ARQUITECTURA SOFTWARE MULTISERVICIO APLICADA AL TRANSPORTE POR CARRETERA

JOSÉ SANTA LOZANO[1,2]
Ingeniero en Informática. Becario FPU. Universidad de Murcia.

BENITO ÚBEDA MIÑARRO[2]
Ingeniero en Telecomunicaciones y Doctor en Informática. Profesor Titular. Universidad de Murcia.

RAFAEL TOLEDO MOREO[2]
Ingeniero Industrial y Doctor en Informática. Profesor Colaborador. Universidad de Murcia.

CRISTINA SOTOMAYOR MARTÍNEZ[2]
Ingeniero en Informática. Becario de Investigación. Universidad de Murcia

RESUMEN: Actualmente, existe una creciente demanda de servicios de a bordo en vehículos que hace pensar en un cambio de un modelo de negocio hasta ahora basado en el despliegue de nuevas funcionalidades a través de la instalación de nuevo hardware. Los nuevos servicios deberían ser creados usando un modelo de desarrollo modular en donde cada nueva funcionalidad se pudiese instalar sin grandes costes asociados. Por este motivo, este trabajo muestra una arquitectura extensible para el desarrollo de servicios para vehículos implementados en software. Se presenta un modelo multi-capa sobre un framework OSGi (*Open Services Gateway initiative)* sostenido por un ordenador de propósito general. Una amplia gama de sensores han sido instalados sobre un vehículo prototipo, en donde ha sido instalada una implementación de referencia de nuestro diseño.

---

[1] Autor que realizará la presentación de la comunicación.

[2] Departamento de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia. Campus de Espinardo, 30071 Murcia, España.

Diversos servicios ejemplifican la generalidad del diseño, abarcando servicios basados en localización (LBS), multimedia, y sistemas de ayuda a la conducción (ADAS).

1 INTRODUCCIÓN

Debido al creciente interés que la sociedad actual tiene en las nuevas tecnologías, cantidad de productos en el ámbito de la informática y de las comunicaciones están emergiendo en ámbitos hasta ahora inexplorados. De entre éstos, el vehículo presenta un marco perfecto para incluir muchas de las funcionalidades hasta ahora disponibles en el trabajo o en el hogar. Sin embargo, esta expansión necesita de un soporte tanto hardware como software adaptado a los requisitos del mercado y a las necesidades del usuario.

Hasta el momento, la cantidad de servicios que los pasajeros pueden usar en el vehículo requieren de un importante despliegue hardware. Cada nueva funcionalidad suele estar implementada como un nuevo dispositivo. Aunque este procedimiento ha sido plausible hasta la fecha, la cantidad de servicios de a bordo está aumentando considerablemente en los nuevos vehículos. Además del reproductor de CD/MP3, el manos libres Bluetooth, y el navegador GPS, cantidad de nuevos servicios se están tomando en consideración, como sistemas de alerta por colisión, dispositivos anti-radares, etc. El modelo de despliegue actual de servicios no es en absoluto escalable, y es en este punto donde las computadoras de propósito general comienzan a ser consideradas como una mejor opción que las dedicadas. La inclusión de nuevos servicios como software ejecutable en una plataforma hardware común presenta diversas ventajas. Primero, el modelo de negocio sufre un cambio radical debido a que tanto la fase de instalación como, sobre todo, las labores de actualización no requieren inversión en nuevo hardware. Al mismo tiempo, el usuario puede estar ante una interfaz común de gestión de los servicios de a bordo, que facilite en gran medida el control de la funcionalidad de a bordo del vehículo.

Existen multitud de factores a tener en cuenta en la introducción de una arquitectura como la mostrada anteriormente. Un vehículo implica un lugar peculiar repleto de cuestiones relativas a la interacción con el usuario. Debido a esto, tenemos que considerar no sólo los requerimientos hardware, sino también los relativos a la interfaz. Como se muestra en [1], la instalación de un ordenador de a bordo en un vehículo presenta una cuestión delicada

donde se deben tener en cuenta las condiciones físicas de colocación y de alimentación. Por otro lado, la interfaz de usuario está sujeta a restricciones legales. En [2] se presentan algunas cuestiones sobre el uso de dispositivos electrónicos en vehículos. Como se detalla en este trabajo, aunque la legislación de diversos países está recientemente concienciada sobre los problemas de usar teléfonos móviles, no existe un acuerdo claro sobre cómo tratar el uso de otros muchos sistemas de a bordo. En cualquier caso, lo que sí está claramente especificado es que el conductor debe ser considerado como una persona que debe poder controlar en todo momento su vehículo. Así pues, se deben tener en cuenta los problemas de seguridad que pueden acarrear la instalación de nuevos servicios en el vehículo. Los servicios orientados al conductor son los que deben ser especialmente estudiados, frente a los orientados a los pasajeros, que pueden tener condiciones de interfaz más relajadas.

