00 sh
  159 root      15   0   468  464   408 S     0.0  0.7   0:00 webCommandServe
  240 root      15   0  4844 4840  4020 S     0.0  7.6   0:00 server
  244 root      15   0  4844 4840  4020 S     0.0  7.6   0:00 server
  245 root      15   0  4844 4840  4020 S     0.0  7.6   0:00 server
  246 root      15   0  4844 4840  4020 S     0.0  7.6   0:00 server
  247 root      15   0  4844 4840  4020 S     0.0  7.6   0:00 server
  270 root      15   0  4844 4840  4020 S     0.0  7.6   0:00 server
  271 root      15   0  4844 4840  4020 S     0.0  7.6   0:00 server
  280 root      15   0  4844 4840  4020 S     0.0  7.6   0:00 server
```

Figura 22. Tiempo de CPU y consumo de memoria, incluyendo hilos replicados

Cuando varios clientes realizan peticiones sobre el servidor que gestiona el equipo, la aplicación middleware crea varios hilos para lograr el equilibrio de carga. La Figura 22 muestra una situación en la que hasta ocho servidores son replicados para asistir a las peticiones de los clientes. Como se está utilizando el kernel `Linux 2.4.20`, los hilos del servidor se identifican como procesos con `PIDs`. Se debe tener en cuenta que el proceso principal `epi.out` sigue utilizando la mayor parte del tiempo de `CPU` ya que todos los servidores se encuentran durmiendo. Pero cuando los clientes son atendidos, el rendimiento del proceso principal se ve afectado.

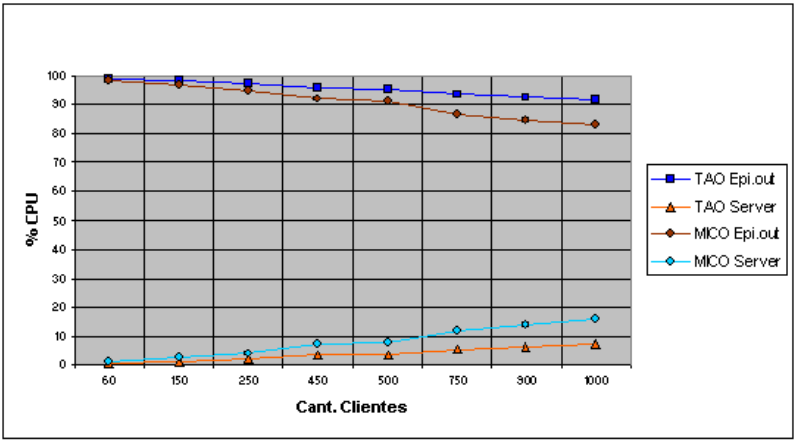


Figura 23. Evolución de la carga de CPU con el número de clientes

La metodología propuesta en el capítulo anterior se aplicó para evaluar el mecanismo de balanceo de carga sobre peticiones de clientes. En particular, la Figura 23 detalla cómo el proceso principal se ve afectado cuando el middleware implementado está atendiendo peticiones externas. Desde el lado del cliente, el número de peticiones durante un minuto se especifica como un parámetro de entrada. Desde el lado del servidor y siguiendo la metodología propuesta, se puede obtener una estimación de carga de `CPU` de la aplicación principal (`epi.out`) y el servidor (`MICO` y `TAO`), respectivamente. Los experimentos realizados han sido repetidos para promediar la variabilidad de los resultados, debido a que las aplicaciones de procesamiento de video no siempre exigen exactamente la misma cantidad de tiempo de `CPU` en cada prueba individual. Como se esperaba, los resultados obtenidos muestran que el tiempo de `CPU` para el proceso `epi.out` disminuye, mientras que el número de hilos de servidores se replica con el número de peticiones de los clientes. Se puede observar que la carga de la `CPU` del proceso principal permanece siempre por encima del 80% en el caso de `MICO`, incluso en casos de más de 1000 clientes demandando servicios. El comportamiento de `TAO` es ligeramente mejor. Los resultados obtenidos ayudan a determinar cómo la carga del procesador seleccionado debe ser sobredimensionada para soportar un funcionamiento normal. Se estima entonces que un sobredimensionamiento del 20% sería suficiente para los casos de los ejemplos anteriores.

4.2. Aplicación de la metodología propuesta para la creación de una ontología ITS

La taxonomía propuesta por la `U.S. DOT` y `RITA` y utilizada como plantilla base hasta el `LoD 4`, Figura 14 del capítulo anterior, consta de dos grandes

agrupaciones, una de ellas centrada en las Infraestructuras Inteligentes y la otra en los Vehículos Inteligentes. Gracias al método `DPK` propuesto, es posible realizar un análisis condicional y discriminatorio sobre párrafos relevantes consiguiendo de este modo una notable reducción del tamaño de la muestra, característica bastante valorada en minería de datos. En el caso de las Infraestructuras Inteligentes la reducción del tamaño de la muestra es de un 95,63% mientras que en los Vehículos Inteligentes la reducción es de 97,6%, Tabla 3.

ITS	Total de Párrafos	Párrafos Filtrados	Reducción del Tamaño de la Muestra
Infraestructuras Inteligentes	34.738	1.519	95,63%
Vehículos Inteligentes	34.738	843	97,6%

Tabla 3. Reducción del Tamaño de la Muestra después de la técnica DPK

ITS	Tamaño en MB	Porcentaje Reducido	Tamaño de Dato Útil
Infraestructuras Inteligentes	13,2	95,63%	0,58 MB
Vehículos Inteligentes	13,2	97,6%	0,32 MB

Tabla 4. Tamaño de Dato Útil después del método DPK

En la Tabla 4 se puede apreciar la reducción del tamaño total de datos. Esta reducción será aún mayor, ya que al eliminar los `stopwords` se reducen entre un 15 y 25 % el tamaño total de la muestra de datos. Por ejemplo, en el caso del contenedor "*Surveillance*" que forma parte de Infraestructuras Inteligentes, la reducción fue de 24,12%, dando un tamaño útil de datos (texto plano) de 0,44 MB.

En la Tabla 5 se muestran los contenedores que representan las palabras claves a buscar con los métodos propuestos y las frecuencias absolutas acumuladas que representan la cantidad de párrafos que hacen referencia a cada contenedor dentro de la colección de documentos analizada.

Infraestructuras Inteligentes	
Contenedor	*ni*
Traffic Control	358
Traveler Information	116
Surveillance	112
Bicycle	109
Enforcement	102
Incident Detection	98
Fleet Management	65
Weather Conditions	64
Medic	50
Ramp Flow	40
Toll Collection	38
Hazardous Material	30
Incident Response	24
Information Dissemination	22
Drayage	22
Automatic Vehicle Location	19
Special Event	17
Emergency Response	16
Pedestrian Safety	16
Parking Fee	15
In-Transit	15
Work Zone	14
Transportation Demand	14
Rail System	13
Intersection Collision	12
Border Crossing	12
Parking Management	11
Incident Clearance	11
Pre-trip Information	11
Animals	11
Data Management	10
Highway Safety	9
Safety & Security	8
Electronic Payment	8
Container Handling	7
En-route Information	6
Weather Information	4
Lane Control	4
Fare Payment	3
Asset Management	3
Frecuencia Total (N)	1519

Vehículos Inteligentes	
Contenedor	*ni*
Route Guidance	193
Collision Avoidance	122
Intelligent Speed	90
Adaptive Cruise Control	76
Lane Change	73
Speed Control	43
Driver Information	37
Road Departure	31
Lane Departure Warning	26
Object Detection	26
Obstacle Detection	24
Rear-end Collision	23
Vision Enhancement	15
Stability Control	15
Intersection Collision	12
Forward Collision Warning	11
Lane Keeping Assistance	9
On-board System	6
Sleepiness	6
In-vehicle Collision	5
Frecuencia Total (N)	843

ni=frecuencia absoluta

Tabla 5. Contenedores de Servicios - Muestra de Estudio

Sobre el total de la muestra, se encontraron 34.738 párrafos de los cuales, para el caso de las Infraestructuras Inteligentes, el tamaño útil de la muestra utilizando el método DPK fue de 1.519, discriminando y descartando los demás párrafos por su poca relevancia en base a los contenedores específicos de servicios buscados. El tamaño útil de la muestra para el caso de Vehículos Inteligentes fue de 843 párrafos. Estos párrafos filtrados representan la frecuencia total de la muestra.

Sobre esta misma base de datos reducida conseguida con el método DPK, se aplica el método IRWDP donde cada palabra tiene una ponderación (frecuencia relativa) para cada colección de documentos. El objetivo principal es tomar una

palabra con su respectivo peso y compararlo con todas las demás compilaciones, para ir formando la matriz término/colección, Tabla 2. Esta operación es realizada por cada uno de los contenedores de servicios ITS, LoD 4, de la taxonomía propuesta por RITA.

Respetando las agrupaciones principales y utilizando las frecuencias absolutas acumuladas obtenidas de cada contenedor, es posible obtener como un estudio exploratorio las frecuencias relativas porcentuales, lo que nos permitirá observar las tendencias investigadoras más relevantes en el campo de los ITS durante los años 2002-2012, Figura 24.

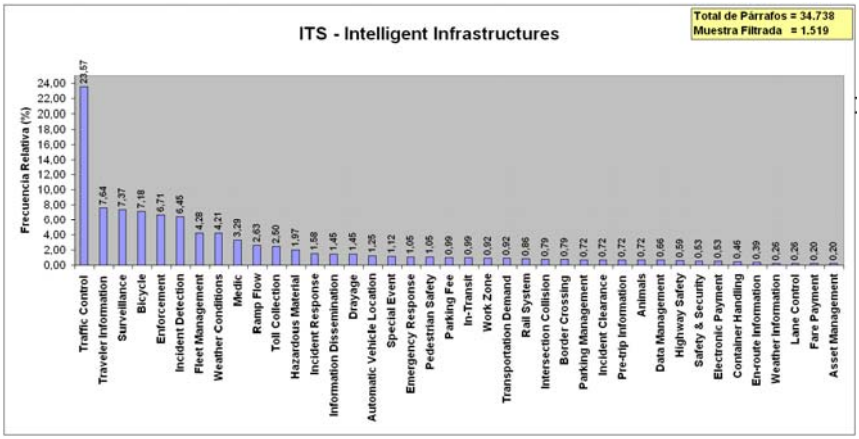


Figura 24. Frecuencia Relativa (%) de Párrafos en Infraestructuras Inteligentes

Puede notarse una clara tendencia de investigación sobre el "*Control de Tráfico*". Esto es fácil de comprender si se tiene en cuenta que en los últimos años los países han apostado enormemente por el mejoramiento de la infraestructura vial y a la seguridad.

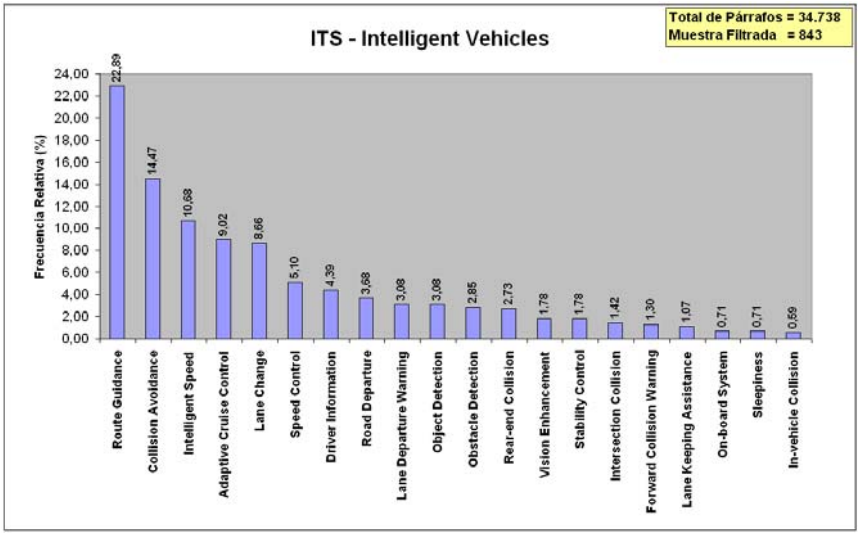


Figura 25. Frecuencia Relativa (%) de Párrafos en Vehículos Inteligentes

En el caso de los Vehículos Inteligentes Figura 25, la tendencia de investigación se encuentra más pareja pero de igual manera denota cierta preferencia sobre las "*Guías de Ruta*". Esto tampoco sorprende ya que éstas se han convertido en una herramienta importante en el alivio de la congestión en la red de transporte urbano y están estrechamente relacionadas con el "*Control de Tráfico*" de las Infraestructuras Inteligentes.

Para cada uno de los contenedores se consideran 150 palabras más relevantes para formar las matrices. Hemos comprobado que 150 términos son más que suficientes para obtener un promedio de entre 20 y 50 pares/triples de palabras. 150 términos resultantes del estudio de cada contenedor tienen muchas palabras en común (semánticamente cercanas), y algunas palabras en común para ser semánticamente distantes.

Una vez obtenidas todas las matrices por cada contenedor, lo siguiente es aplicar el método de reducción de dimensionalidad utilizando la descomposición SVD de LSA. En esencia, LSA se fija en los patrones de distribución de las palabras en las colecciones de revistas. El resultado es una base de datos LSA indexada con valores de similitud que deberán ser calculados por cada palabra de la matriz y por cada colección de revistas. El algoritmo LSA no entiende nada de lo que significan las palabras y no requiere de coincidencias exactas para obtener resultados útiles. Considerando un espacio de coordenadas cartesianas, es posible describir la ubicación de las palabras en un espacio tridimensional con los tres valores

numéricos x, y y z.

Trabajando sobre los resultados de las matrices obtenidas de los métodos propuestos y utilizando la herramienta de minería de datos Rapidminer 5 [153], [154], la Figura 26 muestra un resumen de 7 palabras (filas) de las 150 y la reducción de la dimensionalidad del contenedor *Surveillance*.

ExampleSet (150 examples, 1 special attribute, 3 regular attributes)

Row No.	Word	svd_1	svd_2	svd_3
1	traffic	0.074	0.001	-0.042
2	surveillance	0.077	0.037	0.009
3	time	0.089	0.003	0.056
4	data	0.093	0.025	0.026
5	system	0.077	0.079	0.021
6	systems	0.073	-0.009	-0.066
7	vehicle	0.051	0.071	-0.031

Figura 26. Reducción de la Dimensionalidad del Dato - LSA 3D

Siguiendo con el ejemplo del contenedor "*Surveillance*", resulta difícil visualizar para el ojo humano su representación 3D, Figura 27, Figura 28.

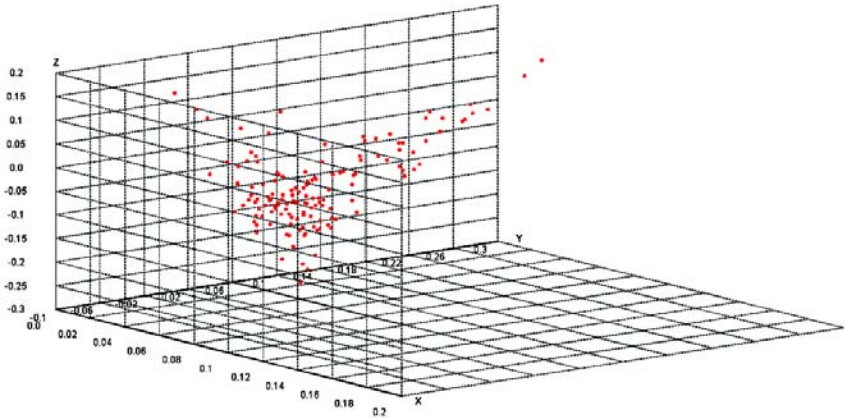


Figura 27. Contenedor "*Surveillance*" - Gráfica de Dispersión 3D

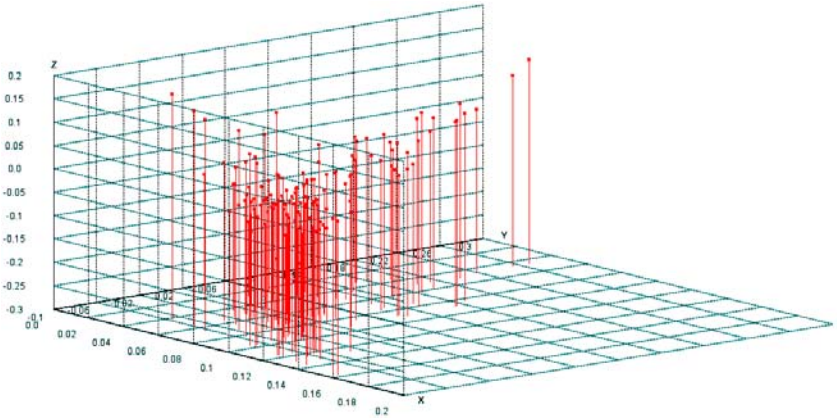


Figura 28. Contenedor "*Surveillance*" - Gráfica de Dispersión con Bastones 3D

Sin embargo, si reducimos al máximo la dimensionalidad de LSA a dos dimensiones, Figura 29, resulta fácil observar la similitud entre palabras teniendo una coordenada en el eje horizontal x y otra en el eje vertical y, presentando la información a partir de una distribución bi-variada Figura 30.

