A Multiplatform OSGi Based Architecture for Developing Road Vehicle Services

José Santa, Benito Úbeda, Antonio F. G. Skarmeta
Department of Information and Communications Engineering
Computer Science Faculty
University of Murcia
Campus de Espinardo, 30071 Murcia, Spain
Ph: +34-968-398771|+34-968-364607|+34-968-364332
Email: josesanta@dif.um.es|bubeda@um.es|skarmeta@dif.um.es

Abstract— **Nowadays, the growing demand on the implantation of onboard services for road vehicles encourages its development as a part of the current and future vehicles. In this sense, the implementation of new facilities cannot lead to an increase of hardware devices in the driver compartment, because space limitations and the need of an easy non-distracting interface with the user must be considered. For these reasons, it is advisable to have a service architecture suitable for further developments, considering the requirements of extensibility. New services should be created using modules shareable with the rest of applications in the vehicle. For this purpose, this work shows an extensible architecture useful in the software development of road services for vehicles. The solution presented is based on the division of services in different levels of abstraction, according to the underlying hardware. This structure is placed over a general purpose computer. A wide range of sensors has been installed in our test vehicle, allowing the implementation of several context aware services, and proving the feasibility of the proposed solution. A set of examples of location based services (LBS), multimedia services, and advanced driver assistance systems (ADAS) has been developed as described in this paper.**

Index Terms— **Embedded Platforms, Service Composition, OSGi, Location-based Services.**

I. INTRODUCTION

Due to the growing interest that the current society has in new technologies, new products in the fields of information and communication technologies are emerging in new environments still unexploited. In this way, vehicles are a perfect frame for installing a lot of useful functionalities traditionally available at work or home environments. However, this expansion needs a suitable hardware and software support adapted for the market and user demands.

So far, the amount of services that the passengers could use in a vehicle required a large hardware deployment. Each new functionality is implemented in a new device which had to be installed. Up to now, this procedure has been feasible because the amount of onboard services has been limited to a radio, a CD player and, sometimes, a GPS navigator. However, nowadays the new location based services (LBS), just as the ones emplaced in other fields, make this production method a non scalable model. At this point, general purpose computers begin to be considered as a better option than the dedicated ones. The addition of new services as executable software over a common hardware platform shows several advantages. First, the business model suffers a radical change because the service updates do not require new costs in hardware installation. At the same time, the user faces a much more intuitive and common interface with the system, similar to a standard PC.

There are a lot of factors to be taken into account in the introduction of an architecture as the one described in this paper. A vehicle involves a peculiar place full of questions relative to the interaction of services with the user. Due to this, we have to consider not only hardware requirements, but also user restrictions. As [1] shows, physical features of vehicles make the installation of new devices, in this case an onboard computer, a delicate matter where both the vehicle and the computer implications have to be considered. On the other hand, the user interface is subjected to legal considerations. In [2] some issues about the use of electronic devices are described. As stated in that work, although the legislation is recently concerned about the use of mobile phones, there is not a clear agreement about how to deal with new in-vehicle systems. However, the driver is conceived as a person who must be able to control his vehicle anytime, in order to assure the safety of passengers and pedestrians. As a conclusion, it is important to improve the driver and passengers comfort with new services, but considering the safety implications as well. Because of this we must distinguish between driver oriented services and passenger oriented services. In the driver side, location based services and advance driver assistance systems (ADAS) are the main functionalities; while other services like multimedia ones are destined to the rest of occupants of the vehicle.

If we want to introduce a service architecture really integrated in the vehicle it is necessary to have an appropriate hardware support. In other words, several devices will be essential if we are interested in LBS, such as a GNSS (Global Navigation Satellite System) sensor and a cellular communications link. [3] and [4] show two architectures of cars suitable for the development purposes of driving state oriented software. INS (Inertial Navigation System) sensors, odometry captors, and satellite navigation sensors are key factors in both systems, where the embedded autonomous control software merges all the information which comes from them in order to make a decision about the vehicle movement.