Algo que se debe tener en cuenta para introducir una arquitectura de servicios realmente integrada en el vehículo es la disposición de un hardware adecuado. Así pues, diversos dispositivos serán esenciales si estamos interesados en servicios basados en la localización (LBS), tales como un sensor GNSS (*Global Navigation Satellite System*) y un módem de comunicación celular. En [3] y [4] se muestran dos arquitecturas de desarrollo válidas para la implementación de software orientado al contexto de circulación. Los sensores INS (*Inertial Navigation System)*, los captores odométricos, y los receptores de posicionamiento GPS son factores clave en dichos sistemas, donde un software de control autónomo embebido fusiona toda la información recibida de todos ellos para tomar una decisión sobre el movimiento del vehículo.

Además del hardware apropiado, es necesaria una plataforma software que permita el despliegue de servicios. Esta plataforma estará localizada en el ordenador de a bordo, y debería tener en cuenta los siguientes requisitos:

- Modularidad. Los nuevos servicios deberían ser implementados como composición de módulos creados previamente.
- Portabilidad. Los servicios y, si es posible, la propia plataforma no deberían estar diseñados para un sistema operativo concreto.
- Facilidad de despliegue. La instalación y actualización de los servicios debe ser un proceso sencillo y eficiente.

La *Open Services Gateway initiative* [5] presenta una plataforma modular con una gran cantidad de características acordes a estos requerimientos, frente a otras soluciones como

Jini o SLP (*Service Location Protocol*). La diferencia principal entre OSGi y estas últimas se centra en el campo de aplicación. OSGi está orientada al despliegue de servicios sobre una pasarela. Por el contrario, Jini, y SLP sobre todo, son válidas para servicios distribuidos sobre una red.

Nuestro trabajo ha estado centrado en el diseño y desarrollo de una arquitectura extensible para servicios, basada en computadores de propósito general. Usando un ordenador de a bordo como pasarela OSGi de servicios, los problemas de espacio en el habitáculo son resueltos. Al mismo tiempo, nuestra propuesta muestra un marco de despliegue de servicios basado en capas, el cual promueve la reusabilidad y el desarrollo modular. El sistema completo ha sido enriquecido con una gran cantidad de servicios que muestran la validez de la solución propuesta. La arquitectura ha sido incluida sobre un vehículo ampliamente sensorizado, lo que nos ha permitido crear varios servicios basados en el contexto.

La literatura sobre conceptos similares es bastante limitada, principalmente debido a que la mayor cantidad de aportaciones vienen de compañías privadas. En [6] se presenta un framework para desarrollar software de a bordo para vehículos orientado al usuario. Nuestra propuesta, a diferencia de lo presentada aquí, no muestra una arquitectura basada en APIs de programación para aplicaciones finales. Realmente, estamos interesados en dar las facilidades necesarias al programador para desarrollar aplicaciones visuales, pero también para crear drivers de acceso a dispositivos físicos, intercambiables durante el ciclo de vida del vehículo. Cada nueva entidad software será añadida de forma modular dentro de la arquitectura. En [7] se encuentra otro ejemplo de solución para el desarrollo de software de a bordo. Éste está centrado en la comunicación entre servicios locales al vehículo, así como entre servicios localizados en el vehículo y en el lado de la carretera. Jini y JXTA presentan las bases para la implementación de servicios, aunque el sistema constituye una plataforma mucho menos integrada que nuestra propuesta, centrada en el concepto de software embebido para el vehículo. En esta línea, [8] muestra un framework realmente ligado a una unidad de a bordo. Éste es desarrollado como una jerarquía de clases .NET sobre la que los servicios son desarrollados. Aunque la idea de construir módulos software es similar a nuestra concepción, la flexibilidad provista por OSGi no es comparable a una tecnología de programación, en este caso .NET, que siempre presenta un alto acoplamiento en los desarrollos.

El resto del artículo está organizado como sigue. En la sección 2 se presentará la arquitectura completa del sistema. La sección 3 incluye una descripción de los servicios desarrollados, junto con una explicación del vehículo prototipo que se ha preparado. Finalmente, la sección 4 contiene la conclusión del trabajo y las futuras vías de investigación en este campo.