ExampleSet (150 examples, 1 special attribute, 2 regular attributes)

Row No.	Word	svd_1	svd_2
1	traffic	0.074	0.001
2	surveillance	0.077	0.037
3	time	0.089	0.003
4	data	0.093	0.025
5	system	0.077	0.079
6	systems	0.073	-0.009
7	vehicle	0.051	0.071

Figura 29. Reducción de la Dimensionalidad del Dato - LSA 2D

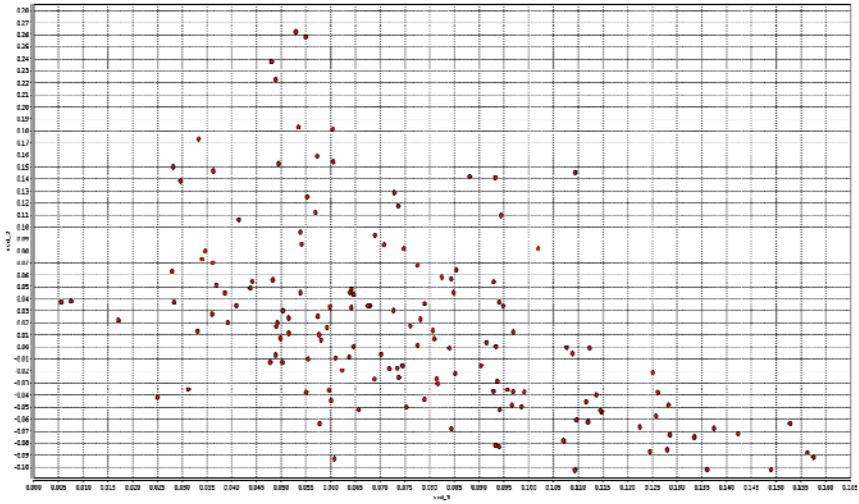


Figura 30. Contenedor "*Surveillance*" - Gráfica de Dispersión 2D

A partir de este punto y utilizando el modelo `average-linkage` de cluster jerárquico aglomerativo

$$D_{JM} = \frac{N_K D_{JK} + N_L D_{JL}}{N_M}$$

Ec. (11), se obtiene la información de distancias, que podrá ser representada como un `dendrograma` o árbol ultramétrico a partir del método `UPGMA`,

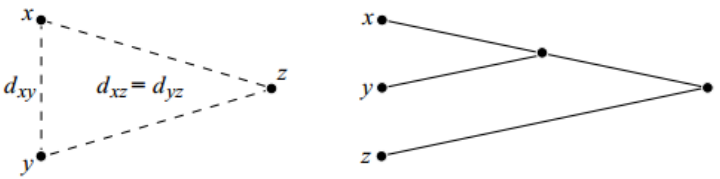
$$dist(C_1 \cup C_2, C_x) = \frac{dist(C_1, C_x) + dist(C_2, C_x)}{2}$$

Ec. (12).

Como resultado se organizarán los datos en subcategorías que se irán dividiendo en otras hasta llegar al nivel de detalle deseado. Este tipo de representación permite apreciar claramente las relaciones de agrupación entre palabras e incluso entre grupos de ellas, aunque no las relaciones de similitud o cercanía entre categorías. Observando las sucesivas subdivisiones podemos hacernos una idea sobre los criterios de agrupación de los mismos, la distancia entre los datos según las relaciones establecidas, etc. Son entonces distancias ultramétricas aquellas que cumplen con el criterio de los tres puntos (the three-point condition) [155] donde,

d es un árbol ultramétrico en O si los elementos de cada subconjunto de tres elementos de O puede ser etiquetado por x, y, z tales que,

$$d(x, y) \leq d(x, z) = d(y, z)$$



A partir del árbol ultramétrico, se extraen los pares y triples de palabras para formar la ontología final. Las distancias ultramétricas definen una topología ultramétrica. Si tomamos 3 secciones cualesquiera, las distancias entre ellas definen un triangulo isósceles, por lo que las distancias mostradas son ultramétricas. Para cualquier par de secciones, el valor de distancia en la matriz se corresponde con la suma de la longitud de las ramas en el camino más corto que los une en el árbol Figura 31.

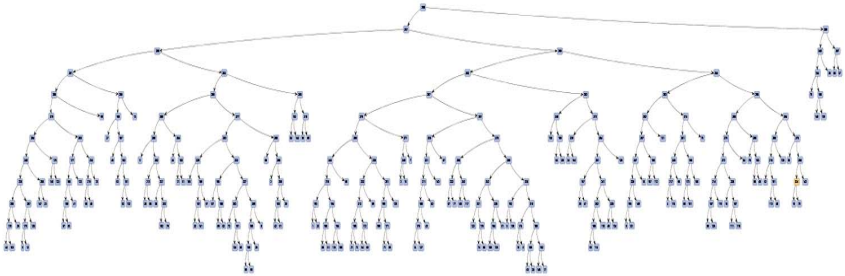


Figura 31. Vista Extendida del Árbol Ultramétrico - Contenedor "*Surveillance*"

	Word	A	B	C	D	E	F	G	H	I	J
1	traffic	0,321	0,095	0,015	0,161	0,066	0,000	0,182	0,036	0,117	0,007
2	surveillance	0,357	0,056	0,079	0,095	0,230	0,008	0,071	0,024	0,056	0,024
3	time	0,413	0,046	0,018	0,028	0,183	0,000	0,064	0,000	0,248	0,000
4	data	0,452	0,071	0,000	0,060	0,238	0,000	0,083	0,000	0,095	0,000
5	system	0,313	0,060	0,012	0,024	0,373	0,000	0,145	0,024	0,048	0,000
6	systems	0,329	0,082	0,027	0,219	0,014	0,000	0,192	0,041	0,082	0,014
7	vehicle	0,147	0,103	0,103	0,265	0,250	0,000	0,088	0,000	0,044	0,000
8	real	0,385	0,077	0,046	0,031	0,292	0,000	0,062	0,000	0,108	0,000
9	information	0,541	0,033	0,000	0,131	0,066	0,000	0,131	0,000	0,098	0,000
10	estimation	0,627	0,078	0,000	0,000	0,000	0,000	0,118	0,000	0,176	0,000
11	technology	0,167	0,071	0,071	0,048	0,619	0,000	0,024	0,000	0,000	0,000
12	demand	0,257	0,000	0,000	0,000	0,057	0,000	0,029	0,000	0,657	0,000
13	network	0,647	0,029	0,000	0,059	0,000	0,000	0,088	0,000	0,176	0,000
14	flows	0,970	0,000	0,000	0,000	0,000	0,000	0,000	0,030	0,000	0,000
15	detection	0,188	0,188	0,188	0,125	0,031	0,000	0,156	0,094	0,000	0,031
16	control	0,367	0,033	0,000	0,100	0,133	0,000	0,300	0,000	0,067	0,000
17	applications	0,207	0,103	0,069	0,069	0,448	0,000	0,103	0,000	0,000	0,000
18	transportation	0,276	0,069	0,069	0,276	0,034	0,000	0,138	0,000	0,103	0,034
19	video	0,069	0,138	0,172	0,379	0,172	0,000	0,069	0,000	0,000	0,000
20	incident	0,321	0,143	0,000	0,036	0,000	0,000	0,393	0,107	0,000	0,000

Tabla 6. Matriz término/colección - Contenedor "*Surveillance*"

Nótese que en la Tabla 6, la colección de documentos F, correspondiente a la colección de 30 publicaciones de la revista *Accident Analysis & Prevention*, presenta muchos valores 0 (cero) en el eje vertical, lo que representa la inexistencia de esas palabras en dicha colección. Nótese también que los valores están representados como frecuencias relativas, dato que ayuda al sistema a interpretar la importancia de la palabra en cada colección. Las palabras que aparecen varias veces en un documento son más significativas que las palabras que aparecen unas pocas veces. En el árbol ultramétrico, todas las ramificaciones tienen la misma longitud desde la base del árbol a través de los diversos segmentos verticales que conducen a los rótulos en la parte inferior Figura 32.

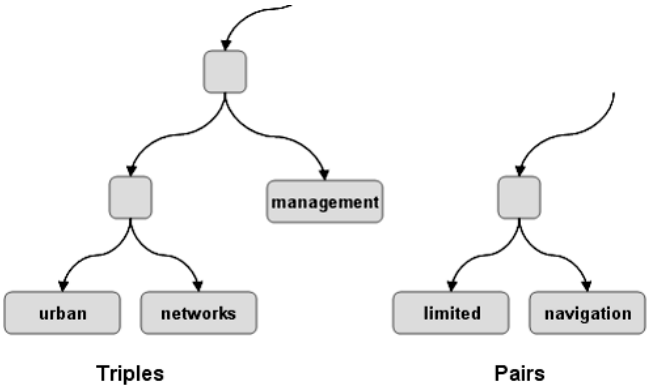


Figura 32. Pares y Triples de Palabras

En la Tabla 7, que corresponde al contenedor "*Data Management*", se puede observar una mayor ausencia de palabras en las colecciones de documentos, lo que provocará que las distancias semánticas entre los términos y la colección sean mayores. Estas distancias pueden ser fácilmente visualizadas en la Figura 33.

	Word	A	B	C	D	E	F	G	H	I	J
1	data	0,207	0,000	0,000	0,276	0,000	0,207	0,310	0,000	0,000	0,000
2	information	0,600	0,000	0,000	0,067	0,000	0,067	0,267	0,000	0,000	0,000
3	management	0,200	0,000	0,000	0,200	0,000	0,067	0,533	0,000	0,000	0,000
4	traffic	0,143	0,000	0,000	0,500	0,000	0,071	0,286	0,000	0,000	0,000
5	transport	1,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
6	systems	0,364	0,000	0,000	0,091	0,000	0,091	0,455	0,000	0,000	0,000
7	system	0,100	0,000	0,000	0,100	0,000	0,000	0,800	0,000	0,000	0,000
8	planning	0,000	0,000	0,000	0,000	0,000	0,000	1,000	0,000	0,000	0,000
9	collection	0,000	0,000	0,000	0,000	0,000	0,250	0,750	0,000	0,000	0,000
10	analysis	0,143	0,000	0,000	0,571	0,000	0,143	0,143	0,000	0,000	0,000
11	spatial	0,000	0,000	0,000	0,429	0,000	0,000	0,571	0,000	0,000	0,000
12	time	0,143	0,000	0,000	0,429	0,000	0,286	0,143	0,000	0,000	0,000
13	points	0,000	0,000	0,000	0,000	0,000	0,000	1,000	0,000	0,000	0,000
14	travel	0,000	0,000	0,000	0,600	0,000	0,000	0,400	0,000	0,000	0,000
15	global	0,000	0,000	0,000	1,000	0,000	0,000	0,000	0,000	0,000	0,000
16	intelligent	1,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
17	report	1,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
18	requests	0,000	0,000	0,000	0,000	0,000	0,000	1,000	0,000	0,000	0,000
19	transportation	0,500	0,000	0,000	0,000	0,000	0,000	0,500	0,000	0,000	0,000
20	traveler	0,750	0,000	0,000	0,250	0,000	0,000	0,000	0,000	0,000	0,000

Tabla 7. Matriz término/colección - Contenedor "*Data Management*"

Figura 33. Vista Extendida del Árbol Ultramétrico - "*Data Management*"

A partir de la identificación de los pares y triples de palabras más cercanas es posible extraer la información que se utilizará para construir la ontología. Utilizando el editor de ontologías open source Protégé [156], se realizará el modelado en un formato `OWL`, pudiendo ser portado a otros como `RDF`, `RDFS`, etc. `OWL` puede utilizarse para representar explícitamente el significado de los términos. Tomando como base los pares y triples de palabras obtenidas del estudio de todos los contenedores de servicios considerados, se seleccionan aquellos conceptos que describen los objetos independientes para construir las clases y subclases. En la Figura 34 se muestra una parte de la ontología `ITS`, concretamente una pequeña parte del contenedor "*Surveillance*", dependiente de dos clases "*Freeway_Management_Systems*" y "*Arterial_Management_Systems*" que a su vez son parte de "*Intelligent_Infrastructures*", subclase dependiente de la clase principal `ITS`. Los conceptos son derivados de la raíz `Thing` que representa la clase general de toda la ontología `ITS`. Dicho de otra forma, las clases a su vez tienen subclases que representan conceptos más específicos de dicha clase. Cada clase posee un único identificador o nombre que lo diferencia de toda la estructura. Las relaciones entre las clases representan un tipo de interacción entre los conceptos del dominio modelado. Las relaciones son definidas formalmente como cualquier subconjunto de un producto de n conjuntos, donde,

$$R : C1 \times C2 \times ... \times Cn$$

La relación `is-a` entre las clases son definidas en términos de la relación `instance-of`. La clase `Surveillance` es una subclase de la clase `Arterial_Management_Systems` si y sólo si todas las instancias de `Surveillance` son también instancias de `Arterial_Management_Systems`. Las instancias representan los objetos concretos del dominio pertenecientes a una clase. La colección de instancias constituye la base de hechos del modelo. Estas clases en sí representan dispositivos, servicios o conjunto de servicios en el entorno de los `ITS` descubiertas por la metodología aquí propuesta.

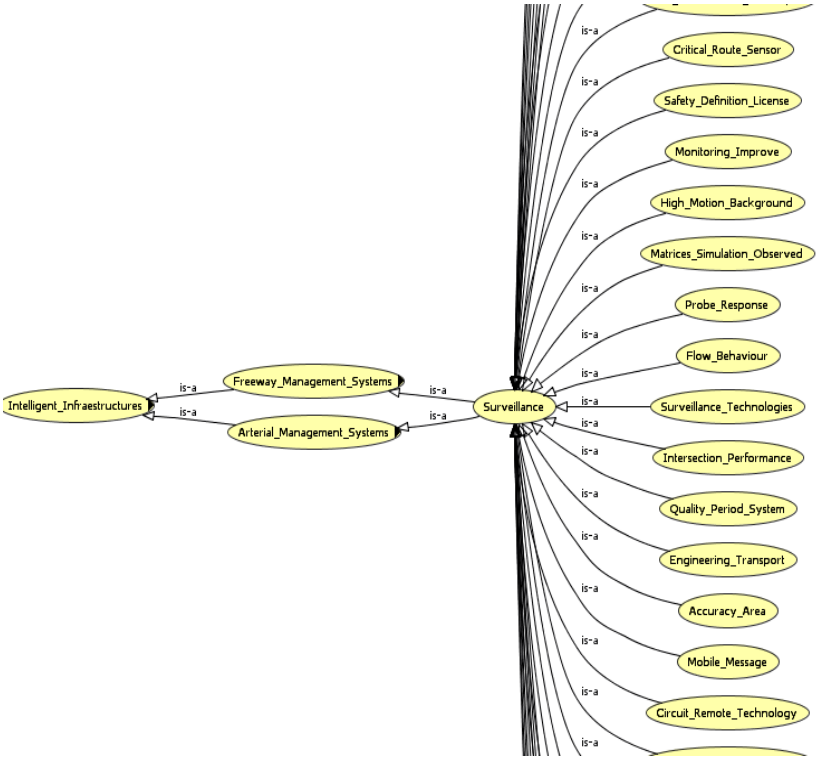


Figura 34. Parte de la Ontología ITS - Contenedor "*Surveillance*"

Por cuestiones de espacio no es posible presentar en esta tesis la ontología completa, los diagramas tipo árbol ni los demás resultados. Para acceder a la ontología en formato `OWL` se puede visitar la página http://edsplab.us.es/ITSresearch/ITS.owl. Para acceder a la ontología en formato `RDFS` se puede visitar http://edsplab.us.es/ITSresearch/ITS.rdfs y para acceder a los demás resultados (`ITS Ontology, Tree Charts, Bubble Charts, Scatter 3D Charts and the Term-Compilation Matrix`) se puede visitar http://edsplab.us.es/ITSresearch/index.htm.

