

Wireless Vehicular Communications and Services for Breakdown Support and Car Maintenance (W-CarsCare'05) Open Platforms for service provision

Juan C. Dueñas¹, José L. Ruiz^{1*},
Jesús Bermejo², José A. Alonso²,
Carlos Acuña³, Cristina Díaz³, Miguel Gonzalez³, Miguel García³, Alonso Álvarez³
¹Departamento de Ingeniería Telemática-ETSI Telecomunicación
²Telvent, S.A.
³Telefónica I+D

Abstract

The service provision oriented architectures (SOA) are turning into one of the main trends in the scope of information technology. It could be glimpsed a new paradigm in the World of the application and service provision derived from the convergence between these two SOAs - which develop a strong drive regarding integration and the application life cycle – besides the increase of the processing capabilities of devices and the communication network wideband. Within this new scenario, services are constructed and linked dynamically and users could employ them from anywhere and anytime. With the present paper, Our present on-going work developed in the scope of EUREKA ITEA OSMOSE [5] is introduced, having the goal of proposing a viable model based on, as far as possible, the use of free/libre/open source (FLOSS) software within the Service provision oriented architectures.

1. Introduction

The progress of the information society, distinguished by the availability of electronic mechanism supporting all the social life scopes, is based in the availability of new telematic services and in service provision and operation and linked dynamically.

The convergence of the electronics, computing and telecommunications sectors makes feasible to achieve the recent innovations in the field of software engineering related with Service Oriented Architectures (SOA) [1]. A service is an “entity which can be used by one person, program or other service, that performs some computing functions, storing information, communicates with other users, physical devices or even with other services”.

A service-oriented architecture is the one, which is organized following certain well-defined rules:

- Services should have interfaces described by a formal language, which defines the operations offered.
- To make the model dynamic, it is necessary that services can be published in registers accessible by standardized mechanisms, which let first service discovery and afterwards to be used by other services and users, see Figure 1.

There are different technologies that can be catalogued within the SOA paradigm, such as the following: Web Services [1], OSGi[4] or Jini[2]. Each one has its own application scope and different goals, for instance, Jini is used in service publishing in short range networks, Web Services try to define the interactions among available services on the Internet, and OSGi tries to integrate services in the

home, car or office environments, in local and personal areas to the Internet.

In this environment it is necessary to face new challenges: the services should offer an adapted behaviour to its content, its deployment should be supported by automatic mechanisms, and users and services should be considered independently from their location (mobility).

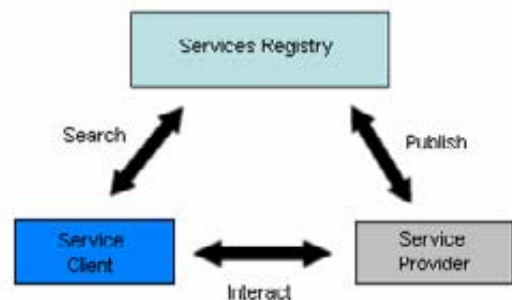


Figure 1 Service oriented architectures

The contents, applications and services provision is an area especially important that should benefit from the dynamism and integration of SOA. There are other relevant factors such as :

1. The **computing capacity ubiquitous (disculpa la ignorancia pero no lo entiendo)**. The electronics evolution has created a wide number of smart devices provided with communication interfaces: onboard computers, PDAs, mobile phones, videogame consoles, the evolution of the electronics, “set-top boxes”, multimedia devices, etc.
2. The **services middleware**, means a medium layer that provides to the application developers,

an abstraction support in non-functional aspects, such as fault tolerance, quality of service, application logic layer delivering, or security. The availability of middleware platforms makes the providers just focus on exclusively of the business logic making rapid development of services easier to them.

3. **XML Technologies**[6], to ease the structured exchange among application, based on heterogeneous technologies

In the rest of the document the research developed in the EUREKA ITEA OSMOSE project is going to be deepened. The research in that project showed an end-to-end (e2e) service provision scenario in the residential environment, but it is also valid in the vehicle environment.

One of the most outstanding sides is the evaluation and viability of using FLOSS as a part of provision architecture. A deep description of the OSGi middleware platform will be done, due to its important role in the development of this work. Ultimately, Conclusions will be showed and the next steps in this research will be presented.

2. End-to-End Services



Figure 2: End to end services (lo mismo que antes E-2-E or end-to-end)

The service gateways (GW) showed in figure 2, has a key role in this scenario. These devices are in charge of interconnecting internal networks among them and giving external connection through Internet with access networks (DSL, cable, satellite...)