2 UNA ARQUITECTURA EXTENSIBLE PARA EL DESARROLLO DE SERVICIOS DE A BORDO BASADA EN OSGI

Como se ha mencionado previamente, OSGi se presenta como un contenedor adecuado para el conjunto de servicios que pueden ser implementados en el vehículo como software. Inicialmente, la concepción de OSGi estuvo centrada en la creación de pasarelas residenciales en las que el software que hacía la casa “inteligente” era instalado. En [9] se puede encontrar un buen ejemplo en donde OSGi es usado como base de desarrollo para un prototipo de casa inteligente, orientado a la localización de sus habitantes para adecuar la electrónica de cada una de las habitaciones. Sin embargo, las ventajas de OSGi se han extendido a otros campos. El entorno del vehículo es, por ejemplo, uno de estos nuevos lugares en donde OSGi presenta una buena integración. Aquí será el ordenador de a bordo (OBU) el que sea considerado como la pasarela de servicios.

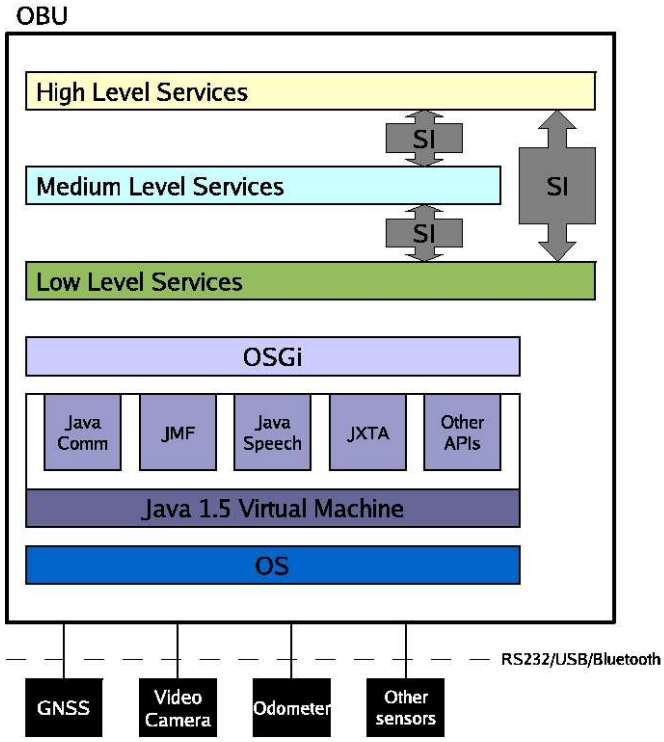

Fig. 1 Arquitectura del OBU para el despliegue de servicios

De esta manera, OSGi es la plataforma sobre la que las arquitectura que se propone en este trabajo es situado. La Fig. 1 muestra el sistema diseñado para crear servicios sobre el OBU. Todos los sensores incluidos en el vehículo se conectan al OBU a través de un medio de comunicación que puede alternar entre conexiones inalámbricas o con cable. El ordenador considerado es un PC ordinario con un sistema operativo no prefijado que tiene instalado una máquina virtual de Java (JVM). Varios APIs de programación se sitúan sobre esta base Java, donde las más importantes son *Java Comm, Java Media Framework* y *Java Speech*. Java Comm es usado en las comunicaciones vía puerto serie, tal como la conexión hacia el sensor GNSS y hacia los captores odométricos; Java Media Framework is usado en el desarrollo de software multimedia; y Java Speech provee de un API con funciones de síntesis de voz. Otras APIs son usadas en la programación gráfica y matemática en Java. OSGi está localizada sobre la base Java, haciendo al OBU capaz de contener diferentes servicios. De hecho, tal y como se explica en [10], las entidades software instaladas sobre OSGi son llamadas *bundles*, los cuales pueden ofrecer servicios.

De vuelta al diagrama, el conjunto de elementos visibles sobre la capa OSGi son los servicios implementados en la plataforma. Éstos están clasificados de acuerdo con su nivel

de abstracción, por lo que están divididos en tres capas: bajo nivel, nivel medio, y alto nivel. Los servicios de bajo nivel constituyen el software necesario para el acceso a los diversos dispositivos instalados en el vehículo. Así pues, estos servicios pueden ser considerados como *drivers* para el hardware del vehículo. Los servicios de nivel medio actúan como *middleware* entre los incluidos en las capas superior e inferior. En este sentido, esta capa realiza tareas de transformación y de adaptación de la información, mejorando la funcionalidad ofrecida por los servicios de bajo nivel. Finalmente, los servicios de alto nivel comprenden al software, implementado como bundles, que presenta una interfaz directa con el usuario.