4.2.1. Validación de la ontología desarrollada

La taxonomía definida por la `U.S. DOT` y `RITA` trata la clasificación de las aplicaciones `ITS`. Procuran la organización sistemática, dando nombres a grupos de elementos y a las mismas aplicaciones. Todos los términos dentro de la taxonomía están conectados mediante un modelo estructural jerárquico. Esta taxonomía

permite acceder a resúmenes clasificados por categorías definidas en diversas áreas de aplicación de los ITS. Consta de 14 aplicaciones "Infraestructuras Inteligentes" y 3 aplicaciones "Vehículos Inteligentes". Cada una de las 17 aplicaciones se encuentra dividida en sub-aplicaciones donde se ofrecen resúmenes de beneficios e información relacionada al área de interés. Sin embargo, la taxonomía no deja de ser una simple clasificación sobre las bondades, costos y beneficios de cada una de las aplicaciones. Si nos centramos en el ámbito de la arquitectura de la información que ofertan, la misma taxonomía puede constituir una herramienta básica o auxiliar para sistemas de navegación, organización o búsqueda de contenidos, pero sin ofrecer ningún tipo de semántica o lógica en el intercambio de datos.

La ontología ITS.rdfs desarrollada adiciona una lógica descriptiva. Los datos y metadatos son almacenados en repositorios los cuales permiten acceder a las demás aplicaciones. Estos repositorios permiten acceder a toda la información sobre las aplicaciones y servicios ITS descubiertos. La ontología es capaz de hacer frente a los problemas que la taxonomía de RITA no puede resolver, por ejemplo la homonimia, Figura 35.

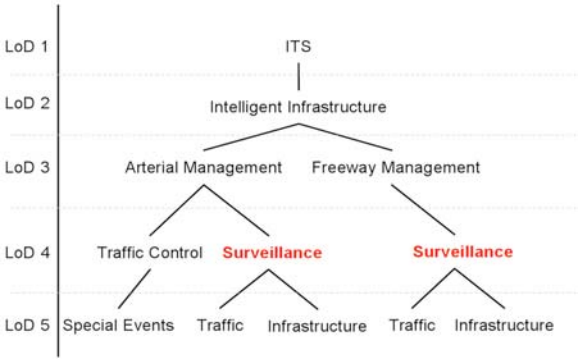


Figura 35. Homonimia en Taxonomía RITA

Nótese como en Intelligent Infrastructure existe una homonimia en el caso de Surveillance, sub-clase tanto de Arterial Management como de Freeway Management. Aún cuando estas homonimias están en el mismo LoD para el sistema son diferentes, y por lo tanto las sub-clases que dependen de éstos también. Si de una base taxonómica de servicios y aplicaciones se tratara, cualquier sistema que busque un servicio Traffic dentro de la taxonomía recibiría ambos, ya que sería incapaz de alcanzar un solo servicio. Aunque la taxonomía contribuye a la semántica de un término en el vocabulario, éstas no definen atributos entre los conceptos y por lo tanto provocaría confusión y conflictos.

La ontología ITS.rdfs se presenta más rica en cuanto a las relaciones entre términos. Estas relaciones nos permiten la expresión de conocimiento específico dentro del dominio sin la necesidad de duplicar términos evitando así homonimias.

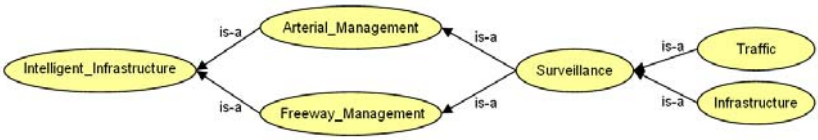


Figura 36. Solución a la homonimia en la ontología ITS.rdfs

Para el Servicio Semántico, Traffic e Infrastructure pueden ser contenedores de servicios, aplicaciones o servicios. Ambos pertenecen a la clase Surveillance y por lo tanto dependen de éste. De la misma forma, Surveillance es una sub-clase de Arterial_Management así como de Freeway_Management y estos son en sí sub-clases de Intelligent_Infrastructure. Sin embargo a diferencia de la taxonomía no existe homonimia ya que contienen diferentes significados y los nodos se encuentran en espacios semánticos distintos. Gracias a los namespaces evitamos ambigüedades en el resultado.

En el desarrollo de la ontología ITS.rdfs propuesta, la metodología empleada se basó en simplificar la problemática sin perder la categoría conceptual. Se abarcó todo el ámbito de la taxonomía propuesta por la U.S. DOT y RITA a partir de los contenedores del LoD 4, y desarrollando a partir de este nivel nuevos servicios y contenedores de servicios. Algunos de éstos han sido simplificados en nombre para evitar redundancias, por ejemplo, se han eliminado Management y System o Systems de los contenedores por ser sustantivos complementos de los nombres de interés de los servicios o contenedores de servicios. A continuación se presenta el resultado obtenido comparando la cantidad de contenedores/servicios propuestos por la U.S. DOT y RITA con la ontología desarrollada ITS.rdfs, tanto para Intelligent Infrastructures [Figura 37] como para Intelligent Vehicles [Figura 38].

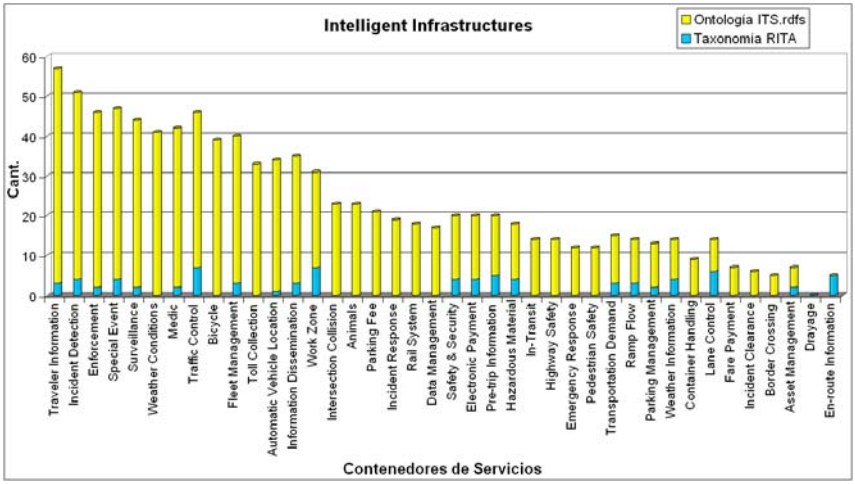


Figura 37. Contenedores encontrados en Intelligent Infrastructures

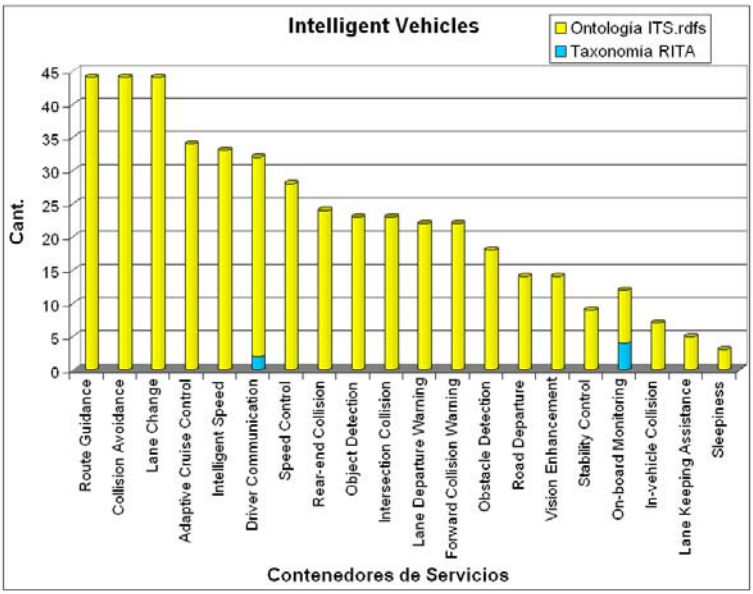


Figura 38. Contenedores encontrados en Intelligent Vehicles

En el caso de `Intelligent Infrastructures`, la cantidad de servicios/contenedores descubiertos llega a los 866 en total, mientras que en la taxonomía `RITA` ofrecen un máximo de 144 servicios/aplicaciones. De la misma forma para el caso de `Intelligent Vehicles`, la cantidad total de

servicios/contenedores descubiertos es 449, contra los 6 servicios/aplicaciones ofertados por la taxonomía de RITA. Es necesario tener en cuenta que la taxonomía no está concebida para ser una base de datos de información semántica ni para trabajar como base de datos de un Servicio Semántico.

Los servicios/contenedores descubiertos podrán ser utilizados como base para el desarrollo de nuevas aplicaciones/servicios en el ámbito de los ITS. La ontología desarrollada servirá como herramienta de referencia para la adquisición de información y para la construcción de sistemas de base de conocimiento que aportarán consistencia, fiabilidad y veracidad al momento de recuperar la información. La ontología ITS.rdfs permitirá compartir conocimiento y posibilitará el trabajo colaborativo al funcionar como soporte común de conocimiento entre diferentes actores que intervienen en la infraestructura urbana, metropolitana y rural.

4.3. Implementación del Servicio Semántico

En este apartado se plantea la creación de una plataforma distribuida que sirva de banco de pruebas para la implementación del Servicio Semántico propuesto en entornos urbanos. El SS estará implementado en una plataforma PC mientras que el exportador e importador semántico serán portados a plataformas embebidas. Se utilizan 3 equipos, un ordenador que actuará como servidor semántico, una placa ARM que contendrá el exportador semántico y otra placa ARM que contendrá el importador semántico. En un principio se utiliza una ontología base y lo que se pretende es medir el rendimiento en kT/seg. añadiendo mayor cantidad de tripletas a la base de datos principal que gestiona el Servicio Semántico. Se realizan las mismas mediciones sobre el exportador y el importador midiendo el comportamiento y rendimiento del sistema durante el proceso.

El equipamiento hardware utilizado será:

A. ***SERVER-EPIARM:***
El SS fue implementado sobre una plataforma GNU/Linux Debian 2.6.26-2 i686. El ordenador de implementación es una AMD Athlon(TM) 1200 MHz, memoria DIMM 1 Gb, disco duro de 20 Gb. El conjunto de compiladores utilizados es la colección gcc-4.1 creado por el proyecto GNU.

B. ***ARM-1 y ARM-2:***
Las implementaciones están basadas en un microprocesador de plataforma ARM 9, utilizando un compilador arm-linux gcc-3.3.2 y librerías glib-2.3.2. Fue necesario realizar una compilación cruzada para el exportador y el

importador debido a la arquitectura de hardware. Las placas poseen una CPU ARM926EJ-S, memoria de 64 Mb y un sistema operativo Linux kernel 2.4.20-celf3, desarrollados por la empresa ACISA en colaboración con investigadores de la Universidad de Sevilla.

En la Figura 39 se presenta la plataforma semántica propuesta utilizando CORBA para comunicar varios recursos distribuidos localizados en diferentes dispositivos, sobre una red TCP/IP. El SS se implementa en el equipo SERVER-EPIARM, mientras que los otros dos equipos, ARM-1 y ARM-2 son los encargados de implementar las típicas operaciones de control de tráfico.

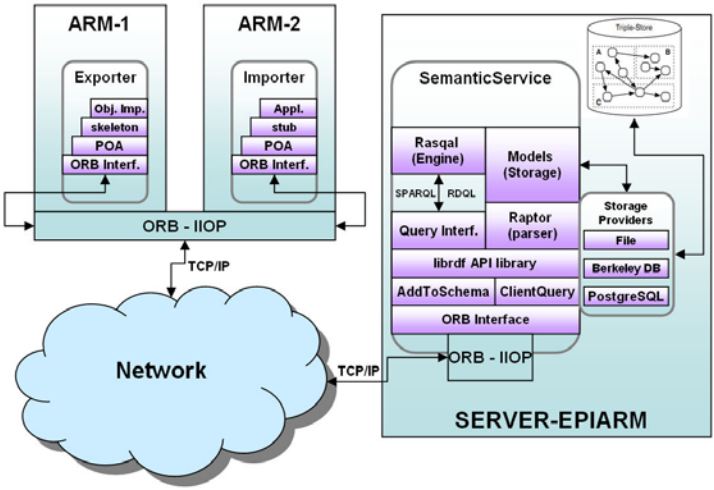


Figura 39. Arquitectura del Servicio Semántico

El Servicio Semántico, importadores y exportadores han sido implementados en CORBA TAO versión 1.6a. Se llevaron a cabo varios experimentos para evaluar el rendimiento del sistema propuesto. Una medida importante de rendimiento es el throughput. La medida está basada en el modelo como impacto de buffers finitos entre las estaciones y el tiempo de trabajo, asumiendo el tamaño del modelo en kilobytes y RTT (tiempo de retorno) en segundos.

$$TputkB = \left\{ \frac{tamañoModelo(kB)}{RTT(\text{sec.})} \right\} \qquad \text{Ec. (14)}$$

donde RTT es el “round trip time” en segundos. Sin embargo, esta medida

no es suficiente al observar que el tratamiento de los datos ontológicos es expresado en tripletas. De esta manera, el estudio se confecciona en base a un análisis mas profundo sobre las tripletas ontológicas,

$$TputkT = \left\{ \frac{(numTriples)/1000}{RTT(\text{sec.})} \right\} \qquad \text{Ec. (15)}$$

donde se representa el cálculo total de kiloTripletas de la ontología, sobre el tiempo de retorno en segundos.

Para comprobar el rendimiento general del sistema se han realizado exhaustivas pruebas sobre una base de datos `Berkeley` como almacenamiento principal. En primer lugar, al ejecutar el `SS` debe cargar un esquema o modelo con el cual se trabajará. Este esquema quedará almacenado en la Base de Datos mientras que los exportadores/importadores semánticos realizarán peticiones, siempre gestionados por el mismo `SS`.

La ontología base `ITS.rdfs` que se utiliza contiene en 524 tripletas y un peso de `108,61kB`. El proceso de análisis y el almacenamiento del esquema principal se realiza en `290ms`, con una tasa de transferencia `TputkB=374.53kB/s`. El rendimiento del `SS` se mide en la Figura 40 como una función del número de tripletas en el esquema principal almacenada en `BerkeleyDB` y en un sistema de ficheros. Sobre esta base ontológica, se realizaron mediciones sobre el rendimiento en `kT/s` agregando mayor número de tripletas, demostrando cómo se comporta el `SS` bajo situaciones de escalabilidad en el tiempo y sobre situaciones críticas.

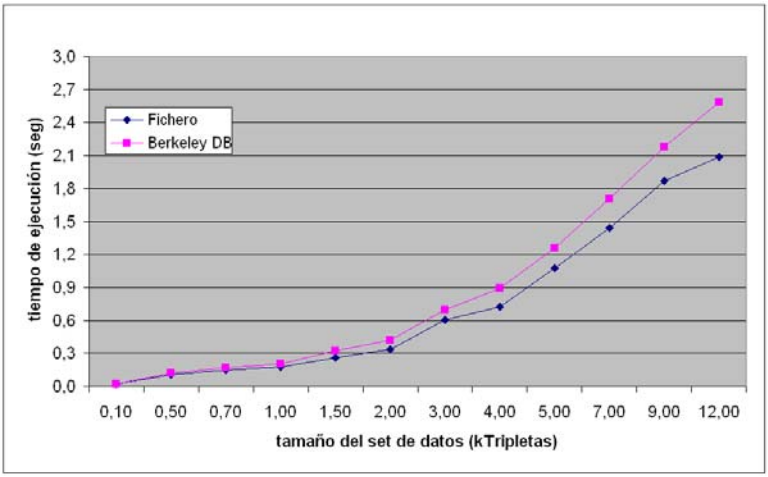


Figura 40. Parseando y Almacenando el Esquema en BerkeleyDB vs. Fichero

Se puede notar que el rendimiento durante el experimento se comporta de forma muy estable en ambos sistemas de almacenamientos, aún con sets de datos de `12kTripletas`.