In addition to the suited hardware, the proper software plat-

1-4244-0667-6/07/$25.00 ©2007 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2007 proceedings.

form necessary to deploy all the services must be also selected. This platform will be located in the onboard computer, and should take into account next requirements:

- Modularity. New services are implemented as a composition of modules previously created.
- Portability. The services, and the underlying platform if possible, are not limited by one operating system.
- Easiness of deployment. The installation and updating of the services must be an easy and efficient process.

The Open Services Gateway initiative (OSGi) [5] presents a large amount of features according to these requirements, against others extended solutions as Jini and Service Location Protocol (SLP). The main difference between OSGi and these last ones is located in the field in which they are applicable. OSGi points to the development of services in a gateway; on the contrary, Jini, and SLP above all, are suitable for really distributed services along a wide area network.

Our work has been centered in the design and development of an extensible architecture for services based on a general purpose computer, where OSGi is situated. Using an onboard computer as a service gateway, the space problems in the implementation of new functionalities is solved. At the same time, our proposal shows a programming framework for services based on layers, which promotes reusability and modular development. The whole system has been enriched with a large number of services, showing the feasibility of the proposed solution. The architecture has been included in a widely sensorized vehicle, which allows us to create several context aware services. Literature about similar concepts is very limited, mainly due to the major contributions are carried out by private companies. In [6] a framework to develop vehicle onboard software oriented to the user is shown. Our work, in contrast with the approach presented here, does not show an architecture focused in programming APIs for final applications. In fact, we are interested in giving facilities to the programmer to develop visual applications, but also in the implementation of access drivers for physical devices, very changeable during the vehicle life cycle. Every new software entity will be added in a modular way inside the architecture. [7] is another example of a solution for onboard service development. This is centered in the communication between local services inside the vehicle, and between local and external ones, located in other cars or at the road edge. Jini and JXTA are the basis for implementing services, and present a less integrated platform than ours, centered in the concept of embedded software in the onboard computer. [8] shows a framework really integrated in the onboard computer. This is developed as a .NET based class hierarchy, so new services must be programmed against the framework. Although the idea of building software components in that work is similar to the layer based architecture proposed, the flexibility provided by a low level underlying technology (.NET) is not comparable with the one given by OSGi. In addition, the communication between components in this proposal is developed with events, a much more coupled mechanism than the one used in OSGi, based in services.

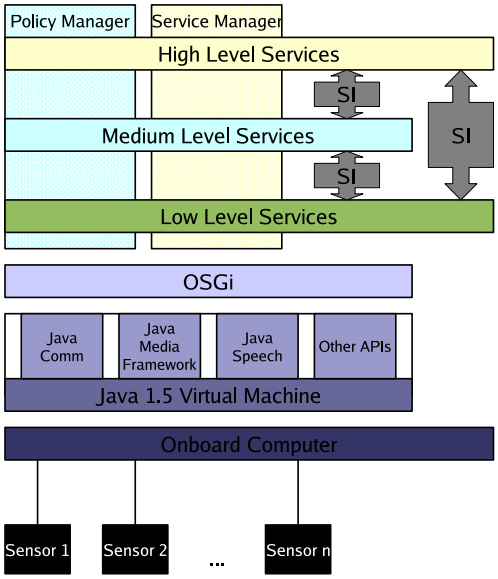

Fig. 1. General infrastructure

The rest of the paper is organized as follows. In Section II the whole architecture of the system is described. Section III includes a description of the services developed, including an explanation of the vehicle used for the deployment. Finally, Section IV contains the conclusions of our work.

