DEVELOPING NEW SERVICES FOR THE AUTOMOTIVE INDUSTRY USING MATRIX

Silvia Alén
CTAG: Technological Automotive Centre in Galicia
Pol. Industrial A Granxa, Calle A, Parcela 249-250, 36400, Porriño (Pontevedra), Spain

Daniel Glez-Peña, Florentino Fdez-Riverola
Escuela Superior de Ingeniería Informática
Edificio Politécnico, Campus Universitario As Lagoas s/n, 32004, Ourense, Spain

Keywords: MAS architecture, Car2Car, Car2Infrastructure, OSGi, JADE.

Abstract: This progress report explores how multi-agent systems can be applied to the automotive industry in order to create an intercommunication platform. Our proposal called MATRIX (*Multiagent Architecture for TRaffic Information eXchange*) provides the basis for information exchange between cars (Car2Car) and between cars and infrastructural elements (Car2Infrastructure). In this work we also discuss how the most prominent technologies in both areas (JADE and OSGi) can be integrated in order to achieve new functionalities and security improvements in present cars.

1 INTRODUCTION

Since its appearance at the end of the 19th century, the car has become a basic article in everyone's life. In early 2005 there were 26.5 million of cars in Spain (DGT, 2006). The same year, almost 2 million of cars were registered (ANFAC, 2006). However, the automotive industry has multiple challenges where the security arouses everyone's interest.

Every year 120,000 deaths by accident take place in Europe. Those accidents suppose for the states members a joint cost of 160,000 million euros, which ascends to 2% of the Europe GIP (Gross Industrial Product).

These worrisome numbers forced the creation of a working group within the European Commission called *e-Safety* (e-Safety, 2006). *e-Safety* group was born with the purpose of proposing a strategy to promote the investigation, development, implantation and use of security intelligent systems based on ICT (Information and Communications Technologies) destined to increase the road security in Europe.

The work group *e-Safety* published its final report at the end of the year 2002, concluding that the application of the ICT to offer security systems for intelligent vehicles is a great instrument for the resolution of security problems on the road transport. The use of the ICT to provide new and intelligent solutions includes the participation and the interaction of the driver, with the vehicle and the road surroundings.

In this integrated and global approach of the automotive industry security, the embedded independent surveillance systems are complemented by cooperative technologies. Those technologies are intended to use *vehicle to vehicle* (Car2Car) or *vehicle to infrastructure* (Car2Infrastructure) communication strategies with the purpose of obtaining data belonging to the road surroundings in order to detect possible dangers and optimize the effectiveness of the onboard security systems (CE, 2003).

The design of new forms of communication between cars is one of the telematic services to increase cars security and protection. One of the most important supporter for the creation of a

Alén S., Glez-Peña D. and Fdez-Riverola F. (2007).
DEVELOPING NEW SERVICES FOR THE AUTOMOTIVE INDUSTRY USING MATRIX.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Internet Technology*, pages 204-209
DOI: 10.5220/0001274702040209

standard communication protocol at European level is the Car2Car Communication Consortium (Car2Car, 2006). Car2Car is a non-profit organisation created by European vehicle manufacturers, which is now open for suppliers, research organisations and other partners. The Consortium is dedicated to the objective of increasing road traffic safety and efficiency by means of inter-vehicle communications. The main goal of Car2Car is to create and establish an open standard European industry for Car2Car communication systems based on wireless LAN components, able to guarantee European-wide inter-vehicle operability.

Taking into account the increasing tendency towards communication between cars in the automotive industry, we present the MATRIX architecture focused not only in increasing the security, but also in supplying information for developing infotainment systems. MATRIX (*Multiagent Architecture for TRaffic Information eXchange*) defines a multiagent system which improves inter-vehicular and vehicle-infrastructure communications by an effective and efficient use of standard technologies.