La estructura jerárquica de servicios descrita tiene un doble propósito. Primero, la creación de nuevos servicios es facilitada por una programación modular. Si alguna funcionalidad debe ser usada, o se espera que lo sea, por alguna nueva aplicación, ésta puede ser encapsulada como un servicio. Por otro lado, usando esta metodología de desarrollo, los problemas de sincronización en el acceso a los dispositivos instalados en el vehículo pueden ser resueltos. Esto es, no solamente el software que se usa como driver es reutilizado, sino que la implementación de servicios de bajo nivel sincroniza el uso del hardware. Este último hecho se ve de manifiesto en los sensores usados ampliamente por los servicios de a bordo, como puede ser el GNSS. Debido a que una gran cantidad de servicios puede requerir información de posicionamiento, el acceso al sensor debe estar coordinado. Incluso, una ventaja añadida viene con la posibilidad de usar técnicas de *caching* de información, por lo que es posible que los servicios puedan servir peticiones que deberían circular hacia los dispositivos reales, usando información almacenada temporalmente.

La comunicación entre capas se lleva a cabo por *interfaces de servicio* (SI). Cada capa define un conjunto de interfaces de servicio que indican las funcionalidades disponibles. Una SI es realmente una interfaz Java que puede estar implementada por uno o más bundles OSGi para ofrecer determinados servicios. Si un bundle situado en las capas superiores necesita de una funcionalidad provista por una SI, éste lanza una consulta al framework OSGi usando la SI como parámetro. Como resultado, el framework devuelve la implementación del conjunto de servicios que desarrollan la funcionalidad requerida.

Existen dos bundles situados en la arquitectura presentada que realizan una tarea bien definida en el sistema. Éstos son *Policy Manager* y *Service Manager*. Éste último has sido desarrollado como un bundle de alto nivel, tal y como puede ser observado en el color usado en la Fig. 1. Su interfaz gráfica muestra el conjunto de servicios que se han instalado en el sistema, ordenados por su nivel de abstracción. A partir de aquí, el usuario puede habilitar, deshabilitar o actualizar los servicios que se encuentran instalados.

Policy Manager ha sido creado como un servicio de nivel medio, ya que contiene un conjunto de funcionalidades usadas de forma transparente por los servicios de alto nivel. Gracias a Policy Manager, los servicios de nivel de usuario no son lanzados a ejecución cuando alguna restricción de ente un conjunto configurado se cumple. Por ejemplo, se podrían implementar interfaces gráficas para mantener la atención del usuario en la carretera cuando el vehículo está en movimiento. La Fig. 2 muestra un diagrama abstracto de las clases que intervienen en el gestor de políticas diseñado. Los servicios de nivel de usuario, implementados como servicios o bundles OSGi, heredan de una clase que incluye la funcionalidad necesaria para hacer que éstos estén restringidos por el sistema de políticas. De esta manera el servicio Policy Manager envía una notificación cuándo una restricción se cumple, y la propia implementación del servicio en cuestión (normalmente de nivel de usuario) decide qué hacer. Para realizar esta tarea, Policy Manager se encuentra continuamente funcionando y chequeando las políticas configuradas, como puede ser una comprobación sobre si el vehículo está en movimiento, si alcanza una velocidad determinada, o si se detecta humedad en el limpiaparabrisas. Policy Manager hace uso también de los servicios de bajo nivel que necesita para acceder a los sensores del vehículo, tal como la odometría o el receptor GNSS.

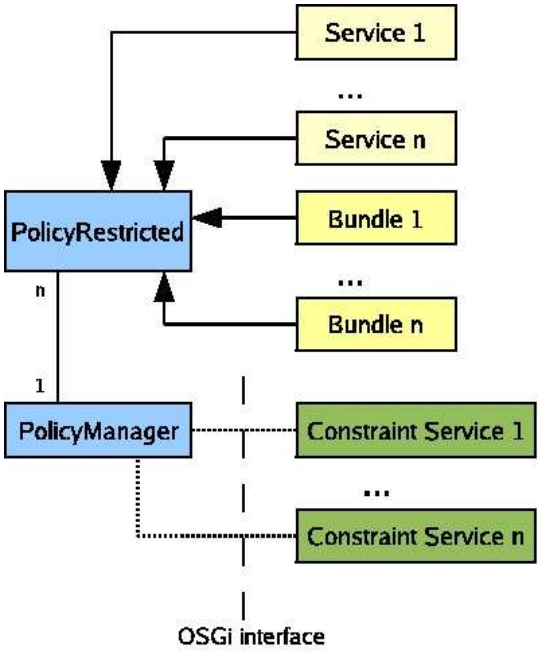

Fig. 2 Subsistema de políticas de ejecución

3 SERVICIOS DE NUEVA GENERACIÓN PARA VEHÍCULOS

La plataforma descrita ha sido implementada y testeada sobre un sistema real. Además, diversos servicios pertenecientes a cada nivel de abstracción de la arquitectura han sido desarrollados para demostrar la viabilidad de la arquitectura presentada en el artículo.