II. A FLEXIBLE ARCHITECTURE BASED ON OSGI FOR THE SERVICE DEVELOPMENT

As it has been stated, OSGi is presented as the suitable container for a set of services implemented as PC software. At first, the OSGi conception was based on the creation of residential gateways in which the software which makes the house "intelligent" was installed. In [9] it can be found a good example where OSGi is used as the basis for the development of an intelligent house prototype oriented to the localization of inhabitants. However, the OSGi advantages have extended to some other fields. The vehicle environment is one of these new applications, where the onboard computer can be considered as a service gateway as well.

OSGi is the frame where the proposed architecture in our work is situated. Fig. 1 shows the system designed to create services over the onboard computer. All the sensors included in the vehicle are connected to the onboard computer. The computer is an ordinary PC with an interchangeable operating system and a Java Virtual Machine (JVM). Several programming APIs are situated over it. The most important ones are *Java Comm*, *Java Media Framework* and *Java Speech*. Java Comm is used in communications via serial port, such as the links to the GNSS and the odometry sensors; the Java Media Framework is useful in the multimedia software development; and Java Speech provides a speech synthesizer to the Java programs. Other APIs are used in the Java programming for graphical and mathematical purposes. OSGi is located over the Java basis, making the computer able to contain several services. In fact, as [10] summarizes, the software

entities installed over OSGi are called *bundles*, which can offer services. Back to the diagram, the set of elements visible over the OSGi layer are the services implemented in the platform. These are classified according to its abstraction level. So they are divided into low, medium and high level services. Low level services have the necessary software for a direct access to several devices in the vehicle. They could be considered as drivers for the vehicle hardware. Medium level services act as middleware between low and high level services. In this sense, this layer performs transformation and adaptation tasks, improving the functionality offered by low level services. Finally, high level services are bundle software with a user interface. This hierarchical structure of services has a double purpose. Firstly, the creation of new services is easier because of the modular programming. If any functionality is used or is expected to be used by several applications, this can be encapsulated as a service. On the other hand, the problems of access to real devices when several services require the use of a specific one are solved. That is, not only the software used as driver is reused, but also the implementation of a low level service here allows avoiding synchronization problems in the access to the device. This issue is really important in widely used sensors, as the GNSS one. Due to a large amount of services may utilize the positioning information, the access to this service must be coordinated. Indeed, services situated in the medium and low level layers of the architecture carry out caching of information, so it is possible to serve some requests without doing a new petition to a sensor, for instance.

Communications between layers is carried out by service interfaces (SI). Each layer defines a set of service interfaces which indicate the available functionalities. A SI is a Java interface which may be implemented by one or more OSGi bundles. The instance of such a bundle is a OSGi service. If a bundle situated in the upper layers needs a feature provided by one of the SI, a query to the OSGi core infrastructure with the SI as a parameter is launched in order to receive the set of available services which implement the functionality.

There are two bundles implemented in the same architecture which perform a well defined task in the proposed system. These are *Policy Manager* and *Service Manager*. They cannot be handle directly by the user. Service Manager has been developed as a high level bundle, as can be observed in the color used in Fig. 1. Its graphic interface shows the set of services which are installed in the system, arranged by their abstraction level. From here, the user can enable, disable or update the installed services in the system.

Policy Manager has been created as a medium level service, since it contains a group of functionalities used transparently by high level services. Thanks to Policy Manager, user level services are not allowed to be executed when any of the configured constraints is true. For instance, we can implement graphical interfaces to keep the user attention in driving when the vehicle is moving. Fig. 2 shows an abstract diagram of classes which explains the followed structure in the policy management. User level services, implemented as OSGi services or bundles, inherit from a class which includes all the necessary functionalities to make these services be constrained by fixed policies. These policies mark the operation of the services and are configured in the same service or bundle. At run time, an uninterrupted service, *PolicyManager*, is in charge of the periodical checking of the constraints associated to the registered services. This entity obtains all the necessary services (by OSGi queries) to check the constraints. These constraint services could be defined by the odometers installed in the wheels, for monitoring the car speed, a weight sensor in the co-driver seat to disable a specific service if there is not a person travelling with the driver when the vehicle is moving, etc.