Likewise, these gateways are capable to remotely host, manage and monitor applications and domestic devices.

The essential difference between this kind of gateways and other kind of terminals (in particular, a PC) is that management and maintenance of the device and the installed applications should be achieved with no intervention of the final user.

The possible scenarios include much as deployment in residential as in mobile environments, such as in vehicles, which could be considered a conceptually extension of the residential environment. This

allows a remote access to the device installed in the user's vehicle and perform diagnostics, retrieve information from the vehicle, or update or fix the software installed. The range of possibilities is wide. This way the service provider can concentrate in just one node in order to show their assets to each platform installed. Regarding the kind of added-value services that can be deployed, the possibilities seem endless: entertainment services (movie and music delivery...), communications (e-mail, IP telephony, teleconference...), location such as GPS, allowing to create added-value services based on location, like seeking the nearest particular service (petrol stations, restaurants, etc), traffic information, maps, location in case of stealing or breakdown, etc...

The gateway operator acts as a manager in this scenario. For achieving this, the operator should be capable of facing the variability of the gateway features (memory, CPU speed, existence and capacity of mass storage) and heterogeneous devices, networks and kind of services. In the case of the vehicle environment, the limits of the devices are even bigger mainly due to the size and the characteristics of an unstable and noisy environments.

These management tasks are inevitably integrated with other services support: billing, call centres, service subscription management, security, etc.

The scenario architecture is broken down, analysing the kind of available support in terms of middleware technologies, platforms and services.

3. Operation support systems

The gateway operator architecture agglutinate a set of complex functionalities, some of them are already available in commercial components, such as call centres. These complex functionalities can be splitted into six blocks.

3.1. Deployment

This is one of the fundamental blocks in the scenario in charge of installing, adapting, deleting, updating and start-up up the services over the service gateways. When the application is deployed it is also responsible of fixing dependencies among services and manage the conflicts between installed components in the service gateway and the ones under installation.

3.2. Call centre

This functional block manages the customer information. It allows entering, editing and deleting customer data, including service subscriptions.

If the customer has any problem with any service or domestic device, this information is forwarded to the gateway manager. The incidences will be received and processed, occasionally being forwarded to the corresponding service provider.

3.3. Gateway Management

This functional block is the one in charge of supporting the residential gateways management as much IP gateway as service deployment gateway, including security management and service permissions, the domestic devices, the deployment service access...Due to its complexity, it can be divided into: **Initial Configuration**, which is in charge of providing a start-up mechanism, means to install the minimum necessary software components to be monitored from the control centre; **Monitoring**, supervising the service gateway network detecting failures, send alarms and retrieve information about resource availability; **Configuration and support**, managing the configuration of the installed services in the gateway and the execution platform itself, for instance taking care of the modification of the network interface configurations or rebooting the gateway

3.4. Service and product catalogues

This is the service where new services and products are published for the final users. Besides the interfaces such as Web Services, a portal is offered to the final users where to select and to activate subscriptions to the new services. Service providers have to take care of updating the catalogue with the functional block mentioned above.

3.5. Providers Management

This service manages the service provider information: inserting, updating and deleting services. Optionally, the service provider will deliver components to the gateway manager, which will be used later in the processes of gateway configuration and deployment.

3.6. Billing

The billings tasks are focused on the charges management derived from the use, downloading and/or subscription to the services. From this information this functional block generates the billing information for each customer.

4. Middleware Technologies used

Once the provision scenario is more or less depicted, the two basic technologies selected can be described: OSGi[4] as basic infrastructure middleware for the service gateways, OSS/J[9] as necessary middleware for the gateways managers and Web Services as interaction support, spread in all the architecture.

OSGi Open Services Gateway initiative

Due to its relevance in the framework of EUREKA ITEA OSMOSE project, and its potential application, this technology is described deeper in the next section (5).

OSS/J

“Open Services for Support through Java Initiative” is an industrial initiative with the purpose of improving the interoperability between different operation support systems in heterogeneous environments in telecommunications. OSS/J promotes a new multilayer architecture based in reusable components and container and application server technologies

The results from OSS/J have been used as a conceptual framework in the functional block definitions showed in section 3.

The results from OSS/J have led to the development of APIs and reference implementations, which from one side reassure the system compatibility and from the other, eases the rapid and successful development of OSS solutions. The interaction model with the OSS APIs are of two kinds: synchronous (request response model) and asynchronous (based in triggering XML messages).