Potential applications of the MATRIX architecture cover all telematic services defined by the working group e-Safety, including: (*i*) security and protection (e-call and the pursuit of vehicles), (*ii*) telematic services like remote diagnosis and proactive maintenance, (*iii*) positioning systems and itinerary like dynamic navigation, management of the POIs (points of interest) or road and traffic information and (*iv*) fleet management and infotainment services (entertainment, access to Internet, information services, e-mail, etc.).

The rest of the paper is organized as follows: Section 2 summarizes the technological context related with this work. Section 3 presents the main issues about the proposed architecture. Finally, Section 4 gives out the concluding remarks.

2 TECHNOLOGICAL CONTEXT

In the last years, software development has become more important in the automotive industry. A car contains an average of 35 million lines of source code and, for example, the Linux OS has about 6 million. This fact demonstrates the importance and the complexity of software development within the automotive industry, without considering the factor that supposes the risk for human lives.

Like in other areas where software is developed, but particularly in the automotive industry, the use of standards is always worthwhile because it supposes a reduction in the development time and guarantees security, reliability and reusability of the applications with the consequent costs reduction.

Nowadays, there are several standards in the automotive industry: AUTOSAR (*Automotive Open System Architecture*) (AUTOSAR, 2006) and MISRA-C *(Motor Industry Software Reliability Association*) (MISRA, 2006) whose purpose is giving attendance to the automotive industry in the implementation of safety software systems, or OSGi (*Open Services Gateway Initiative*) (OSGi, 2006) providing software modules management middleware for interconnected environments. In the following we present those standards in detail.

2.1 OSGi

OSGi is a technology developed by the OSGi Alliance, who involves companies such Sun Microsystems, IBM, BMW, Oracle, Nokia, Toshiba and Telefónica I+D. The main purpose of OSGi is the development of an open specification to provide services for interconnected environments (Chen and Gong, 2001) like houses and cars. The OSGi Alliance facilitates specifications, support for implementations vendors, test suites and compatibility certifications. OSGi has recently released the version 4.

OSGi has been chosen as our base platform in MATRIX because it is one of the mightiest technologies in the sector. For example, Nokia and Motorola are developing a standard based on OSGi for the next generation of smart phones; AMI-C (*Automotive Multimedia Interface Collaboration*) has included OSGi as an intrinsic part of its specifications, and BMW, among others, has incorporated OSGi as a standard part of its high-end platform.

This platform is not only supported by commercial companies, but also by the open-source community, with projects like Apache Felix (Apache, 2006), Eclipse Equinox (Eclipse, 2006) and Knopflerfish (Knopflerfish, 2006).

2.2 AUTOSAR

AUTOSAR is an international consortium created in July 2003 to provide a framework for automotive software, functional interfaces, management and a integration methodology, that is to say, to supply an open industrial standard to develop components for present cars.

Permanent members of this consortium are automotive companies like BMW, DaimlerChrysler, Ford, General Motors, PSA Peugeot Citroën, Toyota and Volkswagen, as well as companies leaders in the manufacture of components for automotive like Bosch, Continental and Siemens VDO. In this context, the main areas within the vehicle are the electrical system, the systems of conduction, the chassis, the functions of comfort, security systems, multimedia/telematic systems and the human/machine interface.

AUTOSAR is trying to establish a standard specification to develop vehicle components, which is due to be completed by August 2006. This process deals with the interdependencies that can appear in a distributed environment, including the underlying system's standard functions and the definition of the module integration of different manufacturers with its interfaces. The specification will also include the maintenance of electronic components and the software upgrades during the whole production cycle.

3 MATRIX ARCHITECTURE

The main idea behind our proposed multiagent design is conceptually simple: each mobile entity or infrastructure should have a representation in our MATRIX system in the form of an intelligent agent able to establish relationships with other mobile or static devices.