4.3.1. Rendimiento del Sistema agregando declaraciones del Exportador

A continuación se midió el rendimiento del sistema agregando nuevas declaraciones del exportador al esquema principal. En este caso, el exportador crea un fichero `RDF` el cual contiene 14 declaraciones que describen de manera detallada el servicio `Car_Counter` (sujeto), sus principales propiedades (propiedades) y sus respectivas características (objetos), Figura 41. De manera general, el servicio `Car_Counter` es el encargado de realizar un conteo de vehículos en intersecciones urbanas y representa a un servicio del tipo `Video_Camera_Services`. Este `RDF` es empaquetado en un string que contiene toda la información del exportador, que será enviada al `SS`. Mientras más detallada sea la información de un servicio, más precisas e inteligentes serán las búsquedas y resultados para los importadores de información.

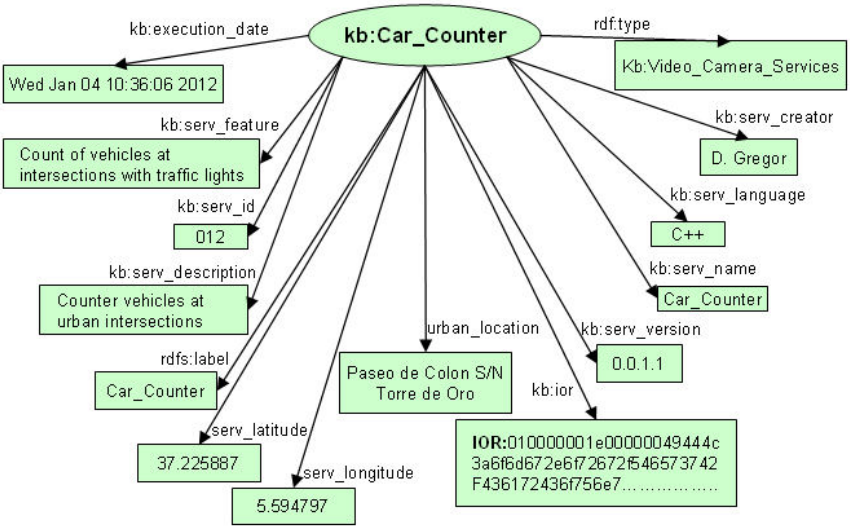


Figura 41. RDF del exportador semántico

Estas declaraciones contienen toda la información necesaria para que el importador pueda localizar al exportador sobre consultas parametrizadas. El SS recibe el `string` del exportador, busca nodos compatibles dentro de la estructura que gestiona y agrega los nuevos recursos en caso de que la concordancia sea cierta. Las 14 tripletas tienen un peso de `2.84kB`. El retardo del SS leyendo y analizando el modelo temporal recibido es de `16ms`, con una tasa de transferencia `TputkB=176.40kB/seg`. Una vez que el modelo principal es corroborado y alguno de sus nodos tiene compatibilidad con los nodos del esquema que gestiona, lo siguiente es agregar las nuevas declaraciones. El retardo buscando nodos compatibles en el esquema principal y agregando las nuevas declaraciones es de `339ms`, con una tasa de transferencia `TputKB=320kB/seg`. Con las nuevas declaraciones agregadas, el esquema principal pasa a tener 538 tripletas como base. Así como en los resultados anteriores, este proceso es ilustrado en la Figura 42, aumentando el número de tripletas en la ontología base.

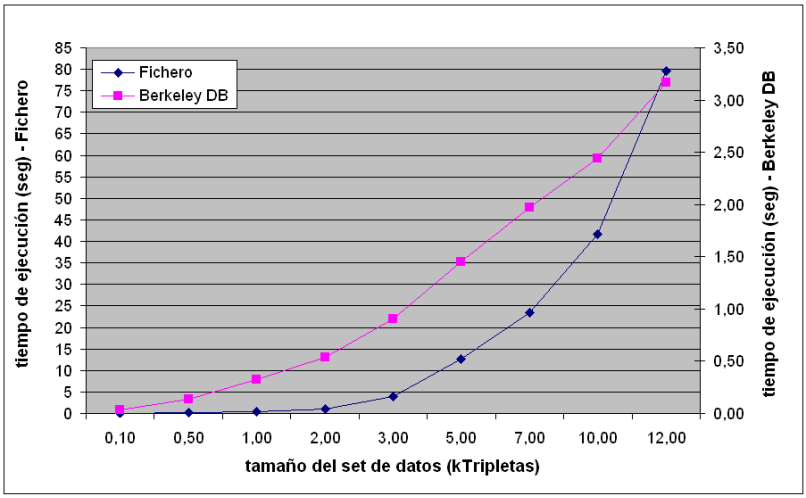


Figura 42. Buscando nodos compatibles y agregando nuevos recursos al esquema

Casi de la misma forma que en el proceso de análisis y almacenamiento del esquema principal, el rendimiento agregando nuevos recursos de un exportador se comporta de manera bastante estable, aún intentando incluir declaraciones sobre un set de datos de `12kTripletas`.

4.3.2. Rendimiento del Sistema sobre peticiones del Importador

Para realizar las pruebas con el importador se decidió utilizar `SPARQL`, siguiendo las recomendaciones de la `W3C`. El Código 5 muestra la consulta `SPARQL` *Q* realizada por el importador al `SS`. La consulta debería retornar el `IOR` del servicio `Car_Counter` así como también el creador del servicio. La consulta empieza con la cláusula `PREFIX` en la línea (1) que asocia la etiqueta `ITS` a la `IRI` `http://edsplab.us.es/kb#`. En la línea (2), la cláusula `CONSTRUCT` será la encargada de devolver al importador un grafo `RDF` construido en base a la sustitución de variables en un set de plantillas tipo tripleta.

```
(1) PREFIX kb: <http://edsplab.us.es/kb#>
(2) CONSTRUCT {?Car_Counter kb:ior ?ior .
(3)            ?x kb:serv_creator ?serv_creator }
(4) WHERE {
(5)  ?Car_Counter kb:ior ?ior .
(6)  ?x           kb:serv_creator ?serv_creator }
(7) LIMIT 1
```

Código 5. Consulta SPARQL Q del importador

En las líneas (5) y (6) se proporcionan los patrones que debe cumplir el resultado de la consulta sobre los datos del grafo. En la línea (7) se establece la cláusula `LIMIT 1`, el cual proporciona un límite superior en el número de resultados devueltos, útil en este sentido para limitar a sólo uno el resultado del `IOR`. El importador entonces realiza la consulta y el `SS` analiza la base de datos en busca de un resultado que satisfaga la petición. El retardo del `SS` realizando este análisis y construcción es de `26ms`. En este mismo proceso, el `SS` empaqueta el modelo en un `string` para enviárselo al importador. El peso del dato empaquetado que será devuelto al importador es de `1.05kB`, y contiene toda la información en formato `RDF` que el importador necesita para alcanzar al exportador. En la Figura 43 se muestra el rendimiento del proceso `ClientQuery` en el `SS` al hacer la consulta (`SPARQL`) sobre la base de datos que gestiona.

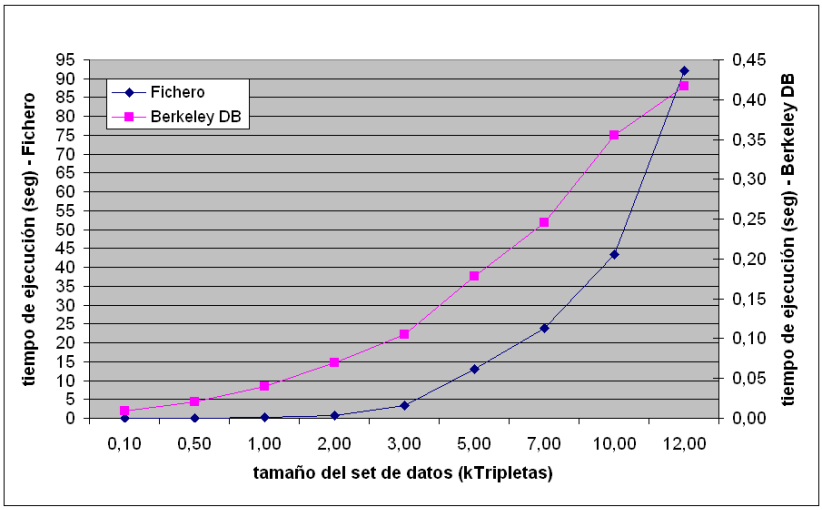


Figura 43. Parseando el esquema y retornando el resultado al importador

Desde el punto de vista de la implementación, el sistema de almacenamiento en ficheros no es factible, debido a la alta latencia obtenida. Este método no tiene indexación de datos. Sólo es útil para pequeños modelos de esquemas almacenados en la memoria. El `SS` a través de `Redland` utiliza múltiples `hash` para almacenar varias combinaciones de sujetos, predicados y objetos que requieren rápidos accesos. El almacenamiento persistente a través de `BerkeleyDB` es el más maduro y apto para modelos grandes, probado en 2-3 millones de nodos [138]. Los resultados obtenidos muestran que, incluso trabajando con un tamaño total de `12kTripletas`, el esquema en general es

analizado y la respuesta se construye en menos de medio segundo en el peor de los casos, utilizando BerkeleyDB. El importador obtiene la respuesta del SS en 41ms. Una vez que el importador recibe el string empaquetado, lo desempaqueta y reconstruye el modelo RDF analizando toda la información, según los requerimientos de la propia consulta. Para las pruebas realizadas, el importador desea encontrar dentro del modelo recibido el IOR del exportador Car_Counter. El retardo del importador extrayendo el IOR es de 56ms, mientras que el retardo realizando la petición al exportador y recibiendo una respuesta de él es de 25ms. Esto denota que en todo el proceso, desde que se realizó la consulta hasta que se recibió la respuesta del exportador, el retardo total fue de 122ms, con una tasa de transferencia TputkB=25.67kB/seg, datos bastante aceptables teniendo en cuenta el flujo de información procesado y la plataforma de ejecución embebida.

4.3.3. Rendimiento utilizando diferentes Sistemas de Almacenamientos

Desde el punto de vista de la aplicación, se realizaron pruebas sobre otro tipo de almacenamiento como la base de datos PostgreSQL y sobre un sistema de almacenamiento en fichero. Como se mencionó anteriormente, el almacenamiento en ficheros no resulta factible para este tipo de sistemas. Si bien almacenarlo en PostgreSQL era una opción, este generaba retardos en inserciones y actualizaciones. En la Figura 44 se puede ver cómo BerkeleyDB procesa mayor cantidad de tripletas en menor tiempo.

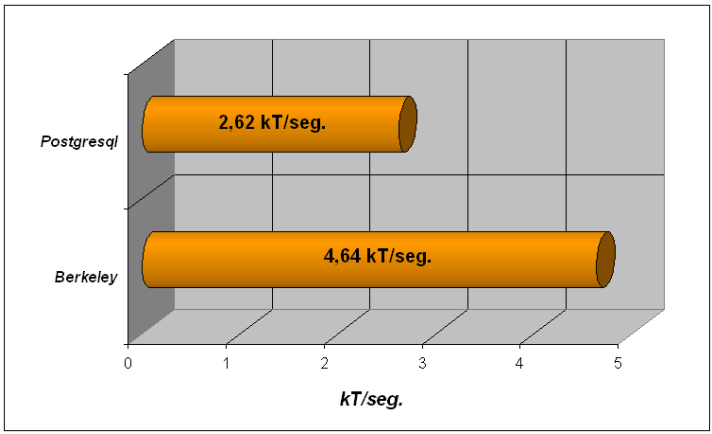


Figura 44. TputkT Parseando y Almacenando el Esquema en las Base de Datos

4.3.4. Trading Service vs. Servicio Semántico en un Escenario Real

Basada en la arquitectura de comunicación propuesta, se ha desarrollado un escenario real enfocado en la recuperación de información de servicios `ITS` en entornos urbanos, Figura 45.

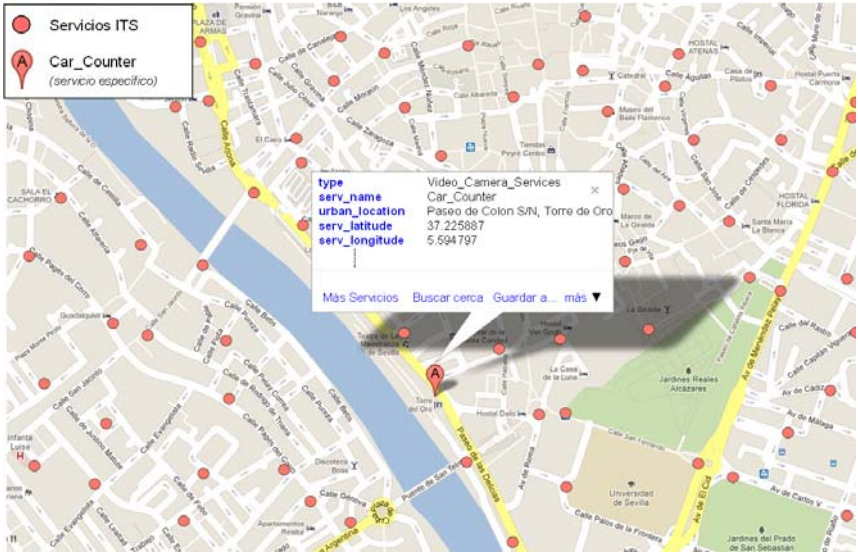


Figura 45. Escenario real recuperando la información de servicios ITS

Para ilustrar las mejoras significativas de los resultados, supongamos que un usuario decide buscar un servicio específico en la ciudad de Sevilla. Este servicio llamado `Car_Counter` tiene como misión principal la de contar coches en `"La Avenida Paseo de Colón"` y pertenece al dominio `Video_Camera_Services`. Vamos a suponer también que hay 45 posibles servicios en esa zona y tanto el Servicio Trading como el Servicio Semántico están disponibles. Obviamente, el peso de base de datos será mayor en el caso de `SS` debido al hecho de que la información se expresa en tripletas `SPO`.

El importador del Trading Service utiliza la siguiente consulta tipo restricción para alcanzar el servicio:

```
{INTERFACE_NAMES[Video_Camera_Services], "serv_name == 'Car_Counter' and
 'Paseo de Colon' ~ urban_location", "min Distance", "1"}
```

Código 6. Consulta del Importador Trading

El resultado obtenido por el importador del Trading Service es mostrado en la Tabla 8. Como se puede apreciar, se encontraron dos servicios. Aun cuando las restricciones especificadas por el importador limitan el resultado a un único servicio, el Trading Service tiene problemas para limitar el número de resultados en situaciones con homónimos. Para que el importador del Trading Service cumpla con este requisito, se le debe especificar el nombre del servicio completo "`serv_name == 'Car_Counter#0`" con el fin de obtener un solo resultado. De lo contrario, el `Trading Service` no será capaz de distinguir entre los `serv_name` y devolverá dos servicios (o más) de nombres parecidos.

PARES	
Nombre	**Valor**
serv_name	Car_Counter #0
urban_location	Paseo de Colon S/N Torre de Oro
serv_description	Servicio que cuenta vehículos urbanos en intersecciones
host_name	EpiArm12
Nombre	**Valor**
serv_name	Car_Counter #1
urban_location	Paseo de Colon 07
serv_description	Servicio que cuenta vehículos en intersecciones urbanas
host_name	EpiArm13

Tabla 8. Resultados de la consulta del Importador Trading

El importador Semántico utiliza la sintaxis del Código 7 para la recuperación de la información,

```
PREFIX kb: <http://edsplab.us.es/kb#>
CONSTRUCT {?Car_Counter kb:ior ?ior .