These are the most relevant APIs in the end to end provision scenario:

Trouble Ticket API

To allow the incidence management from the call centre and from network management systems: creation, updating, searching and reports.

Billing API

To make easy the communication between different billing systems on runtime. It simplifies the data routing among different nodes and with different billing registers (CDRs, SDRs, IPDRs).

Service Activation API

This API is based in the concept of order (asynchronous business process that satisfies the user's service needs) and service... Basically, it is in charge of managing the life cycle of the available services

Service Inventory API

This API provides functionalities of CRM, service and resources management. The product inventory information, services and resources let define planning processes and evaluate the availability/status of the assets.

The following functionalities are offered:

- Entities management, specifications and inventory relationships.
- Data and metadata consulting.
- Notification delivery of Inventory events

Quality of Service API

This API makes easier the exchange of data between QoS and non-QoS components within the context of the business process of the service providers. The

quality of service is defined as the effect of service operation, which determines the degree of user satisfaction when using it. This is divided into two kinds of services: end to end, and carrier services. The extreme to extra services are traced from one terminal to another. The user of one extreme decides about the quality of such services and whether is satisfied with the QoS provided or not. The carrier services are the telecommunications ones which provide the transmission capacities of the signal between two access points in the network, therefore these services are for supporting extreme to extreme services.

Web Services

These are application components whose functionalities and interfaces are offered to potential users, by using a series of standardized technologies: XML (Extensible Mark-up Language), SOAP[7](Simple Object Access Protocol), WSDL[8] (Web Services Description Language) and HTTP protocol.

The technologies mentioned in the latter definition encloses with the fundamental ideas of a SOA. Taking XML as a reference language, services are described using the WSDL language are will be accessed exchanging SOAP messages. The service interaction is quite similar to procedure calls, but using XML/SOAP.

The Web services register, necessary role in a SOA, can be freely implemented by different technologies, such as UDDI (Universal Description Discovery and Integration) or ebXML.

The benefits derived from the increase in the capacity interaction and collaboration among organizations, explains the current importance of web services in the information and communications technologies world. All this thanks to the use of a flexible, language and platform independent application model and technologies.

5. OSGi

The next section is based in the document "About the OSGi Platform" [23] (©2004 OSGi Alliance) and is included for documentation purposes.

OSGi is a development environment for networked services. OSGi is standards based, and component oriented. Basically, the OSGi environment provides means to manage remotely the components life cycle on a client platform. The different operations on the components (installation, removing, start and stop, updating) don't affect the platform operation.

OSGi components are software (programs, libraries) that can interact with other components, discovering and using them dynamically. Under the OSGi specification, an increased number of components are developed.

The main OSGi architectural element, the Service Platform, is an application server, Java-based, that can be found in:

- Mobile phones
- Cars and other transportation means
- Telematic devices
- Residential gateways
- Industrial computers
- Desktop PCs

OSGi main characteristics are:

- It is a development framework, Java-based, with a flexible deployment API that controls the life cycle of applications.
- It consists on an environment that allows running different components sharing resources (like the same Java Virtual Machine) and, in a secure environment (sandbox) that avoid these components cannot damage the environment, nor interfere with others.
- It defines a cooperative model when applications can discover and use dynamically services and resources provided by other applications in the same OSGi platform.
- It is a flexible architecture that allows remote management of thousands and even millions of service platforms from a unique domain.
- It includes a series of standardized services: Logging, Configuration, Http, XML, Wiring, IO, Device Discovery and driver loading, User authentication and Authorization, Preferences, and UPnP. Additionally, it provides a number of standardized utilities.

Key Features of the OSGi Service Platform

The OSGi Framework provides some functions to manage the life cycle of components:

- A packaging format for the applications: the Bundle format, similar to standard Java Archive (JAR) file, and fully compatible with ZIP.
- Install a bundle.
- Start/Stop a bundle. Starting a bundle makes certain resources available, stopping a bundle performs a clean-up. Every bundle is running in the same JVM, therefore saving resources.
- Update a bundle. It is a dynamic process that happens without restarting the JVM.
- Uninstall a bundle.

Remote Component Management

OSGi platform is designed to allow remote management. This function is accomplished in a protocol independent way, using a standard API

defined by OSGi. By not specifying a protocol, the management system can optimise for the carrier characteristics (including special characteristics for i.e. mobile phones, residential gateways, ...).

Secure Execution Environment

OSGi offers a comprehensive security model with several layers: JVM security mechanisms; safety features of the Java language; the (optional) level of defence is Java 2 code based security (specific permission schema for every bundle); and the final level is the OSGi Framework that strictly separates bundles from each other (bundles need appropriate permissions to even detect other bundles on the OSGi Service Platform).