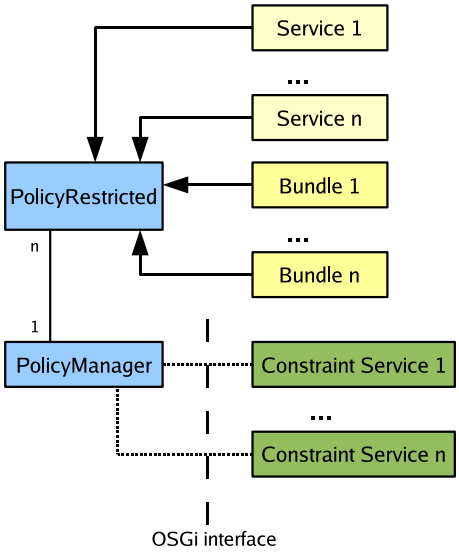

Fig. 2. Policy subsystem

III. NEW GENERATION SERVICES FOR VEHICLES

The described platform has been implemented and tested over a real system. Moreover, several services belonging to each level of abstraction of the architecture have been developed to demonstrate the viability of the proposed solution.

III-A. TECHNOLOGICAL ISSUES

The hardware platform used is based on a prototype of a vehicle widely sensorized used at the University of Murcia [3] in several research projects regarding autonomous navigation with GNSS/EGNOS and INS sensors. Thanks to the sensors installed in the vehicle, it is possible to create services dependant on the current state of the car. Among all the sensors it is worth noting the importance of the odometer captors situated in each wheel, and the GNSS/EGNOS receiver. Odometry is useful to monitor the movement of the vehicle, while the GPS/EGNOS receiver is a key tool for location tasks. The prototype vehicle is shown in Fig. 3. It includes an onboard PC based on a single board computer (SBC) architecture of VIA. A tactile screen allows the interaction with the user pressing directly on it. The common physical interface through mouse and keyboard are located in the lower part of the dashboard. The operating system installed on the computer is a Linux

Fig. 3. Prototype vehicle used in the development

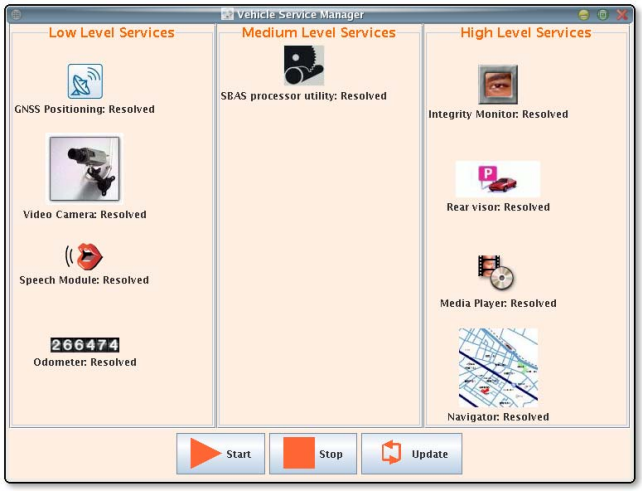

Fig. 4. Service Manager Service

Fedora Core 4, and the Java Virtual Machine 1.5 has been used. Two OSGi frameworks have been probed and installed, Knopflerfish [11] and Oscar [12]. Both are open source implementations of the OSGi Release 3 [5], and have been used to test the operation of the proposed architecture. Regarding the communications, the vehicle is provided with links to several network technologies, such as Bluetooth, 802.11 and GPRS/UMTS, covering all the connectivity necessities.