        ?x kb:serv_latitude ?serv_latitude .
        ?x kb:serv_longitude ?serv_longitude .
        ?x kb:urban_location ?urban_location

}
WHERE {
 ?Car_Counter kb:ior ?ior .
 ?x       kb:serv_latitude ?serv_latitude .
 ?x       kb:serv_longitude ?serv_longitude .
 ?x       kb:urban_location ?urban_location
          FILTER regex(?urban_location, "Paseo de Colon Torre de O", "i")
}
LIMIT 1
```

Código 7. Consulta del Importador Semántico

El importador semántico propuesto utiliza SPARQL para realizar la consulta sobre la base de datos que gestiona el SS. El resultado de la consulta se muestra en la Tabla 9:

TRIPLES		
Sujeto	**Predicado**	**Objeto**
Car_Counter	urban_location	Paseo de Colon S/N, Torre de Oro
Car_Counter	serv_latitude	37.225887
Car_Counter	serv_longitude	5.594797
Car_Counter	ior	IOR:010000001e00000049444c3a6f6d672e6f72672f5……

Tabla 9. Resultado de la consulta del Importador Semántico

Como diferencia con el Trading Service, que puede devolver resultados parciales sin ninguna otra indicación [157], la base de datos del Semantic Service asegura que el resultado obtenido es exitoso, de lo contrario devolvería un resultado de error. Específicamente, el uso de namespaces asegura la no duplicación en los resultados devueltos evitando problemas muy comunes como la homonimia en la recuperación de la información. La utilización del SS como arquitectura inteligente para la recuperación de información a través de tripletas y la utilización de lenguajes como SPARQL hacen que los datos puedan ser estructurados en formatos RDF/XML para la representación de la información. La Figura 46 ilustra el dato estructurado retornado al importador semántico.

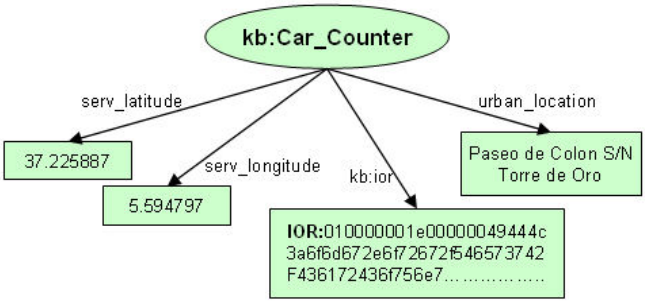


Figura 46. RDF/XML estructurado recibido por el importador semántico

La capacidad de especificar esquemas de metadatos en `RDF` permite a las aplicaciones acceder a un esquema particular o un recurso público accesible vía `Web` y recuperar la estructura y la semántica de ese conjunto concreto de elementos. Desde un punto de vista extremadamente realista, esto no asegura totalmente la búsqueda y la interoperabilidad de intercambio entre los diversos conjuntos de metadatos, pero facilita en gran medida la tarea y reduce significativamente los fallos en comparación con el `Trading Service`, donde los datos y la información están presentados en pares y los resultados pueden ser redundantes.

4.3.4.1. Retardos y Velocidad de Transferencia

La Figura 47 compara el retardo en la recuperación de información para los importadores de Trading y el Servicio Semántico, teniendo en cuenta que ambos tienen que buscar en una lista de 45 servicios. Los resultados muestran que el importador Semántico, incluso con un flujo de datos mayor, proporciona mejores resultados que el importador Trading.

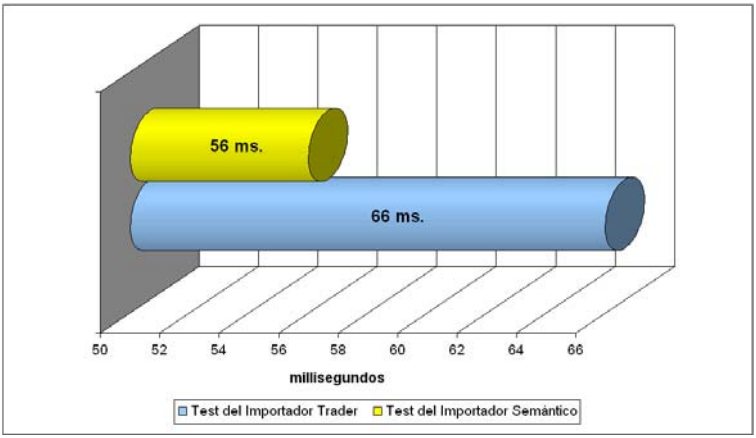


Figura 47. Retardo en la recuperación de la información

En la Figura 48 se compara la velocidad de transferencia en `kB/seg`. Obviamente, el rendimiento del `SS` es mayor, ya que está trabajando con tripletas.

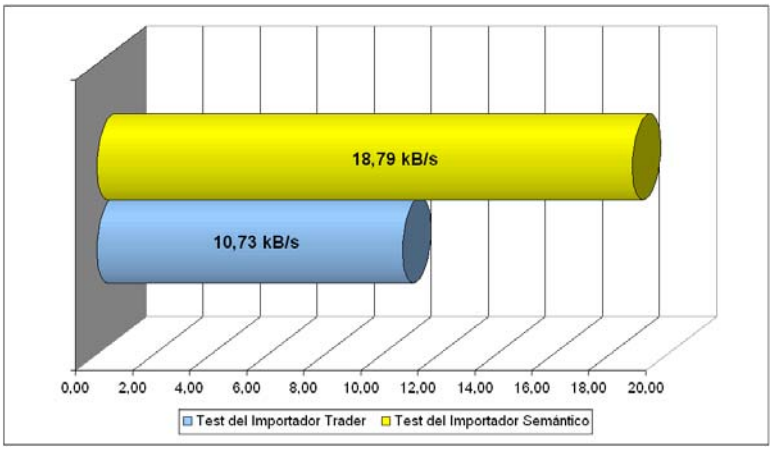


Figura 48. Throughput en la recuperación de la información

La mejora tanto en la exportación como en la importación semántica es bastante notable en comparación con los exportadores e importadores del `Trading`. Esta mejoría se debe a que el `SS` procesa un volumen mayor de datos en comparación al volumen de datos gestionado por el `Trading Service`. El `SS` es capaz de devolver los resultados a las peticiones de manera mucho más rápida ya que el mecanismo de almacenamiento y búsqueda están tercerizados.

Todo el proceso semántico es delegado principalmente a las librerías internas de redland, rasqal y raptor. Así también, el SS delega todo el mecanismo de comunicación a TAO CORBA. De esta forma el mecanismo de comunicación no interfiere con el mecanismo semántico, encargado de gestionar y almacenar metadatos de todo el sistema.

4.3.5. Rendimiento del Flujo de la Información de la Ontología ITS.rdfs

Se llevaron a cabo varios experimentos sobre la ontología completa conseguida en el apartado anterior para evaluar el rendimiento del flujo de la información sobre sistemas embebidos. La ontología ITS.rdfs contiene 2882 declaraciones. El Servicio Semántico utiliza esta ontología como contenedor de servicios descriptivos semánticos y el flujo de la información las trata como tripletas SPO [158].

Para comprobar el rendimiento general hemos almacenado la ontología obtenida en una base de datos Berkeley [140] sobre un equipo PC-AMD Athlon(TM) 1.200 MHz. Para la comunicación distribuida el Servicio Semántico es capaz de proveer el soporte en entornos distribuidos en conjunción con un set de librerías base como Redland [138] (RDF Language Bindings) para interactuar con ontologías escritas en formatos RDF y RDFS. Se utiliza un parser Raptor [141] (RDF Syntax Library) para analizar las secuencias de símbolos y determinar la estructura gramatical y un lenguaje de consultas, y Rasqal [142] (RDF Query Library) para construir y ejecutar consultas. Ambos, Raptor y Rasqal han sido diseñados para trabajar con la librería Redland. El objetivo de la tecnología distribuida de comunicación utilizada en estas pruebas es la de gestionar la información ontológica e interoperar con los servicios descubiertos por las metodologías propuestas en las secciones anteriores. En primer lugar medimos el rendimiento parseando y almacenando la ontología ITS.rdfs en la base de datos alojada en el equipo PC-AMD. El rendimiento obtenido es bastante estable durante el experimento, Tabla 10.

Parsear y Almacenar el esquema ITS.rdfs en BerkeleyDB	
TputkT Promedio	3.57 kT/seg.
Tiempo Total	806 ms.
Tamaño de Dato Total	625.58 kBytes.

Tabla 10. Rendimiento del experimento 1

En entornos urbanos, los diferentes servicios de tráfico en su mayoría son

implementados en dispositivos cuyas arquitecturas son embebidas. El siguiente proceso que se calculó fue el rendimiento del sistema agregando nuevas declaraciones de un servicio en la ontología almacenada Tabla 11. Este servicio opera y se ejecuta en un dispositivo con plataforma `ARM926EJ-S` y la función principal es la de exportar la información que deberá ser agregada a la ontología `ITS.rdfs`. El servidor (exportador) crea un fichero del tipo `RDF` que contiene 14 declaraciones (tripletas). Este `RDF` es empaquetado dentro de un string y contiene toda la información del servidor que será útil en un sistema `C/S` distribuido, para que el cliente (importador) pueda acceder a él.

Leer y Analizar el Modelo Temporal recibido del Exportador	
Velocidad de Transferencia (TputkB)	176.40 kB/seg.
Retardo Total	16 ms.
Tamaño de Dato Total	2.84 kBytes.

Tabla 11. Rendimiento del experimento 2

Con las nuevas 14 declaraciones agregadas más las 2882 declaraciones que la ontología `ITS.rdfs` poseía, el esquema principal pasa a tener 2896 tripletas. Por otro lado, el importador, que se encuentra implementado en otro dispositivo con plataforma `ARM926EJ-S`, realiza una consulta al `Servicio Semántico` que gestiona la Base de Datos que contiene la ontología `ITS.rdfs` en busca de un resultado que satisfaga una petición `SPARQL` [109], Código 8. El retardo resolviendo la consulta y construyendo la respuesta para ser enviada al cliente es de `26ms`. El peso del dato empaquetado que será devuelto al cliente es de `1.05kB` y contiene toda la información en formato `RDF` que el importador necesita para entablar comunicación con el exportador que se encuentra en otro dispositivo. El importador obtiene esta respuesta empaquetada en `41ms`. Tabla 12.

```
(1) PREFIX kb: <http://edsplab.us.es/kb#>
(2) CONSTRUCT {?Car_Counter kb:ior ?ior .
(3)            ?x kb:serv_creator ?serv_creator }
(4) WHERE {
(5)  ?Car_Counter kb:ior ?ior .
(6)  ?x           kb:serv_creator ?serv_creator }
(7) LIMIT 1
```

Código 8. Consulta SPARQL del cliente (importador)

Retardo resolviendo la consulta y construyendo la respuesta para enviar al Importador	
Retardo Total	26 ms.
Tamaño Total del Dato a Enviar	1.05 kBytes.
Retardo Total (Lado-Importador)	41 ms.

Tabla 12. Rendimiento del experimento 3

En este apartado se presentaron los resultados obtenidos del análisis de rendimiento del flujo de la información. Por un lado se midió el rendimiento parseando y almacenando la ontología `ITS.rdfs` en una base de datos en Berkeley obteniendo un promedio `Tput` en kilotripletas de `3.57kT/seg`. en `806ms`. Se midió la velocidad de transferencia leyendo y analizando el modelo temporal recibido del exportador semántico alcanzando los `176.40kB/seg`. con un retardo de `16ms`. Por último se midió el retardo resolviendo la consulta del importador semántico el cual alcanza los `26ms`. El `SS` empaqueta el resultado obtenido de la consulta y lo envía al importador, este paquete de datos pesa `1.05kBytes`. El importador recibe estos resultados en `41ms`. Los resultados obtenidos demuestran las grandes posibilidades que conllevan la implementación semántica y la flexibilidad de representar metadatos en entornos de los `ITS`. La incorporación de ontologías favorecería en gran medida la comunicación entre dispositivos y aplicaciones logrando la comprensión común entre ellas.

Capítulo 5 - Conclusiones, Futuros Trabajos y Publicaciones

5.1. Conclusiones

Las siguientes conclusiones se desglosarán en tres epígrafes. En el primero se expondrán las ventajas de la computación distribuida en las redes de tráfico urbano, destacando las metodologías propuestas. En el segundo se presentan las conclusiones de las metodologías propuestas para la medición de la carga computacional de la capa middleware sobre entornos embebidos, la construcción de la ontología a partir de la recuperación de la información y las bondades de la incorporación del Servicio Semántico en el ámbito de los ITS actuales. Por ultimo, en el tercer epígrafe se presenta la conclusión a las pruebas realizadas resaltando las mejoras que proporciona la incorporación de la ontología como base del conocimiento en el ámbito de los ITS. Se presentan las conclusiones sobre la implementación y los resultados experimentales del Servicio Semántico propuesto, exponiendo la factibilidad y efectividad de todo el enfoque.

5.1.1. Sistemas cooperativos en entornos urbanos

En esta tesis se ha presentado una plataforma de middleware embebido en entornos urbanos inteligentes capaz de interconectar equipos diseminados por las ciudades y de ofrecer nuevos servicios de valor añadido, integrados en un entorno inteligente. Las ventajas de desarrollar entornos cooperativos en entornos urbanos son enormes. La creación y la implementación de arquitecturas distribuidas diseminadas en la red e interconectados por sistemas cooperativos de comunicación universales tipo middleware permitirán la cooperación y el intercambio de información entre infraestructuras, vehículos y usuarios. Estos sistemas cooperativos redundarían enormemente en beneficio del sector del transporte y de los ciudadanos. Además de mitigar la congestión del tráfico y la contaminación que provoca, los ciudadanos disfrutarían de una mayor seguridad vial y desplazamientos más rápidos. Gracias a la disponibilidad semántica de la información se podrá tener una descripción formal de las bondades de un servicio en particular. Podrá ser más fácil localizar, seleccionar y combinar servicios o aplicaciones que cumplan tareas concretas. Estos servicios y aplicaciones podrán ser fácilmente programados para actuar en base a eventos generados en el tráfico.

En la actualidad no existe un sistema integrado en todos los aspectos que intercambien información semántica entre sistemas o dispositivos. Los avances presentados en esta tesis sobre la incorporación de ontologías como base del conocimiento e intercambio de datos entre todos los agentes que intervienen en los ITS aportarían mayor expresividad en la representación de la información.

La utilización de metadatos y lenguajes SPARQL, RDF, RDFS, etc. permitirán la compatibilidad Web permitiendo la interoperabilidad entre diferentes entidades. La cooperación de estos dispositivos, aplicaciones y servicios utilizando ontologías permitirá la estandarización y la incorporación de nuevos estándares de funcionalidad.

5.1.2. Metodología propuesta

Se ha demostrado que la incorporación de sistemas middleware simplifica el proceso de desarrollo de aplicaciones al independizarse de los lenguajes o plataformas de cada uno de los equipos que componen la red. Estos sistemas middleware facilitan en gran medida el desarrollo de sistemas complejos de diferentes tecnologías y arquitecturas posibilitando el uso de dispositivos de escasos recursos. Sin embargo, esta posibilidad tiene también un coste en términos de consumo computacional debido al software necesario para implementar la capa middleware. Por ello se ha propuesto una metodología para medir el impacto de la capa middleware en la operación normal de un procesador ARM9. Más concretamente, se midió el impacto y el rendimiento de dos middleware de código abierto (MICO y TAO) en equipos urbanos cuyo propósito principal es el control de tráfico a través del procesamiento de vídeo. La metodología se aplicó para evaluar la carga de CPU sobre peticiones de clientes, bajo ráfagas de peticiones de entre 50 y 1000 por minuto. El principal resultado obtenido es que la ejecución de una capa middleware completa como MICO o TAO es compatible con una elevada carga computacional del procesador y se estima que, en el peor de los casos, implica un reducción de un 20% de CPU. El método propuesto puede resultar de gran utilidad para evaluar en qué medida una aplicación principal puede verse afectada por las peticiones de clientes y para dimensionar adecuadamente el procesador a utilizar.