3.1 Detalles sobre el hardware del prototipo

La plataforma hardware considerada está basada en un prototipo de un vehículo ampliamente sensorizado usado en la Universidad de Murcia [3] en varios proyectos de investigación concernientes a la navegación autónoma en vehículos, a través de sensores GNSS e INS. Gracias a los sensores instalados en el vehículo, es posible crear servicios dependientes del estado actual del móvil y del contexto de circulación. Entre todos ellos, son destacables los captores odométricos situados en cada rueda, y el receptor GNSS. La odometría es útil para monitorizar el movimiento del vehículo, mientras que el sensor GPS es una pieza fundamental en las labores de localización del móvil.

Fig. 3 Vehículo usado en el desarrollo del prototipo

El vehículo en cuestión se muestra en la Fig. 3. Éste incluye un OBU constituido por un PC de pequeñas dimensiones, un *single board computer* (SBC) de arquitectura VIA. Una pantalla táctil permite la interacción con el usuario mediante la pulsación directa sobre ella. La típica interfaz mediante ratón y teclado también están disponibles mediante un soporte situado debajo del salpicadero. El sistema operativo instalado en el OBU es un Linux Fedora Core 4, y se ha usado la máquina virtual de Java 1.5. Se han instalado dos frameworks OSGi, Knopflerfish [11] y Oscar [12]. Ambos son implementaciones de código abierto de la Release 3 de OSGi [5], y permiten ejecutar correctamente la implementación de referencia realizada. Con respecto a las comunicaciones, el vehículo se ha provisto del hardware necesario para conectarse a varias tecnologías de red, tales como Bluetooth, 802.11 y GPRS/UMTS, cubriendo todas las necesidades de comunicación.

3.2 Servicios implementados

Tal y como se ha dicho anteriormente, existe un servicio de alto nivel que se implementa sobre la arquitectura propuesta, con el objetivo de permitir al usuario manejar el resto de los servicios de la plataforma. Este es Service Manager, cuya implementación de referencia se muestra en la Fig. 4. Esta aplicación muestra todos los servicios instalados actualmente en el framework OSGi. El usuario puede iniciar, parar o actualizar cada uno de ellos de manera

sencilla. Los servicios que se desarrollan para estar incluidos en la plataforma tienen unas características especiales en su archivo de despliegue JAR, distinguiéndolos del resto de los instalados en el framework OSGi. Todos los servicios están dispuestos en la pantalla principal de Service Manager de acuerdo con su nivel de abstracción.