III-B. IMPLEMENTED SERVICES

As we have stated previously, there is a high level service implemented over the proposed architecture with the aim of allowing the user to manage the rest of services. This is the *Service Manager*, shown in Fig. 4. This application shows all installed services in the OSGi framework belonging to the proposed system. The user can start, stop and update them in a easy way. The services which are deployed to be included in our architecture have special features in their JAR archive, distinguishing them from the rest installed in the OSGi framework. All services are arranged in the screen according to their level of abstraction. The low level services developed so far are:

- *GNSS Positioning*. It is a generic service of access to the positioning sensor. Several GPS receivers are supported, and the service is designed through an extensible architecture which allows us to include new sensors without a great effort.
- *Video Camera*. Service to connect with a camera situated in the rear part of the vehicle. The logic of the service uses Java Media Framework (JMF) to obtain the image received from the camera.
- *Speech Module*. Encapsulates a voice synthesizer available for the rest of services which can emit spoken events, so the user do not have to avert his sight from the road to see the screen. This software has been developed using FreeTTS, a partial implementation of the Java Speech API.
- *Odometer*. This service provides information from the odometry sensors. The odometry of the vehicle is composed by four captors that obtain the rotation velocity in every wheel, estimating the vehicle velocity.

As middleware service the SBAS Processor Utility is included. This contains several tools concerning the Satellite Based Augmentation Systems (SBAS). In this sense, SBAS information, available from the GNSS sensor or received from Internet through SISNeT [13], is used for correction and integrity purposes. Corrections are adapted to a more common format, as RTCM is, and the HPL (Horizontal Protection Level) integrity parameter is calculated [14]. The cellular network link is necessary to connect with the SISNeT server when the SBAS signal is hidden.

High level services have a direct relation with the user. The set of implemented high level services are:

- *Integrity Monitor*. It is an application used to monitor the integrity of the location emitted by the positioning system. For this purpose, it uses the SBAS Processor Utility service, but the Speech Module service is necessary as well. for warning the user with a spoken alert when the integrity factor exceed a fixed limit.
- *Rear Visor*. With this service the user can see the rear part of the vehicle, thanks to the Video Camera service. The purpose of this application is to improve the sight of the driver in parking tasks.
- *Media Player*. It is a multimedia player which can load several video and audio formats. Since multimedia contents can be played in a software application we can see how video player devices, and even CD players and radios, can be replaced with software installed in the onboard computer.
- *Navigator*. This program has navigation and electronic fee collection (EFC) functions thanks to the use of a GIS (Geographical Information System) reference. The application depends on the GNSS Positioning service, since the vehicle position is needed. The SBAS Processor Utility service is used to obtain the position integrity, and the Speech Module service is useful to warn the user

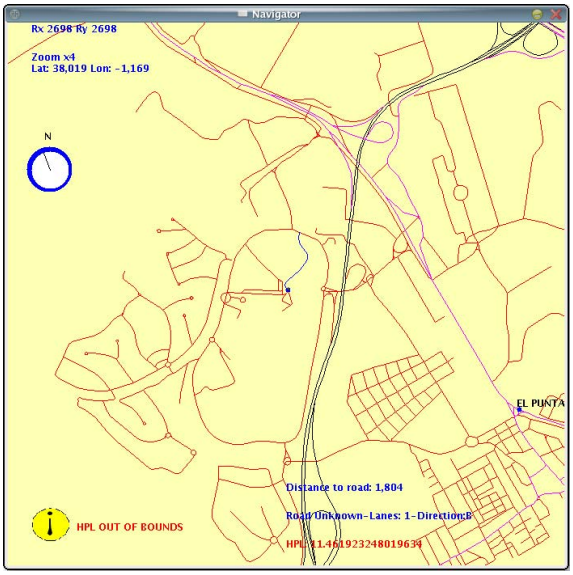

Fig. 5. Navigator service

about important events.

High level services are also linked to the policy system previously explained. All of them have to implement the necessary tasks to be made when the Policy Manager service indicates that the service is restricted because one of its associated constraints is true. At the moment, the movement restriction has been implemented. In particular, Policy Manager use the Odometer service to check if the vehicle speed is higher than a fixed threshold. If this is true, all high level services which are registered with this constraint will be notified about the event and consequently will do the proper action. For example, the Rear Visor service removes the image from the video camera, showing a warning message.