También se ha propuesto una metodología para la construcción de una ontología en el dominio de los ITS a partir de la IR y la SLR. Se presentaron dos nuevos métodos novedosos, DPK e IRWDP los cuales se encargan de reducir el volumen de la muestra de estudio y de armar la matriz de términos/colección definiendo las frecuencias relativas de las palabras. A partir de las matrices construidas se aplicó la reducción de dimensionalidad con el método LSA/SVD. Se realizó la aglomeración jerárquica tipo clustering y se construyeron los árboles ultramétricos para extraer los pares/triples de palabras. Como resultado, se consiguió desarrollar la ontología donde se clasificaron los servicios ITS más relevantes. Las técnicas propuestas fueron aplicadas en la construcción de la infraestructura ontológica en el campo de los ITS y puede servir como guía de desarrollo en construcciones ontológicas similares, en otras áreas de dominio.

El campo de estudio puede ser ampliado tomando como muestras una mayor cantidad de publicaciones. A pesar de que las publicaciones utilizadas como muestra se encuentran en colecciones y que se realizaron extracciones de párrafos, descartando entre 95% y 98% del total de información, los párrafos seleccionados han sido semánticamente clasificados ya que están estrechamente relacionados de acuerdo a su similitud y a la búsqueda de palabras claves. Esto no resulta útil para muchas técnicas en minería de datos, pero para la elaboración de ontologías donde los sujetos, predicados y objetos deben representar un significado semántico, las técnicas propuestas demuestran que gracias a la discriminación de datos irrelevantes se obtienen los resultados esperados.

Finalmente, se ha desarrollado un Servicio Semántico en TAO CORBA que será el encargado de dar soporte de comunicación utilizando ontologías como base del conocimiento. El tratamiento de los datos semánticos fueron delegados a las librerías: Redland, Raptor y Rasqal. El SS ofrece la interconexión y hace de intermediario entre importadores/exportadores. Los exportadores facilitan los servicios y los importadores los consumen. Desde el punto de vista de la interoperabilidad que ofrece el SS, la ventaja más llamativa es la recuperación y la disponibilidad inteligente de la información. La incorporación de semánticas en la localización, invocación, descubrimiento, composición, ejecución y monitorización de servicios, se muestra como una solución factible en sistemas distribuidos. Gracias a la incorporación del SS en la infraestructura propuesta se proporciona un mecanismo formal para describir semánticamente todos los aspectos relevantes de los servicios urbanos facilitando la automatización en el descubrimiento, combinando e invocando servicios urbanos en plataformas distribuidas. Esta ventaja unida a las posibilidades de CORBA para trabajar en sistemas heterogéneos permite mapear los servicios en diferentes plataformas para interactuar con nuevos dispositivos implementados en diferentes arquitecturas.

5.1.3. Conclusiones sobre los resultados obtenidos

Gracias a la incorporación de los métodos DPK e IRWDP, es posible extraer la información, recopilar, reducir, clasificar y automatizar en gran medida la creación de una ontología en el ámbito de los ITS. Apoyados en los resultados obtenidos de los métodos propuestos, se realizaron pruebas de rendimiento sobre la ontología utilizando el Servicio Semántico. En las pruebas realizadas se obtuvieron un promedio de 3.57kT/seg. y 806ms. sobre un peso total de 625.58kBytes, parseando y almacenando el esquema ontológico completo. Utilizando un sistema distribuido cliente/servidor implementados en plataformas embebidas ARM9, se midió el rendimiento del flujo de datos entre

éstos y la ontología almacenada en una Base de Datos Berkeley gestionada por el Servicio Semántico. Los resultados experimentales demostraron la factibilidad y efectividad del enfoque. La recuperación de la información representa un reto importante en el campo de los ITS. Si un dispositivo desea hacer uso de alguno de los servicios ofrecidos por la ontología, (ej. El Servicio de Vigilancia en el Tráfico o El Servicio de Prioridad en el Control del Tráfico, Servicio de Gestión de Semáforos, Servicio de Avisos en Paneles, etc.), una ontología ofrecerá la manera de acceder a ellas, más toda la información relativa de manera descriptiva, lógica e inteligente.

Se hicieron comparaciones entre métodos de almacenamiento, encontrando los mejores resultados en sistemas de almacenamiento en base de datos tipo BerkeleyDB. Gracias a la incorporación de las consultas tipo SPARQL del lado del importador semántico se evita trabajar con aplicaciones rígidas y poco flexibles. El importador sólo necesitaría modificar la consulta para obtener otro tipo de resultados. Se realizaron pruebas de rendimientos y se hicieron comparaciones con el Trading Service de TAO CORBA con muy buenos resultados por parte del Semantic Service, partiendo de la base que tanto el importador como el exportador se encontraban ejecutándose sobre una plataforma embebida ARM9. En el SS, las pruebas fueron realizadas sobre un número máximo de 12 kTripletas. Evidentemente el throughput se ve afectado a mayor número de kiloTripletas. Sin embargo, para el caso del almacenamiento en BerkeleyDB, se mantiene estable con un promedio de 3.53kT/seg. En el proceso ClientQuery se logra un promedio de 27.53kT/seg. Apoyados en estos resultados se desarrolló una aplicación real enfocada en la recuperación de la información donde el importador semántico demuestra ser mucho más eficiente en comparación que el importador del Trading Service. A pesar de la variedad de formatos existentes en el mundo de la información, en muchos casos efímeros e incompatibles, la información representada en pares no satisface las necesidades de accesibilidad, diseminación y conservación que caracterizan a la gestión del conocimiento.

Esta tesis doctoral pone de relieve las infinitas posibilidades que ofrece la recuperación de información semántica, así como la flexibilidad de uso de metadatos en entornos ITS, donde la importancia en el tratamiento de los datos es crucial. El objeto principal de la aplicación de ontologías en el campo de los ITS es la posibilidad de volver a utilizar la base de conocimientos existente para mejorar la exactitud de la información recibida/enviada, información que facilita la comunicación entre los dispositivos, aplicaciones y usuarios, logrando un entendimiento común entre ellos. El SS propuesto será capaz de proporcionar un mecanismo formal para describir semánticamente todos los aspectos relevantes de los servicios urbanos, facilitando la automatización en el descubrimiento, la

combinación y la invocación de los servicios en plataformas altamente distribuidas. Gracias a la adaptabilidad de las características del SS, éste será capaz de utilizar ontologías de cualquier tipo y trabajar en base al conocimiento general. El SS podrá ser adaptado fácilmente para interactuar con servicios y todo tipo de información que dependerán directamente de la ontología que gobierna. La localización semántica de servicios de determinadas características y propiedades de forma dinámica e inteligente sin duda promete mayor interacción y compromiso con el tipo de dato y la exactitud en la información.

5.2. Futuros Trabajos en el ámbito de los ITS y tratamiento de la información semántica

En lo que respecta al desarrollo de la ontología, las técnicas, métodos y herramientas utilizadas para su desarrollo, como futuro trabajo se podría extender el tamaño de la muestra y hacer comparaciones con los resultados aquí obtenidos. En el caso de la técnica IRWDP, se podría limitar la longitud de las palabras con un mínimo de 4 caracteres y un máximo de 15 ya que, en ocasiones, algunos caracteres indeseados pueden formar parte de la recopilación. Se podría utilizar el método WPGMA y compararlo con los resultados aquí obtenidos con el método UPGMA. Así también, LSA no es una solución perfecta ya que los lenguajes son sistemas demasiado complejos como para segmentarlos por temáticas agrupando familias de términos. Las palabras por sí mismas no tienen mucho significado para el sistema, somos los humanos los que les damos sentido. En éste sentido se deberían explorar otras herramientas o técnicas semánticas y compararlas con los resultados aquí conseguidos.

En lo que respecta a los lenguajes ontológicos, si bien RDFS se presenta como un modelo simple para la representación de metadatos, este no indica cardinalidad ni se pueden derivar tipos por unión o intersección de los datos. Es necesaria la incorporación de una herramienta de relaciones de clases o propiedades más ricas como por ejemplo OWL. Actualmente este lenguaje se implementa más en java y está más enfocado a los WS. Es necesario implementarlo en otros lenguajes como C o C++ para poder portarlo a plataformas embebidas ya que implementar una maquina virtual java en un sistema embebido afectaría directamente al rendimiento de toda la arquitectura.

Gracias al diseño portable de las ontologías éstas pueden utilizarse como contenedores de grandes bases de datos que den soporte a servicios de valor añadido para el usuario en sistemas inteligentes de transporte como: gestión de cobros en vehículos públicos, gestión de rutas en sistemas de transporte, mercancías, personas, gestión de rutas para vehículos de emergencia, localización y

alarmas de vehículos frente a robos, botón de pánico frente a robos y secuestros, solicitud automática de auxilio, detección de rutas mas cortas con datos de tráfico, alarmas anti-colisión, detección de obstáculos en aparcamientos, pagos electrónicos, control de velocidad media del vehículo, información al conductor sobre: señales de tráfico, aparcamientos (precio, distancia), estado de las carreteras, cambios de trazado en rutas, obras, etc.

Desde la perspectiva del `Servicio Semántico` desarrollado, podría extenderse en el sentido de enlazarlo formando grafos federados de información semántica. La federación hará que se puedan utilizar un mayor tamaño de datos. El `SS` federado proporcionaría un único servicio lógico a los importadores/exportadores mientras que los físicos podrán ejecutarse en diferentes sitios remotos. Asimismo, se podría definir una arquitectura distribuida más amplia con mayor cantidad de dispositivos y en varias arquitecturas implementados directamente sobre el entorno urbano. Se podría dotar al `Servicio Semántico` de una estrategia o mecanismo autónomo para comprobar que los servicios publicados en la base de datos ontológica que gobierna, estén correctamente operativos y disponibles. Adicionalmente se sugiere la creación y prueba con un mayor numero de dispositivos urbanos, exigiendo al SS a interactuar bajo altas ráfagas de peticiones.

Son innumerables las posibilidades de continuar desarrollando las ideas y tecnologías presentadas. El uso de ontologías en tareas de recuperación de la información es una actividad emergente que va ganando terreno a medida que se van generando e integrando distintas ontologías. La incorporación de técnicas semánticas, recuperación de la información y la utilización de metadatos en el campo de los ITS será sin duda un punto clave en la interoperabilidad y el trabajo cooperativo de los futuros sistemas distribuidos.

5.3. Publicaciones en Congresos y Revistas

Los resultados obtenidos en el marco de esta tesis han dado lugar a publicaciones tanto en revistas de reconocido prestigio así como en congresos internacionales. Entre las aportaciones más relevantes a revistas se pueden destacar:

[1] S. L. Toral, D. Gregor, M. Vargas, F. Barrero, F. Cortés, "***Distributed Urban Traffic Applications based on CORBA Event Services***", International Journal of Space-Based and Situated Computing (IJSSC), Vol. 1, No.1 pp. 86-97, Switzerland, 2011.

[2] M.R. Martínez-Torres, S.L. Toral, F.J. Barrero, D. Gregor, "***A Text Categorization Tool for Open Source Communities based on Semantic Analysis***", International Journal on the Human Aspects of Computing, Behaviour & Information Technology, DOI:10.1080/0144929X.2011.624634, 2011.

[3] S. L. Toral, F. Barrero, F. Cortés, D. Gregor, "***Analysis of embedded CORBA middleware performance on urban distributed transportation equipments***", Computer Standards & Interfaces, Volume 35, Issue 1, Pages 150–157, DOI:10.1016/j.csi.2012.06.004, July 2012.

[4] D. Gregor, S.L. Toral, T. Ariza, F. Barrero, "***An Ontology-based Semantic Service for Cooperative Urban Equipments***", Journal of Network and Computer Applications, Volume 35, Issue 6, Pages 2037-2050, DOI 10.1016/j.jnca.2012.08.002, September 2012.

Finalmente, entre las aportaciones realizadas a congresos se pueden destacar las siguientes:

[1] S. L. Toral, D. Gregor, J. M. Milla, M. Vargas, F. Barrero, F. Cortés, "***Smart Cameras for Cooperative Urban Appplications***", IADIS International Conference on Collaborative Technologies, Friburgo - Germany, July 2010.

[2] F. Cortés, D. Gregor, S. L. Toral, F. Barrero, "***A Wireless CORBA Adaptation for Building a Multi-user Environment***", IADIS International Conference on Collaborative Technologies, Friburgo - Germany, July 2010.

[3] F. Cortés, D. Gregor, S. L. Toral, F. Barrero, "***Development of a Bluetooth CORBA based Intelligent Infrastructure for Multi-user Urban Environments***" INCOS Second International Conference on Intelligent Networking and Collaborative Systems, Thessaloniki, Greece, November 2010.

[4] F. Cortés, D. Gregor, S. L. Toral, F. Barrero, "***Development of a Bluetooth Intelligent Multi-user Environment based on Wireless CORBA***", International Conference on Complex, Intelligent, and Software Intensive System, Page(s): 343–349, Thessaloniki, Greece, August 2011.

Bibliografía

[1] ICT Shok, "*Cooperative Traffic ICT*" v2, http://www.tivit.fi/fi/dokumentit/63/SRA+Cooperative+Traffic+ICT+v2.pdf, 2009.

[2] Fuencisla Sancho Gómez, "*Últimos avances en ITS en la red estatal de autopistas de peaje*", Delegación del Gobierno en las Sociedades Concesionarias de Autopistas Nacionales de Peaje, Congreso ITS-España, abril, 2011.

[3] Gardels K., "*Automatic Car Controls for Electronic Highways*", General Motors Research Lab. Warren, Michigan, Report GMR-276, June, 1960.

[4] Mitropoulos, G.K. et al., "*Wireless Local Danger Warning: Cooperative Foresighted Driving Using Intervehicle Communication*", Intelligent Transportation Systems, IEEE Transactions, IEEE journals, Page(s): 539-553, 2010.

[5] Thomas, T., van Berkum, E.C., "*Detection of incidents and events in urban networks*", Intelligent Transport Systems, IET, Volume: 3, Issue: 2, IET JOURNALS, Page(s): 198-205, 2009.

[6] Malik, H., Rakotonirainy, A., "*The Need of Intelligent Driver Training Systems for Road Safety*", Systems Engineering, ICSENG '08. 19th International Conference, IEEE CONFERENCES, Page(s): 183-188, 2008.

[7] PROMETEO, Plataforma Tecnológica Española de Sistemas con Inteligencia Integrada, http://www.prometeo-office.org, 2012.

[8] ETSI, European Telecommunications Standards Institute, http://www.etsi.org, 2012.

[9] CAR 2 CAR, Communication Consortium, http://www.car-to-car.org, 2012.

[10] A. Hinsberger, H. Wieker, G. Riegelhuth, H. Zurlinden, "*Benefits and technology of an intelligent roadside unit system for vehicle to infrastructure and infrastructure to centre communication*", 14th World Congress on Intelligent Transport Systems, pp 1-8, 2007.

[11] M. Bechler, W.J. Franz, and L.C. Wolf, "*Mobile Internet Acces in FleetNet*", in Proc. KiVS Kurzbeiträge, pp.107-118, 2003.

[12] A. Festag, G. Noecker, M. Strassberger, A. Libke, B. Bochow, M. Torrent-Moreno, S. Schnaufer, R. Eigner, C. Catrinescu, J. Kunisch, (X), "*(NoW) Network on wheels: project objectives, technology and achievements*", 5th International Workshop on Intelligent Transportation (WIT), pp. 211–216, 2008.

[13] G. Toulminet, J. Boussuge, C. Laurgeau, "*Comparative synthesis of the 3 main European projects dealing with Cooperative Systems (CVIS, SAFESPOT and COOPERS) and description of COOPERS Demonstration Site 4*", 11th International IEEE Conference on Intelligent Transportation Systems, ITSC 2008, pp. 809-814.

[14] S.W. Han, S.K. Song, H.Y. Youn, "*CALM: An Intelligent Agent-based Middleware Architecture for Community Computing*", Proceeding of SEUS-WCCIA'06, pp. 89 – 94, 2006.

[15] EasyWay, "*EasyWay project overview*", http://www.easyway-its.eu, Retrieved April 14, 2009.

[16] IEEE Computer Society, "*IEEE 802.11 Standard*", http://standards.ieee.org/getieee802/download/802.11-2007.pdf, 2007.