Our final purpose is that real devices will take advantage of the communication capabilities of the multiagent paradigm. Different examples which serve as real applications of our architecture are the following: (*i*) a car could publish its GPS position so it can be further located by other agents, (*ii*) a car could send messages to other cars (Car2Car) or could detect the closest infrastructural elements like powerboats or lodging (Car2Infrastructure) and (*iii*) a car could interchange information between many other applications in a local or remote way to solve a particular problem.

Figure 1 shows an overview of the MATRIX architecture. The top level shows a logic view of the

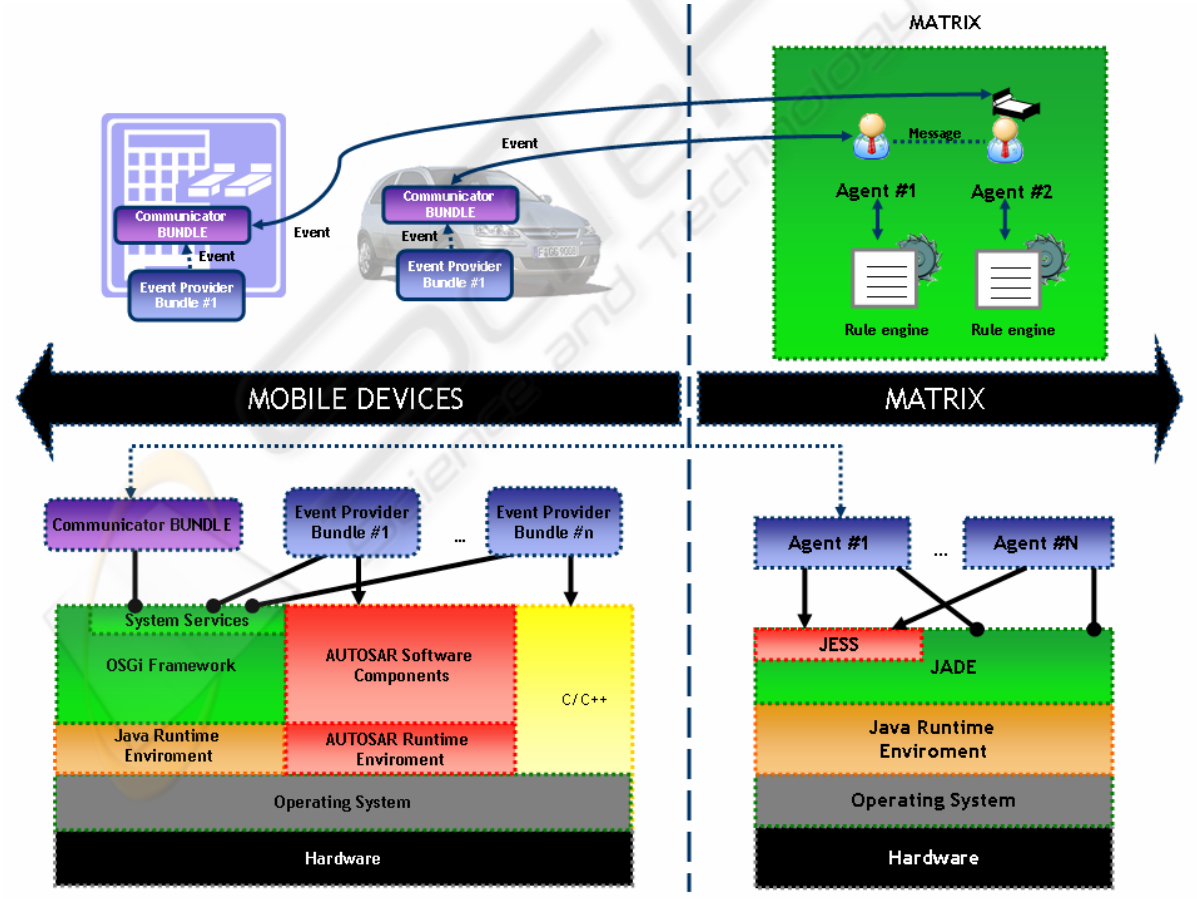

Figure 1: MATRIX multiagent architecture.

system where real elements live and interchange information in the model, whereas the bottom level includes the related technologies that support the system.