Cooperation Between Applications

The OSGi Service Platform allows bundles to contribute code as well as services to the environment. With the download of libraries, applications can share common code, making applications smaller. This is a great difference with other Java application server models (MIDP, J2EE, Avalon, JMX, ...), which provide a closed container in which the application runs in isolation.

The OSGi Platform also provides a way to lightweight publish, find, and bind services inside the JVM. In OSGi, a service is a mechanism that allows one bundle to provide functionality to other bundles. The implementations of the service objects can be provided by different vendors, different optimisations or for different devices. This characteristic allows the existence of OSGi COTS.

Simplified Deployment

OSGi provides mechanisms that simplify the process of deployment. Once an OSGi Service Platform is installed in an environment, deploying applications is smooth because the Service Platform hides the differences of the underlying environments (underlying JVM and operating system become visible to the applications).

Dynamic Nature

OSGi makes possible the platform and JVM in that a lot of operations: installing bundles, registering new services, or updating an existing component.

Thus, the OSGi Service Platform brings standardized hot update technology to small embedded devices, desktops, and servers enabling these devices to run continuously, even when their software is configured, updated, or augmented.

Policy Free

The OSGi Service Platform provides many mechanisms but does not dictate how those mechanisms must be used. Policy free leaves the

platform operator in charge. The OSGi Service Platform can be used in a completely managed environment like a mobile phone, as well as running on the desktop without any supervision.

6. Provisioning Basic architecture

Provisioning architecture for an extreme-to-extreme scenario is based on technologies described in the chapter 4 and 5, and supports the functionalities described in chapter 3. The provisioning comprises all the required activities to publish a service, deploy on an execution environment, configure and left it in a stable initial state.

Figure 4 shows the gateway manager service architecture. In this case, the executing basic environment is based on J2EE. Every service is deployed over this environment. A RDBMS give persistency support to the gateways information.

The archs in the figure show the relationships of the OSS/J standard APIs with services (in order to clarify, SI stands for Service Inventory, SA stands for Service Activation, QoS stands for Quality of Service and Bi stands for Billing).

This figure shows two additional elements not commented: Web Services engine, that publishes services, and helps interaction between residential gateways, service providers, and services deployed in management centre, and UDDI, the registry for these services.

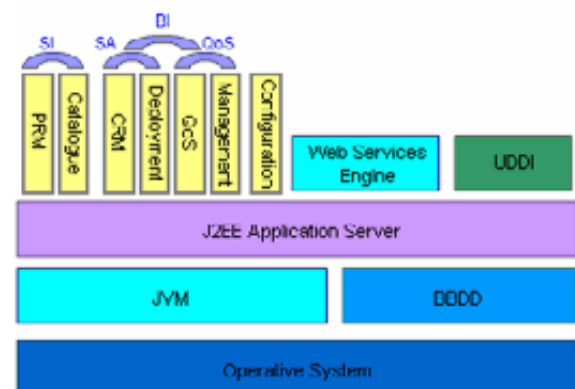


Figure 4: Gateway manager architecture.

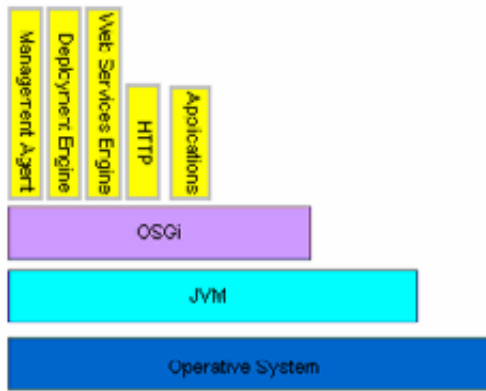


Figure 5: Gateway architecture.

This figure simplifies what “management agent” actually is, in fact, it agglutinates some functionalities like configuration management (provided by the OSGi service “Configuration admin.”) permissions and users management; bundles state change; OSGi executing environment properties management; and devices gateway connected management.

“Deployment engine” component helps deployment services in the gateway manager to avoid being a bottleneck.

The gateway also has a Web Services publishing engine.

7. The OSMOSE Project and the open source middleware

The work described in this paper is related to the development of the EUREKA ITEA OSMOSE project, when are involved important companies in the market like: *Philips, France Telecom, Telvent o Telefónica I+D.*

Project main goals are closely related with open source middleware (its name tells everything: Open Source Middleware for Open Systems in Europe). Project work is based in ObjectWeb[10] consortium source code, and tries to extend the architecture in some layers: frameworks (reused components, with a standardised API) and execution platforms.