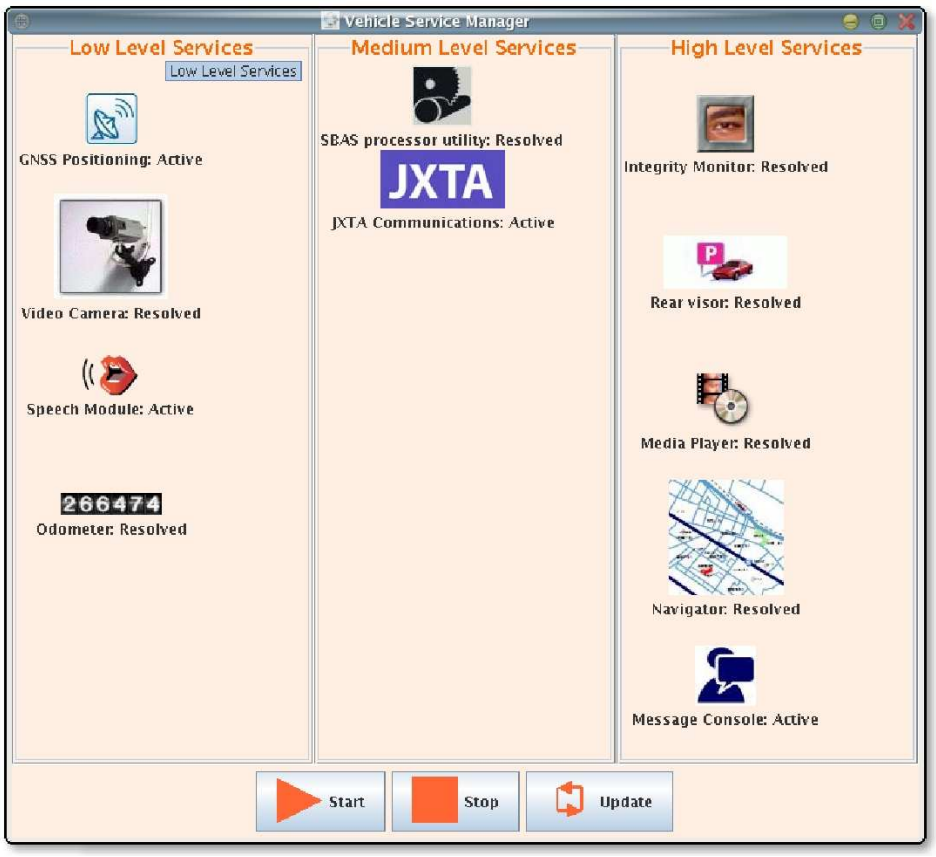

Fig. 4 Servicio de gestión de la plataforma

Tal y como se puede observar en la Fig. 4, los servicios de bajo nivel desarrollados hasta la fecha son:

- *GNSS Positioning*. Éste es un servicio genérico de acceso al sensor de posicionamiento. Diversos receptores GPS son soportados, y el servicio está diseñado a través de un diseño extensible que permite la inclusión de nuevos sensores sin un gran esfuerzo.
- *Video Camera*. Este servicio es usado para establecer una conexión con una cámara situada en la parte trasera del vehículo. La lógica del servicio usa Java Media Framework para obtener la imagen recibida desde la cámara.

- *Speech Module*. Éste encapsula un sintetizador de voz disponible para el resto de los servicios que requieran emitir alertas habladas. Éste sistema permite que el conductor no tenga que desviar la vista de la carretera. Éste software ha sido implementado usando FreeTTS, que es una implementación parcial del API Java Speech.
- *Odometer*. Este servicio provee de información desde los sensores de odometría. La odometría del vehículo está compuesta por cuatro captores que obtienen la velocidad de rotación de cada rueda, estimando la velocidad del móvil.

En la capa intermedia se incluye un servicio middleware llamado SBAS Processor Utility que contiene diversas utilidades relativas a los sistemas satelitales de aumento de la señal (SBAS) emitida por los GNSS. En este sentido, la información SBAS recibida desde el propio sensor GNSS, o a través de Internet mediante SISNeT [13], es usada para propósitos de corrección y de cálculos de integridad de la posición. Las correcciones son adaptadas a un formato más común, como es RTCM, y se calcula el factor de integridad HPL (*Horizontal Protection Level*) [14]. La red celular es necesaria para conectar con el servidor SISNeT cuando la señal SBAS no está disponible. El servicio JXTA Communications ofrece funciones de comunicación al resto de los servicios del sistema. Así, es posible establecer un paso de mensajes sencillo entre vehículos, o ente el vehículo y la infraestructura.

Las aplicaciones de alto nivel tienen una relación directa con el usuario. Hasta el momento, el conjunto de servicios implementados en la capa de alto nivel son los siguientes:

- *Integrity Monitor*. Ésta es una aplicación usada para monitorizar la integridad de la posición emitida por el sistema de posicionamiento. Para este propósito, el servicio usa el servicio SBAS Processor Utility, además del Speech Module para alertar al conductor cuando el factor de integridad excede un límite prefijado.
- *Rear Visor*. Con esta aplicación, el usuario puede ver desde la parte trasera del vehículo, gracias al servicio Video Camera. El propósito de esta funcionalidad es mejorar la visión del conductor en el aparcamiento, sobre todo.
- *Media Player*. Éste es un reproductor multmedia con el que se pueden abrir diversos formatos de video y audio. Tal y como demuestra este servicio, es destacable observar cómo los reproductores de video, e incluso los de CD y la radio, podrían reemplazarse por software instalable en el OBU.

- *Navigator*. Este programa dispone de funcionalidad de navegación y de peaje electrónico, gracias al uso de un sistema de información geográfica (GIS). La aplicación depende del servicio GNSS Positioning, ya que la posición del vehículo es necesaria para localizarlo en la cartografía. El servicio SBAS Processor Utility es usado para calcular la integridad de la posición, y el Speech Module es útil para avisar al usuario acerca de eventos de interés.
- *Message Console*. Esta aplicación hace uso del servicio JXTA Communications para crear un sistema de avisos de seguridad en carretera. Los mensajes que se emiten pueden recibirlos tanto el resto de los vehículos cercanos a la incidencia, como las entidades situadas al lado de la infraestructura.