[17] CEN, European Committee for Standardization, http://www.cen.eu, 2012.

[18] Chou W., "*The growing role of IT in transportation*", IEEE Journals, Volume 5, Issue 6, Pages 5-6, 2003.

[19] IEEE Computer Society, IEEE Microwave Theory and Techniques Society, "*IEEE 802.16 Standard*", http://standards.ieee.org/getieee802/download/802.16-2009.pdf , 2009.

[20] ETSI Technical Committee, "*Intelligent Transportation System*", http://www.etsi.org/website/homepage.aspx, 2011.

[21] Kosch T., et al, "*Communication Architecture for Cooperative Systems in Europe*", Communications Magazine IEEE Journals, Volume 47, Issue 5, Pages 116-125, 2009.

[22] R. Bossom et al., "*European ITS Communication Architecture - Overall Framework*", COMeSafety System Architecture, http://www.comesafety.org, October, 2008.

[23] Msadaa I.C., Cataldi P., Filali F., "*A Comparative Study between 802.11p and Mobile WiMAX-based V2I Communication Networks*", Next Generation Mobile Applications, Services and Technologies (NGMAST), Fourth International Conference, 2010.

[24] Daneshfar F., RavanJamJah J., Mansoori F., Bevrani H., Azami B.Z., "*Adaptive Fuzzy Urban Traffic Flow Control Using a Cooperative Multi-Agent System based on Two Stage Fuzzy Clustering*", Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th, Page(s): 1-5, 2009.

[25] Brickley O., Chong Shen, Klepal M., Tabatabaei A., Pesch D., "*A Data Dissemination Strategy for Cooperative Vehicular Systems*", Vehicular Technology Conference, VTC2007-Spring. IEEE 65th, Page(s): 2501-2505, 2007.

[26] Rockl M., Robertson P., "*Data Dissemination in Cooperative ITS from an Information-Centric Perspective*", Communications (ICC), 2010 IEEE International Conference on, Page(s): 1-6, 2010.

[27] Terziyan V., Kaykova O., Zhovtobryukh D., "*UbiRoad: Semantic Middleware for Context-Aware Smart Road Environments*", Internet and Web Applications and Services (ICIW), International Conference IEEE, 2010.

[28] W.-H. Lee, S.-S. Tseng, W.-Y. Shieh, "*Collaborative real-time traffic information generation and sharing framework for the intelligent transportation system*", Journal Information Sciences, Volume 180 Issue 1, Pages 62-70, January, 2010.

[29] F. Barrero, S. L. Toral, S. Gallardo, "*eDSPLab: remote laboratory for experiments on DSP applications*", Internet Research, Vol. 18, no. 1, pp. 79-92, 2008.

[30] S. L. Toral, M. R. Martínez-Torres, F. Barrero, "*Virtual Communities as a resource for the development of OSS projects: the case of Linux ports to embedded processors*", Behavior and Information Technology, Vol. 28, Iss. 5, pp. 405-419, 2009.

[31] L.-X. Zhu, F.-Y. Wang, "*Component-based constructing approach for application specific embedded operating systems*", Proc. 2003 Intelligent Transportation Systems, Vol. 2, pp. 1338-1343, 2003.

[32] K. Yaghmour, "*Building Embedded Linux Systems*", O'Reilly, 2003.

[33] S. Spanos, A. Meliones, G. Stassinopoulos, "*The internals of advanced interrupt handling techniques: Performance optimization of an embedded Linux network interface*", Computer Communications, Vol. 31, Iss. 14, pp. 3460-3468, 2008.

[34] S. Fuchs, D. Bankosegger, "*Developing value networks for I2V co-operative services: An Austrian example*", IET Intelligent Transport Systems, Vol. 3, Iss. 2, pp. 216-224, 2009.

[35] A. Tuominen, T. Ahlqvist, "*Is the transport system becoming ubiquitous? Socio-technical roadmapping as a tool for integrating the development of transport policies and intelligent transport systems and services in Finland*", Technological Forecasting and Social Change, vol. 77:1, pp. 120-134, 2010.

[36] Thanikesavan Sivanthi, Ulrich Killat, "*Valuing the Design Flexibility of a Distributed Real-time Embedded System*" Industrial Embedded Systems, SIES 2008, International Symposium, Junio de 2008.

[37] W. Ruh, T. Herron and P. Klinker, "*IIOP Complete: Understanding CORBA and Middleware Interoperability*" Addison-Wesley, Reading, MA, 1999.

[38] Arnold, K., Gosling, J., Holmes, D., "*The JAVA™ Programming Language*", Addison Wesley Professional, Fourth Edition, 2005.

[39] David, S. Linthicum, "*Next Generation Application Integration: From Simple Information to Web*", Part II: Application Integration Technology, Chapter 6. Middleware Basics, Types of Middleware, August 15, 2003.

[40] Luis E. Mendoza, "*Estudio de Tecnologías Middleware para Sistemas Peer-to-Peer*", VII Jornadas Internacionales de las Ciencias Computacionales, Colima, México, 2005.

[41] W3C, World Wide Web Consortium, Oficina Española, "*Guía Breve de Servicios Web*", mayo de 2010.

[42] Pavel Kulchenko, Randy J. Ray, "*Programming Web Services with Perl*", O'Reilly & Associates, Inc., 2003.

[43] Werner Vogels, Cornell University, "*Web Services Are Not Distributed Objects*", IEEE Internet Computing, November - December 2003.

[44] R. Chinnici et al., "*Web Services Description Language (WSDL) Version 1.2*", World Wide Web Consortium, http://www.w3.org/TR/2007/REC-wsdl20-20070626/, June 2007.

[45] "*Universal Description, Discovery and Integration*", Organization for the Advancement of Structured Information Systems, http://uddi.xml.org/, 2002.

[46] F. Curbera et al., "*Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI*", IEEE Internet Computing, vol. 6, no. 2, pp. 86-93, Mar./Apr. 2002.

[47] W3C Recommendation, "*Simple Object Access Protocol (SOAP) 1.2*", World Wide Web Consortium, April 2007, http://www.w3.org/TR/soap12-part1/.

[48] W. Wu et al., "*Service Oriented Architecture for VoIP Conferencing*", Special Issue on Voice over IP — Theory and Practice, Int'l. J. Commun. Sys., vol. 19, no. 4, pp.445-61, May 2006.

[49] I. Brunkhorst, S. Tonnies, W. Balke, "*Multimedia Content Provisioning using Service Oriented Architectures*", Proc. IEEE ICWS, Beijing, China, October 2008.

[50] Ahmed N., Linderman M., Bryant J., "*Towards Mobile Data Streaming in Service Oriented Architecture*", Reliable Distributed Systems, 29th IEEE Symposium, Page(s): 323-327, 2010.

[51] R. Varadan et al., "*Increasing Business Flexibility and SOA Adoption through Effective SOA Governance*", IBM Systems J., vol. 47, no. 3, pp. 473-490, 2008.

[52] A. Arsanjani et al., "*SOMA: A Method for Developing Service-Oriented Solutions*", IBM Systems J., vol. 47, no. 3, pp. 377-396, 2008.

[53] M. Henning, "*A New Approach to Object-Oriented Middleware*", IEEE Internet Computing Magazine 1, pages 66-75, 2004.

[54] A. Tripathi, "*Challenges designing next-generation middleware systems*", Communications of the ACM 6, pages 39-42, 2002.

[55] OMG, Common Object Request Broker Architecture: Core Specifications, http://www.omg.org/technology/documents/formal/corba_iiop.htm, version 3.0.3, Dec. 2006.

[56] Z. Tari, O. Bukhres, "*Fundamentals of Distributed Object Systems: The CORBA Perspective*", John Wiley & Sons, Inc., 2001.

[57] Ural Mutlu, Reuben Edwards, Paul Coulton, "*QoS aware CORBA Middleware for Bluetooth*", Consumer Electronics, IEEE Tenth International Symposium, ISCE, Junio de 2006.

[58] K.-C. Liang, D. Chyan, Y.-S. Chang, W.-T. Lo, S.-M. Yuan, "*Integration of CORBA and object relational databases*", Computer Standards & Interfaces, 25 (4), pages 373-389, 2003.

[59] Ciaran McHale, "*CORBA Explained Simply*", February 27, 2007.

[60] Diego Sevilla Ruiz, Michi Henning, Steve Vinoski, "*PROGRAMACIÓN AVANZADA EN CORBA CON C++*", Editorial Pearson Educación S.A., 2002.

[61] Object Management Group, "*The Common Object Request Broker: Architecture and Specification*", Version 3.0.1, November 2002.

[62] Object Management Group, "*CORBA for embedded Specification*", OMG document ptc/06-08-03, Agosto de 2006.

[63] Gokhale A. and Schmidt D. C., "*The performance of the CORBA dynamic invocation interface and dynamic skeleton interface over high-speed ATM networks*", IEEE GLOBECOM '96, Noviembre de 2006.

[64] Steve Viki, "*CORBA: Integrating Diverse. Applications Within Distributed Heterogeneous Environments*", IEEE Communications Magazine, vol. 14, no. 2, Febrero de 1997.

[65] Foster, I. and Kesselman, C. (editors), "*The Grid: blueprint for a future Computing infrastructure*", Morgan Kaufmann Publishers, USA, 1999.

[66] Foster, I.; Kesselman, C. y Tuecke, S., "*The Anatomy of the Grid: Enabling Scalable Virtual Organizations*", Lecture Notes in Computer Science, vol. 2150, 2001.

[67] Gokhale, A.; Schmidt, D.C "*Techniques for optimizing CORBA middleware for distributed embedded systems*", INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings. IEEE, Marzo de 1999.

[68] Object Management Group, "*Minimum CORBA Specification*", edición 2.3, Disponible en http://www.omg.org/, document id: 02-08-01, Agosto de 2002.

[69] John Bard; Vincent J. Kovarik Jr., "*Software defined radio – The Software Communications Architecture*", Editorial Wiley, Catálogo Británico ISBN 978-0-470-86518 (HB) - 2007.

[70] A. Puder, "*MICO: An Open Source CORBA Implementation*", IEEE Software, Page 17-19, Abril de 2004.

[71] Bo Chen; Cheng, H.H., "*A Review of the Applications of Agent Technology in Traffic and Transportation Systems*", Intelligent

Transportation Systems, IEEE Transactions, Volume: 11, Issue: 2, Page(s): 485 – 497, 2010.

[72] L. Li, F. Wang, Q. Zhou, "*Integrated Longitudinal and Lateral Tire/Road Friction Modeling and Monitoring for Vehicle Motion Control*", IEEE Trans. Intelligent Transportation Systems, Volume: 7, no. I, pp. 1-19, Mar. 2006.

[73] F. Wang, D. Zeng, L. Yang, "*Smart Cars on Smart Roads: An IEEE Intelligent Transportation Systems Society Update*", IEEE Pervasive Computing, Volume: 5, no. 4, pp. 68-69, 2006.

[74] Tekli J., Damiani E., Chbeir R., Gianini G., "*SOAP Processing Performance and Enhancement*", Services Computing, IEEE Transactions, Volume: PP, Issue: 99, Page(s): 1, 2011.

[75] Phan K. A., Tari Z., Bertok P., "*Similarity-Based SOAP Multi-cast Protocol to Reduce Bandwidth and Latency in Web Services*", IEEE Transactions on Services Computing (IEEETSC) Volume: 1, No2, pp.88-103, 2008.

[76] Dasheng Wang; Lei Ren; Jing Li, "*Modeling intelligent transportation systems with multi-agent on SOA*", Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference, Page(s): 717 – 720, 2010.

[77] Saleem M.Q., Jaafar J., Hassan M.F., "*Model driven security frameworks for addressing security problems of Service Oriented Architecture*", Information Technology (ITSim), 2010 International Symposium, Page(s): 1341 - 1346, 2010.

[78] Zhiyun Guo, Meina Song, Qian Wang, "*A framework of enterprise cloud application*", Web Society (SWS), 2010 IEEE 2nd Symposium, Page(s): 729 – 732, 2010.

[79] Kabbani N., Tilley S., Pearson L., "*Towards an evaluation framework for SOA security testing tools*", Systems Conference, 2010 4th Annual IEEE, Page(s): 438 – 443, 2010.

[80] M.P. Papazoglou et al., "*Service-Oriented Computing: State of the Art and Research Challenges*" Computer, Vol. 40, no. 11, 2007.

[81] Jepsen, T.C., "*Just What Is an Ontology, Anyway?*", IT Professional, Volume 11, Issue: 5, 2009.

[82] A. Gómez-Pérez, "*Some Ideas and Examples to Evaluate Ontologies*", tech. report KSL-94-65, Knowledge System Laboratory, Stanford Univ., 1994.

[83] Yajing Zhao, Jing Dong, Tu Peng, "*Ontology Classification for Semantic-Web-Based Software Engineering Services Computing*", IEEE Transactions on Volume: 2, Issue: 4, 2009.

[84] Huhns M.N., Singh M.P., "*Ontologies for agents*", Internet Computing, IEEE, Volume 1, Issue: 6, 1997.

[85] Lee D., Meier R., "*Primary-Context Model and Ontology: A Combined Approach for Pervasive Transportation Services*", Pervasive Computing and Communications Workshops, PerCom Workshops '07. Fifth Annual IEEE International Conference, 2007.

[86] K. Chen, C. Dow, and S. Guan, "*NimbleTransit: Public transportation transit planning using semantic service composition schemes*", 11th Int. IEEE Conference, Intelligent Transportation Systems, pp. 723–728, Beijing China, 2008.

[87] Fernandez A. and Ossowski S., "*A Multiagent Approach to the Dynamic Enactment of Semantic Transportation Services*", Intelligent Transportation Systems, IEEE Transactions on, vol. 12, no. 2, pp. 333-342, June 2011.

[88] Jun Zhai, Yan Cao, Yan Chen, "*Semantic information retrieval based on fuzzy ontology for intelligent transportation systems"*, Systems, Man and Cybernetics (SMC), IEEE International Conference, Page(s): 2321 - 2326, 2008.

[89] B. Kitchenham, S. Charters, "*Guidelines for Performing Systematic Literature Reviews in Software Engineering*", Version 2.3, Technical Report, Keele University and University of Durham, 2007.

[90] He Zhang, Muhammad Ali B., Paolo Tell, "*Identifying relevant studies in software engineering*", Information and Software Technology, Volume 53 Issue 6, June, 2011.

[91] M. Gruninger, M.S. Fox, "*Methodology for the design and evaluation of ontologies*", Proceedings of International Joint Conference of Artificial Intelligence Workshop on Basic Ontological Issues in Knowledge Sharing, pp. 1-10, Montreal, Canada, 1995.

[92] M. Fernández, A. Gómez-Pérez, N. Juristo, "*Methontology: from ontological art towards ontological engineering*", AAAI 97 Spring Symposium on Artificial Intelligence in Knowledge Management, Stanford University, pp. 33-40, California, USA, 1997.

[93] J. Nanda, T.W. Simpson, S.R.T. Kumara, S.B. Shooter, "*A methodology for product family ontology development using formal concept analysis and web ontology language*", Journal of Computing and Information Science in Engineering 6 (2), 103–113, 2006.

[94] Gruber, T. R., "*A translation approach to portable ontology specifications*". Knowledge Acquisition Vol. 5:199-220, Abril de 1993.

[95] W3C Members, "*RDF/XML Syntax Specification*", W3c Recommendation, Feb, 2004.

[96] Veronika Vaneková, Ján Bella, Peter Gurský, Tomáš Horváth, "*Fuzzy RDF in the Semantic Web: Deduction and Induction*", Pavol Jozef Šafárik University in Košice, Institute of Computer Science, Park Angelinum 9, 04001 Košice, Slovakia, 2008.

[97] W3C Members, "*RDF Vocabulary Description Language 1.0: RDF Schema*", W3c Recommendation, 10 February, 2004.

[98] Mcguinness D.L., Fikes R., Hendler J., Stein L.A., "*DAML+OIL: an ontology language for the Semantic Web*", Intelligent Systems, IEEE, Volume: 17, Issue: 5, 2002.