Some test beds are used as validation of the work developed in every layer: telecommunications, avionics, and home gateways scenarios.

Currently, middleware is a critical element in distributed systems. Thus, is easy to see the success of technologies like Sun J2EE, OMG CORBA or Microsoft .Net [15].

Open source software begins to be a feasible alternative for robust software system development. A good example of this is the success of GNU/Linux [21] operating system, or Apache [11] web server.

Source Code based projects progress in speed and quality due to source code availability, Internet access ubiquity and work group tools existence. These projects are developed following an iterative improvement model with close collaboration between developers and users.

One of the main reasons to introduce open source software in environments like the previously mentioned is the economic investment needed for massive deployments in an extreme-to-extreme solution. To cut cost on home gateway architecture is essential to give massive access to this technology with low prices. And this could be easily achieved when APIs and tools are in the public domain.

OSMOSE project use open source tools: application servers (Jonas, JBoss); web services engines, directories and tools (Axis, Juddi); OSGi platforms (Oscar, Knopflerfish); and Operating Systems (Linux). Most of them are under LGPL and ASL licenses.

In spite of that, one essential and basic tool (the OSGi execution environment) does not have such a license: Java Virtual Machine. It is possible to find open source virtual machines in the market [12] but their current development status is not so advanced and mature as their commercial alternatives [14][13], which do not allow free distribution.

8. Conclusions

This paper shows how technologies developments under SOA, computing ubiquity and Internet connectivity, have a great impact on services provisioning process.

Middleware technologies as support for application developers, XML use as neutral and universal standard for information exchange, and operation support system, play a key role in the evolution of the information society.

We have demonstrated that it is possible to link these concepts to develop a complete extreme-to-extreme provisioning scenario. And we have evaluated the viability of using FLOSS to develop these scenarios.

The work showed in this paper is integrated in the EUREKA ITEA OSMOSE project, currently under development. Its future tasks include apply these concepts in different scenarios with other devices than home gateways like mobile phones or car embedded systems.

References

- [1] Web Services and Service Oriented Architectures. Article available at: http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html

- [2] Jini resource community, available at <http://www.jini.org>
- [3] Open Services Gateway Initiative. "OSGi Service Platform," Specification Release 3.0, March (2003)
- [4] OSGi, available at the consortium website at <http://www.osgi.org>
- [5] EUREKA ITEA OSMOSE, official project website at <http://www.itea-osmose.org>
- [6] XML, eXtensible Markup Language. W3C specification, information available at <http://www.w3.org/XML/>
- [7] SOAP, Simple Object Access Protocol. W3C standard, specification available at <http://www.w3.org/TR/soap/>
- [8] WSDL, Web Services Description Language. W3C standard, specification available at: <http://www.w3.org/TR/wsdl>
- [9] OSS/J, website of initiative at: [en http://ossj.org](http://ossj.org)
- [10] ObjectWeb, página web disponible en <http://www.objectweb.org>
- [11] Apache Software Foundation, website available at <http://apache.org>
- [12] GCJ GNU Compiler for Java, available at: <http://gcc.gnu.org/java/>
- [13] Java Virtual Machine, Sun Microsystems, available at <http://java.sun.com/j2re1.4.2>
- [14] J9 Virtual Machine, disponible desde la web de IBM en <http://www.ibm.com>
- [15] Microsoft, .NET, <http://www.microsoft.com/net>
- [16] Jonas, <http://www.objectweb.org/jonas>
- [17] Jboss, <http://jboss.org>
- [18] Axis, <http://ws.apache.org/axis/>
- [19] Oscar, <http://oscar-osgi.sourceforge.net/>
- [20] Knopflerfish, <http://www.knopflerfish.org/>
- [21] GNU/Linux, <http://www.gnu.org>
- [22] Juddi, <http://ws.apache.org/juddi/>
- [23] "About the OSGi Service Platform. Technical Withpaper", ©OSGi Alliance 2004, http://www.osgi.org/documents/osgi_technology/osgi-sp-overview.pdf

Final note

The substantial material used in this paper has been extracted from a paper exposed in a Spanish Conference (Telecom I+D 2004).

* The work developed by Juan C. Dueñas and José L. Ruiz is part of their contribution to the OSMOSE project (Eureka 2023, ITEA ip00004), funded partially by the Telvent corporation and Science and Technology Office under the reference TIC2002-10373-E.