Los servicios de alto nivel están enlazados con el sistema de políticas explicado previamente. Todos ellos deben implementar las tareas necesarias que deben realizarse cuando el gestor de políticas (Policy Manager) detecta que algunas de las restricciones se cumple. Por el momento, se ha considerado la restricción de movimiento. En concreto, Policy Manager usa el servicio Odometer para chequear si la velocidad del vehículo es mayor que un cierto umbral. Si es éste el caso, todos los servicios de alto nivel que están registrados con esta restricción serán notificados sobre el evento para que realicen las acciones oportunas. Por ejemplo, el servicio Rear Visor oculta la imagen de la cámara de video, notificando la situación al usuario.

La Fig. 5 muestra un ejemplo de ejecución del servicio Navigator. La pantalla ejemplifica una situación en la que el vehículo circula por el Campus de Espinardo de la Universidad de Murcia. Como puede observarse también, la ventana indica un mensaje de alerta debido a que el HPL se encuentra fuera de los límites definidos. Aunque el programa emite este evento gráficamente, una alerta hablada es realizada para que el conductor no cometa ninguna imprudencia, desviando la vista de la carretera. El resto de la información incluida en la ventana comprende a la posición actual, la distancia hasta la calle más cercana, y la información relativa a ésta.

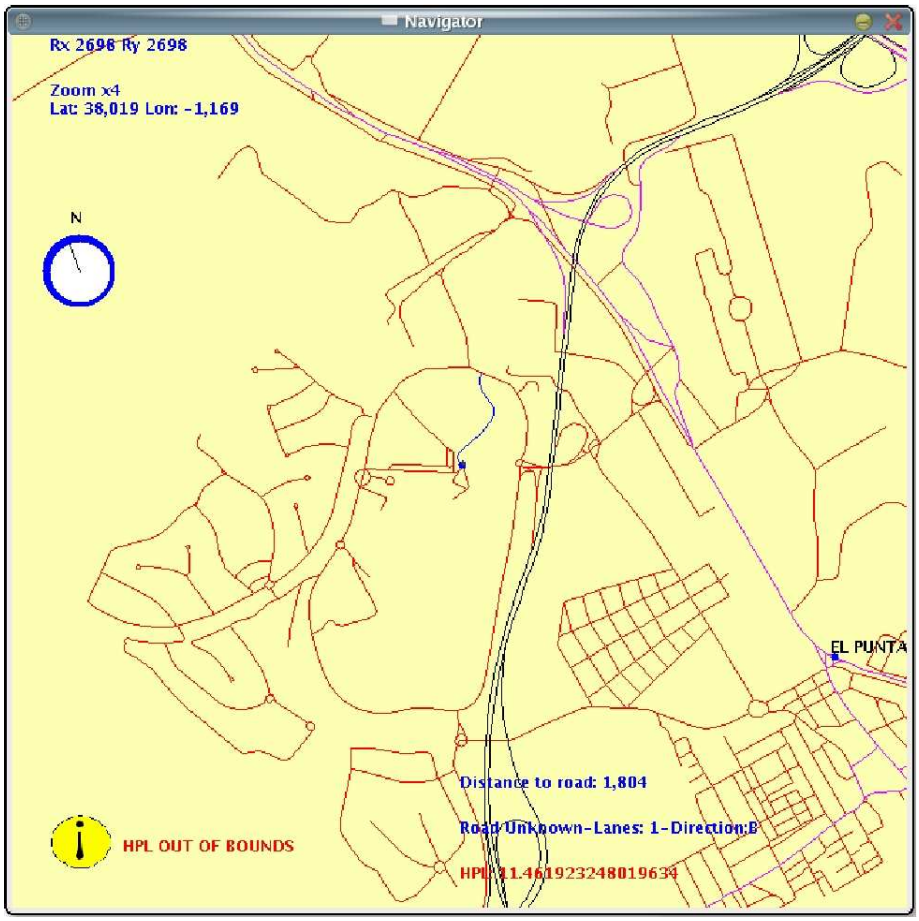

Fig. 5 Servicio de navegación

4 CONCLUSIONES

La arquitectura propuesta incluye una plataforma basada en un ordenador de propósito general con la habilidad de soportar servicios de valor añadido para el conductor y los pasajeros. OSGi, después de mostrar sus ventajas en el entorno del hogar, es considerado como un elemento clave en el desarrollo de funcionalidad software para vehículos. Debido a esto, un framework OSGi has sido usado para soportar todos los servicios de a bordo. Éstos han sido clasificados en niveles de abstracción, permitiendo la creación de nuevos servicios de una manera modular. Un vehículo real provisto de diversos sensores da fe de la viabilidad del diseño expuesto, facilitando la creación de servicios orientados al contexto.