[99] F. van Harmelen, "*A Model-Theoretic Semantics for DAML+OIL (March 2001)"* W3C Note 18, World Wide Web Consortium, www.w3.org/TR/2001/NOTEdaml+oil-model-20011218, December 2001.

[100] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks, "*OWL Web Ontology Language Semantics and Abstract Syntax*", Technical report, W3C working draft, Mar. 2003.

[101] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, "*RQL: A declarative query language for RDF*", WWW, pp. 592–603, 2002.

[102] Seaborne A., HP Labs Bristol, "*RDQL - A Query Language for RDF*", http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/, W3C Member Submission, 9 january, 2004.

[103] L. Miller, A. Seaborne, A. Reggiori, "*Three implementations of SquishQL, a simple RDF query language*", ISWC, pp. 399–403, 2002.

[104] E. Prud'hommeaux, A. Seaborne (Eds.), "*SPARQL query language for RDF*", http://www.w3.org/TR/rdf-sparql-query/, W3C Recommendation, 15 January 2008.

[105] Gutierrez C., Hurtado C. A., Alberto O., "*Foundations of Semantic Web databases*", Journal of Computer and System Sciences, Volume 77, Issue 3, Pages 520-541, Database Theory, May 2011.

[106] Stocker M., Seaborne A., Bernstein A., Kiefer C., Reynolds D., "*SPARQL Basic Graph Pattern Optimization Using Selectivity Estimation*", 17th International Conferencia on WWW, Beijing – China, 21-25 April, 2008.

[107] Kanwei Li, "*Cost Analysis of Joins in RDF Query Processing Using the TripleT Index*", Thesis submitted to the Faculty of the Graduate School of Emory University for the degree of Master of Science in Mathematics and Computer Science, 2009.

[108] W3C Members, "*Web Services Description Language (WSDL) Version 2.0*", http://www.w3.org/TR/wsdl20/, W3C Recommendation, 26 June, 2007.

[109] W3C Members, "*SPARQL Protocol for RDF*", W3C Recommendation, http://www.w3.org/TR/rdf-sparql-protocol/, 15 January, 2008.

[110] Knarig Arabshian, Peter Danielsen, "*Semi-automated Ontology Creation for High-level Service Classification*", Alcatel-Lucent, Bell Labs Murray Hill, NJ, USA, Seventh International Conference on Semantics, Knowledge and Grids, 2011.

[111] S. L. Toral, D. Gregor, M. Vargas, F. Barrero, F. Cortés, "*Distributed Urban Traffic Applications based on CORBA Event Services*", International Journal of Space-Based and Situated Computing, Suiza, 2011.

[112] IEEE Standards, "*IEEE Standard for a Smart Transducer Interface for Sensors and Actuators - Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats*", Page(s): 0_1 – 430, 2004.

[113] Steffen Lamparter, Björn Schnizler, "*Trading services in ontology-driven markets*", University of Karlsruhe (TH), Karlsruhe, Germany, Symposium on Applied Computing, 2006.

[114] R. Schantz and D. Schmidt, "*Middleware for Distributed Systems: Evolving the Common Structure for Network-centric Applications*", Encyclopedia of Software Engineering, Wiley and Sons 2002.

[115] I. Gorton, A. Liu, P. Brebner, "*Rigorous evaluation of COTS middleware technology*", Computer 36 (3), 50-55, 2003.

[116] S. L. Toral, F. Barrero, F. Cortés, D. Gregor, "*Analysis of embedded CORBA middleware performance on urban distributed transportation equipments*", Computer Standards & Interfaces, Volume 35, Issue 1, Pages 150–157, DOI:10.1016/j.csi.2012.06.004, July, 2012.

[117] RITA and U.S. Department of Transportation, "*Taxonomy of Intelligent Transportation Systems Application*", http://www.itslessons.its.dot.gov/its/benecost.nsf/images/Reports/$File/Taxonomy.pdf.

[118] USDOT, "*United States Department of Transportation*", http://www.dot.gov.

[119] RITA, "*Research and Innovative Technology Administration*", http://www.its.dot.gov/index.htm, 2010.

[120] Cristina López-Pujalte, Vicente P. Guerrero Bote, Félix de Moya Anegón, "*Retroalimentación por Relevancia: Nueva Perspectiva desde la Programación Evolutiva*", Facultad de Biblioteconomía y Documentación, Universidad de Extremadura y Facultad de Biblioteconomía y Documentación, Universidad de Granada, http://www.scimago.es/publications/jotri-03b.pdf.

[121] Salton G., Buckley C., "*Improving retrieval performance by relevance feedback*", Journal of the American Society for Information Science, 41(4), 288-297, 1990.

[122] Jieping Ye, Janardan R., Park C.H., Park H., "*An optimization criterion for generalized discriminant analysis on under sampled problems*", Pattern Analysis and Machine Intelligence, IEEE Transactions, Volume: 26 , Issue: 8, Page(s): 982 - 994 , 2004.

[123] Deerwester S., Dumais S. T., Furnas G.. W., Landauer T. K., and Harshman R., "*Indexing by Latent Semantic Analysis*", Journal of the American Society for Information Science, 41, 391-407, 1990.

[124] Foltz P. W., "*Using Latent Semantic Indexing for Information Filtering*", R. B. Allen (Ed.) Proceedings of the Conference on Office Information Systems, 40-47, Volume 11 Issue 2-3, Cambridge, 1990.

[125] Foltz P. W., "*Latent Semantic Analysis for text-based research*", Behavior Research Methods, Instruments and Computers, 28(2), 197-202, 1996.

[126] Landauer, "*On the computational basis of learning and cognition: Arguments from LSA*", The Psychology of Learning and Motivation, 41, 1-63, 2002.

[127] S. Leach, "*Singular Value Decomposition (SVD) - A Primer*", Draft Version, Department of Computer Science, Brown University.

[128] Kirk Baker, "*Singular Value Decomposition Tutorial*", March 29, 2005.

[129] Sileshi, M., Gamback, B., "*Evaluating Clustering Algorithms: Cluster Quality and Feature Selection in Content-Based Image Clustering*", Computer Science and Information Engineering, 2009 WRI World Congress, Volume: 6, Page(s): 435 – 441, 2009.

[130] Kai Li, Lan Wang, Lifeng Hao, "*Comparison of Cluster Ensembles Methods Based on Hierarchical Clustering*", Computational Intelligence and Natural Computing, 2009. CINC'09. International IEEE Conference, Volume: 1, Page(s): 499-502, 2009.

[131] C. Ding, X. He, "*Cluster merging and splitting in hierarchical clustering algorithms*", 2nd International IEEE Conference Data Mining, Page(s): 139-146, Maebashi, Japan, 2002.

[132] B. S. Everitt, S. Landau, and M. Leese, "*Cluster Analysis*", London: Arnold, 2001.

[133] Sokal R.R., Michener C.D., "*A Statistical Method for Evaluating Systematic Relationships*", University of Kansas Science Bulletin, Vol. 38, Page(s): 1409-1438, 1958.

[134] Gibas C., Jambeck P., "*Developing bioinformatics computer skills*", Publisher: O'Reilly Media, April 2001.

[135] Bockenhauer H. J., and Bongartz D., "*Algorithmic aspects of bioinformatics*", 2007.

[136] Chung-Horng Lung, Xia Xu, Marzia Zaman, Anand Srinivasan, "*Program restructuring using clustering techniques*", Journal of Systems and Software, Volume 79, Issue 9, Pages 1261–1279, September 2006.

[137] Sokal, R. R., and Sneath, P. H. A., "*Principles of numerical taxonomy*", San Francisco, CA: W. H. Freeman, 359p, 1963.

[138] Beckett D., "*Redland RDF Library*", http://librdf.org/docs/api/index.html, [Last accessed: 10/12/2011], 2011a.

[139] David Beckett, "*The Design and Implementation of the Redland RDF Application Framework*", 10th international conference on World Wide Web, Institute for Learning and Research Technology, University of Bristol, 2001.

[140] Oracle, BerkeleyDB 11g Release 2 (Library versión 11.2.5.1) Documentation, August 2010.

[141] Beckett D., "*Raptor RDF Parser Library*", http://librdf.org/raptor/libraptor.html, [Last accessed: 10/12/2011], 2011b.

[142] Beckett D., "*Rasqal RDF Query Library*", http://librdf.org/rasqal/librasqal.html, [Last accessed: 10/12/2011], 2011c.

[143] T.-L. Tseng, W.-Y. Liang, C.-C. Huang, T.-Y. Chian, "*Applying genetic algorithm for the development of the components-based embedded system*", Computer Standards & Interfaces, 27 (6), 621-635, 2005.

[144] C. Arth, H. Bischof, C. Leistner, "*TRICam - An Embedded Platform for Remote Traffic Surveillance*", 2006 Conference on Computer Vision and Pattern Recognition Workshop, Vol. 17-22, 2006.

[145] S. L. Toral, M. R: Martínez-Torres, F. Barrero, F. Cortés, "*An empirical study of the driving forces behind online communities*", Internet Research, 19 (4), 378-392, 2009.

[146] S. L. Toral, M. Vargas, F. Barrero, "*Embedded Multimedia Processors for Road-Traffic Parameter Estimation*", IEEE Computer, Volume 42, Issue 12, pages 61-68, 2009.

[147] S. Toral, M. Vargas, F. Barrero, M. G. Ortega, "*Improved Sigma-Delta Background Estimation for Vehicle Detection*", Electronics Letters, 45 (1), 32-34, 2009.

[148] J. M. Milla, S. L. Toral, M. Vargas, F. Barrero, "*A Dual-Rate Background Subtraction Approach for Estimating Traffic Queue Parameters in Urban Scenes*", IET Intelligent Transport Systems, in press, doi: 10.1049/iet-its.2012.0020, 2013.

[149] M. Piccardi, "*Background subtraction techniques: a review*", In IEEE International Conference on. Systems, Man and Cybernetics, 2004, Vol. 4, pp. 3099-3104, 2004.

[150] A. Prati, I. Mikic, M.M. Trivedi, R. Cucchiara, "*Detecting moving shadows: algorithms and evaluation*", Transactions on Pattern Analysis and Machine Intelligence, 25 (7), 918-923, 2003.

[151] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, "*Detecting moving objects, ghosts, and shadows in video streams*", Transactions on Pattern Analysis and Machine Intelligence, 25 (10), 1337-1342, 2003.

[152] J.C.S. Jacques Jr, C.R. Jung, S.R. Musse, "*Background subtraction and shadow detection in grayscale video sequences*", Proc. of the XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'05), pp. 189-196, 2005.

[153] Rapid-I (2012), Interactive Design. Products: RapidMiner, http://rapid-i.com.

[154] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, T. Euler, *"YALE: Rapid prototyping for complex data mining tasks"*, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06), 2006.

[155] R. C. Deonier, S. Tavare, M. S. Waterman, "*Computational Genome Analysis: An Introduction*", Statistics for Biology and Health, Springer-Verlag New York Inc., 2007.

[156] Protégé 4.1, free and open source ontology editor and knowledge-base framework, http://protege.stanford.edu, 2011.

[157] Michi Henning, Steve Vinoski, "*Advanced CORBA® Programming with C++*", Part of the Addison Wesley Professional Computing Series, Feb 17, 1999.

[158] D. Gregor, S.L. Toral, T. Ariza, F. Barrero, "*An Ontology-based Semantic Service for Cooperative Urban Equipments*", Journal of Network and Computer Applications 35 (2012), pp. 2037-2050 DOI information: 10.1016/j.jnca.2012.08.002, Oct 09, 2012.

Listado de Acrónimos

A

ACC	(Adaptive Cruise Control)
ADAS	(Advanced Driver Assistance Systems)
AI	(Artificial Intelligence)
APIs	(Application Programmer Interface)
ARM	(Advanced RISC Machines)

B

B2B	(Business to Business)
BDB	(BerkeleyDB)
BGPs	(Basic Graph Patterns)

C

C/S	(Client/Server)
C2C-CC	(Consorcio de Comunicación Car2Car)
CALM	(Communications Air-interface, Long and Medium range)
CEN DSRC	(Comité Europeo para la Estandarización)
COOPERS	(Co-operative Systems for Intelligent Road Safety)
CORBA	(Common Object Request Broker Architecture)
CPUs	(Central Processing Unit)
CVIS	(Cooperative Vehicle Infrastructure Systems)
CVO	(Commercial Vehicle Operations)

D

DAB	(Digital Audio Broadcasting)
DAML+OIL	(DARPA Agent Markup Language + Ontology Inference Layer)
DCOM	(Distributed Component Object Model)
DD	(Device-Device)
DII	(Dynamic Invocation Interface)
DL	(Description Logic)
DMB	(Digital Multimedia Broadcasting)
DOT	(Deptartment of Transportation)
DPK	(Discrimination of Paragraphs with Keywords)
DRE	(Distributed Real-time and Embedded)
DSI	(Dynamic Skeleton Interface)

DSP	(Digital Signal Processors)
DSRC	(Dedicated Short Range Communication)
DSS	(Decision Support Systems)
DVB-H	(DVB-Handheld)
DVB-T	(Digital Video Broadcasting-Terrestrial)

E

ESP	(Electronic Stability Program)
ETSI	(European Telecommunications Standards Institute)

F

FOL	(First Order Logic)

G

GIOP	(General Inter-ORB Protocol)
GPS	(Global Positioning System)
GSM	(Global System for Mobile Communications)

H

HACM	(Hierarchical Agglomerative Clustering Methods)
HS	(Hardware/Software)

I

IDL	(Interface Definition Language)
IIOP	(Internet-IOP)
IOR	(Interoperable Object Reference)
IR	(Information Recovery)
IRWDP	(Information Retrieval with Weighted Data in Paragraphs)
IT	(Information Technology)
ITI	(Intelligent Transportation Infraestructure)
ITS	(Intelligent Transportation Systems)

J

JVM	(Java Virtual Machine)

K

KB	(Knowledge Base)

L

LoD	(Level of Detail)
LSA	(Latent Semantic Analysis)
LSI	(Latent Semantic Indexing)

M

MAS	(Multi Agent Systems)
MIPS	(Microprocessor without Interlocked Pipeline Stages)
MTE	(Middleware Technology Evaluation)

N

NLP	(Natural Language Processing)
NoW	(Network on Wheels)

O

OMG	(Object Management Group)
ORB	(Object Request Broker)
OWL	(Web Ontology Language)

P

PID	(Process ID)
POA	(Portable Object Adapter)

Q

QGS	(Quasi-Gold Standard)
QoS	(Quality of Service)

R

RDF	(Resource Description Framework)
RDFS	(RDF-Schema)
RDQL	(RDF Data Query Language)
RITA	(The Research and Innovative Technology Administration)
RMI	(Remote Method Invocation)
RPC	(Remote Procedure Call)
RQL	(RDF Query Language)
RTT	(Round Trip Time)

S

SAFESPOT	(Cooperative Systems for Road Safety)
SAP	(Single Access Point)
SLR	(Systematic Literature Review)
SO	(Sistema Operativo)
SOA	(Service Oriented Architecture)
SOAP	(Simple Object Access Protocol)
SoC	(System-on-Chip)
SPARQL	(SPARQL Protocol and RDF Query Language)
SS	(Semantic Service)
SVD	(Singular Value Decomposition)

T

TIC	(Tecnologías de Información y Comunicación)

U

UE	(Unión Europea)
UPGMA	(Unweighted Pair Group Method with Arithmetic)
URI	(Uniform Resource Identifier)
URL	(Uniform Resource Locator)
URNs	(Uniform Resource Names)
UTCS	(Urban Trafic Control System)

V

V2I	(Vehicle to Infrastructure)
V2V	(Vehicle to Vehicle)
VMS	(Variable Message Sings)

W

W3C	(World Wide Web Consortium)
WILLWARN	(Wireless Local Danger Warning)
WLAN	(Wireless LAN)
WPGMA	(Weighted Pair Group Method with Arithmetic Mean)
WS	(Web Services)
WSDL	(Web Services Description Language)

X

XML	(Extensible Markup Language)