The implementation of the architecture on the mobile device side (left part of Figure 1) is based on OSGi, whereas its representation in MATRIX (right part of Figure 1) is based on JADE agents (JADE, 2006). Concretely, the components built in this architecture are:

- **Communicator bundle**: establishes the association with its agent in MATRIX to send and receive events. This connection will be established through Internet.
- **Event provider bundle**: devices could have installed one or several bundles of this type. Each bundle will be programmed to generate a concrete event to be finally sent to the agent through the Communicator Bundle. It is possible that the implementation of a bundle of this type requires the use of AUTOSAR in order to access low level information related to car's internal state (i.e.: airbag activation detection, oil levels, fuel, etc.).
- **Agent**: represents a device inside MATRIX system and manages events and messages sent by the device and by other agents. The decisions taken by the agent are established by a rule engine.
- **Rule engine**: eases the implementation of the agent's behaviour, which can also be easily modified in order to adapt MATRIX to a concrete problem.

3.1 Implementing Knowledge in Agents

MATRIX is not an end product, but rather represents an architecture that requires an adaptation to each concrete problem on which it will be applied. For that reason, the modification of the decisions taken by the agents in response to the arrival of different messages/events should be easily modified, reusing its code as much as possible.

To achieve this feature, the use of rule engines is considered essential, because they allow representing the logic of the agent reasoning process in separated rule files. The key benefit is the minimization of the adaptation cost being necessary for each particular implementation. There are multiple rule engines available like JRules, Drools, CLIPS, SMOOTH, JESS, etc. For our proposed architecture we chose JESS (Friedman, 2003) because the agent platform used in this work, JADE, was integrated natively with this rule engine through the JESSBehaviour.

Here is an example of a JESS rule file that could define the behaviour of an agent representing a vehicle that receives an event when the airbag is activated. After the event notification, a Mayday message is sent to all agents of type *ambulance* registered in the DF (*Directory Facilitator*) and at a near GPS position.

```
(defrule airbag-message
     (MatrixEvent (type airbag) )
     =>
     ;SearchDF asserts in the memory all the
MatrixAgent of a given type
     ;In this case, we will search emergency
agents
     (SearchDF (type emergency) )
   )
```

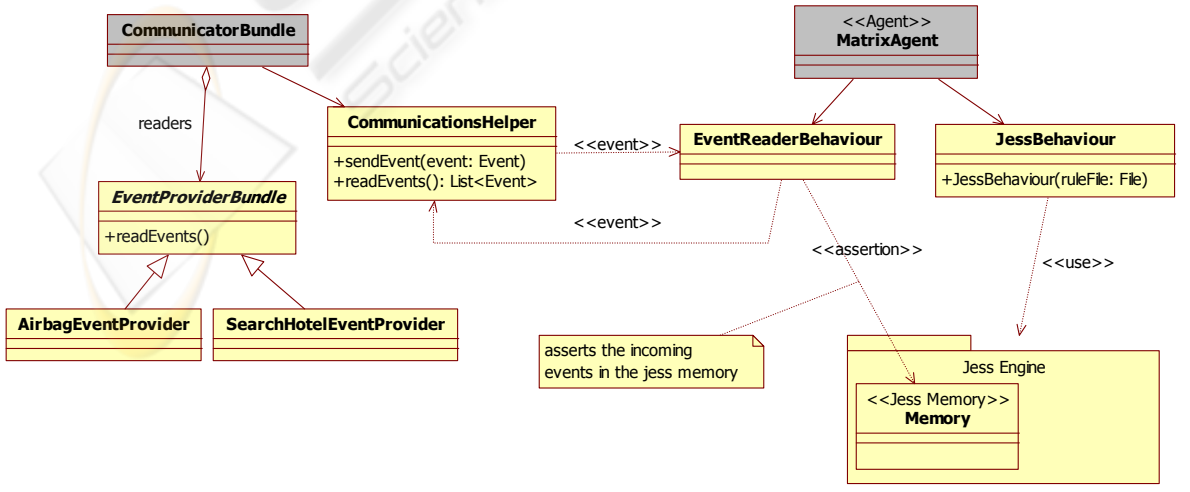

Figure 2: Class diagram of the core MATRIX architecture.