El conjunto de servicios implementados muestra el carácter genérico de la arquitectura propuesta, así como los diferentes ámbitos que cubre el sistema en el espectro de la funcionalidad de a bordo para los ocupantes del vehículo. En este sentido, no solamente se han tratado los servicios basados en la localización, sino también los multimedia y los de asistencia a la conducción. Los servicios de navegación y de monitorización de la integridad desarrollados en el prototipo son considerados dentro del grupo de LBS, mientras que el reproductor multimedia y el visor trasero caben dentro de los servicios multimedia y de asistencia a la conducción, respectivamente.

La creciente implantación de funcionalidades de a bordo en los vehículos de nueva generación justifica la implementación de sistemas como el presentado en este artículo, con el objetivo de evitar una excesiva proliferación de dispositivos en el habitáculo. Es por esto que, añadir servicios implementados como software embebido en un ordenador compartido de propósito general, puede jugar un papel clave en la futura fabricación de vehículos.

5 AGRADECIMIENTOS

Los autores agradecen el apoyo dado por el Ministerio de Educación y Ciencia en las labores de investigación, bajo la beca AP2005-1437 en el marco del programa FPU, y a la Agencia Espacial Europea (ESA) por soporte económico bajo el proyecto GIROADS 332599. Un especial agradecimiento también para el Ministerio de Fomento por su continuo apoyo en las labores sobre ITS.

6 BIBLIOGRAFÍA

[1] CRAIG SIMONDS (2003). “Software for the Next-Generation Automobile”. IT Professional, November/December 2003, pp 7-11.

[2] WARD VANLAAR (2005). “Legislation, Regulation and Enforcement for Dealing with Distracted Driving in Europe”. International Conference on Distracted Driving, Toronto.

[3] SKARMETA A.G., MARTÍNEZ H., ZAMORA M.A., ÚBEDA B., GÓMEZ F.C, TOMÁS L.M. (2002). “MIMICS: Exploiting Satellite Technology for an Autonomous Convoy”. IEEE Intelligent Systems. N IV. V. vol. 17, pp. 85-89.

[4] MASSAKI WADA, XUCHU MAO, HIDEKI HASHIMOTO, MAMI MIZUTANI, MASAKA SAITO (2004). “iCAN: Pursuing Technology for Near-Future ITS”. IEEE Intelligent Systems, January/February 2004, pp 18-23.

[5] OSGi Alliance. OSGi web site. http://www.osgi.org

[6] E. C. NELSON, K. V. PRASAD, V. RASIN, C. J. SIMONDS (2004). “An embedded architectural framework for interaction between automobiles and consumer devices”. 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS’04)

[7] LUCIANO BARESI, CARLO GHEZZI, ANTONIO MIELE, MATTEO MIRAZ, ANDREA NAGGI, FILIPPO PACIFICI (2005). “Hibrid service-oriented architectures: a casestudy in the automotive domain”. 5th International Workshop on Software Engineering and Middleware (SEM’05), Lisbon.

[8] ZOLTÁN BENEDEK (2004). “A Framework Built in .NET for Embedded and Mobile Navigation Systems”. 2nd International Workshop on .NET Technologies, Czech Republic. May 2004.

[9] SUMI HELAL, WILLIAM MANN, HICHAM EL-ZABADANI, JEFFREY KING, YOUSSEF KADDOURA, ERWIN JANSEN (2005). “The Gator Tech Smart House: A Programmable Pervasive Space”. Computer. Vol. 38, no. 3, pp. 50-60.

[10] CHOONHWA LEE, DAVID NORDSTEDT, SUMI HELAL (2003). “Enabling Smart Spaces with OSGi”. IEEE Pervasive Computing. Vol. 02, pp 89-94, Jul-Sept.

[11] Knopflerfish OSGi web site. http://www.knopflerfish.org

[12] Oscar OSGi web site. http://oscar.objectweb.org

[13] TORAN-MARTI, F. AND VENTURA-TRAVESET, J. (2004) “The ESA SISNeT Project: Current Status and Future Plans”. European Navigation Conference GNSS 2004, Rotterdam.

[14] JOSÉ SANTA, BENITO ÚBEDA, RAFAEL TOLEDO, ANTONIO F. G. SKARMETA (2006). “Monitoring the position integrity in road transport localization based services”. Vehicular Technology Conference 2006 Fall, Montréal.