Fig. 5 shows an example of execution of the Navigator service. The screen shows the current location of the vehicle in the Campus of Espinardo at the University of Murcia. As can be observed, the window displays a warning message because the HPL (Horizontal Protection Level) is out of the fixed bounds. Although the program reports the event graphically, a spoken alert is also raised since the user can be currently driving. The rest of the data included in the window are the current position, the distance to the nearest road, and the information relative to this.

IV. CONCLUSIONS

The proposed architecture considers a platform based on a general purpose computer with the ability of supporting value added services to the driver. OSGi, after showing all its advantages in domestic environments, is considered as a key element in the development of software functionalities for vehicles. Because of this, an OSGi framework has been used for placing all onboard services. These have been classified in abstraction levels, allowing the creation of new services in a modular way. The vehicle used in our concrete implementation has a significant number of sensors, facilitating the services to be context aware.

The set of implemented services shows the feasibility of the proposed architecture and the different environments the onboard services can cover. In this sense, not only the location based services, but also the multimedia and driver assistance ones are considered in this paper. The navigator and the integrity monitor services developed in this work are considered into the LBS group, while the multimedia player and the rear visor are classified into the multimedia and driver assistance service types, respectively.

The growing implantation of functionalities in new generation vehicles encourage the implementation of systems like the one presented in this paper, in order to avoid an excessive unrecommended number of devices in the vehicle panel. In that way, adding services as software embedded in a shared general purpose computer can play a key role in the future vehicle manufacturing.

V. ACKNOWLEDGMENTS

The authors would like to thank the Spanish Ministerio de Educación y Ciencia for sponsoring the research activities under the grant 02622/BPS2/05, in frames of the FPU program. Special thanks also for the founding apported by the ANSO project ITEA-04016 and to the Spanish Ministerio de Fomento for its continous support.

REFERENCES

[1] Craig Simonds. “Software for the Next-Generation Automobile”. *IT Professional, November/December 2003, pp 7-11. January 2003.*

[2] Ward Vanlaar. “Legislation, Regulation and Enforcement for Dealing with Distracted Driving in Europe”. International Conference on Distracted Driving, Toronto. October 2005.

[3] Skarmeta A.G., Martínez H., Zamora M.A., Úbeda B., Gómez F.C, Tomás L.M. “MIMICS: Exploiting Satellite Technology for an Autonomous Convoy”. IEEE Intelligent Systems. N IV. V. vol. 17, pp. 85-89. July 2002.

[4] Massaki Wada, Xuchu Mao, Hideki Hashimoto, Mami Mizutani, Masaki Saito. “iCAN: Pursuing Technology for Near-Future ITS”. IEEE Intelligent Systems, January/February 2004, pp 18-23. January 2004.

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

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

[7] Luciano Baresi, Carlo Ghezzi, Antonio Miele, Matteo Miraz, Andrea Naggi, Filippo Pacifici. “Hibrid service-oriented architectures: a case-study in the automotive domain”. 5th International Workshop on Software Engineering and Middleware (SEM’05), Lisbon. September 2005.

[8] Zoltán Benedek. “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. “The Gator Tech Smart House: A Programmable Pervasive Space”. Computer. Vol. 38, no. 3, pp. 50-60. March 2005.

[10] Choonhwa Lee, David Nordstedt, Sumi Helal. “Enabling Smart Spaces with OSGi”. IEEE Pervasive Computing. Vol. 02, pp 89-94, Jul-Sept. July 2003.

[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. “The ESA SISNeT Project: Current Status and Future Plans”. European Navigation Conference GNSS 2004, Rotterdam. May 2004.

[14] José Santa, Benito Úbeda, Rafael Toledo, Antonio F. G. Skarmeta. “Monitoring the position integrity in road transport localization based services”. Vehicular Technology Conference 2006 Fall, Montréal. September 2006.