Table 1: Software elements to implement in order to adapt MATRIX to the problem of *looking for accommodation*.

Element needed	Location	Description
Event Provider Bundle 1	Hotel	Collects information coming from the hotel manager (available prices) and generates an event to be sent to the agent through the Communicator Bundle. The agent also knows the hotel GPS position.
Event Provider Bundle 2	Vehicle	Collects user information about the desired hotel distance and the price limits.
Rules	Vehicle agent	***If*** *event type is "look for accommodation"* ***then*** *it searches for nearest agents of type hotel sending them a requesting price message (CFP).*
Rules	Hotel agent	***If*** *type of message is "look for accommodation"* ***then*** *it sends prices to the solicitor agent.*

```
   (defrule call-emergency
     ?m <- (MyAgent (name ?n) (GPSposition
?p ) )
     ?a <- (MatrixAgent( (GPSposition ?q)
(< (distance ?p ?q) 100) ) )
     =>
     (send (ACLMessage (communicative-act
PROPOSE) (sender ?n) (receiver ?a) (content
"Mayday!") ) )
     (retract ?a)
   )
```

Please note that in the above piece of code there are predefined functions related to the JADE framework, such as *SearchDF*, which can be useful for the implementation of new rules during the adaptation of the MATRIX platform to a concrete problem.

3.2 Software Design

Figure 2 shows the class diagram used to define the MATRIX basic design. Our proposal incorporates the following classes:

- **CommunicatorBundle**: establishes the communication through sockets TCP/IP using package *javax.microedition.io,* which is part of OSGi standard. This bundle detects the existence of another bundles of type EventProviderBundle and listens to events that they could generate. In addition, the CommunicatorBundle receives events from its agent in MATRIX coming from other devices.
- **EventProviderBundle**: this abstract class represents a bundle that generates events. A specific implementation might be a bundle which controls the airbag state and generates an event if the
- airbag was activated (class *AirbagEventProvider*), or a class that interacts with the user to get the accommodation details before generating the corresponding event (class *SearchHotelEventProvider*).
- **MatrixAgent**: super class of all agents represented in MATRIX. Implements a JADE agent.
- **EventReaderBehaviour**: represents a cyclical behaviour that reads events sent by the device, modifying the rule engine's memory.
- **JESSBehaviour**: this is a class representing the behaviour that demonstrates the integration between JADE and JESS. It is a cyclical behaviour instantiated with a JESS rule file and wrapping the rule engine.

3.3 Example of Use Cases

In this subsection we include two examples of MATRIX's adaptation in order to obtain a given functionality. For each case, we detail which OSGi bundles should be developed and which rules would lead the behaviour of the agents in MATRIX. For this purpose the rules have been coded in pseudo-code, not in JESS, to make the reading more easy.

The first example describes a hypothetical communication that could take place between a stopped vehicle and a non-mobile device. The car driver is looking for a hotel with a price not exceeding some threshold and situated *near* his GPS position. Table 1 shows the software elements needed in our system, its location and a brief description of its mission.

The second example describes a vehicle–vehicle communication where one of the cars has suffered an emergency situation and warns the nearest available ambulance. As in case of the previous scenario, Table 2 shows the software elements needed in MATIX, its real location and a brief description of how the component must operate inside the system.

Table 2: Software elements to implement in order to adapt MATRIX to the functionality of *looking for ambulances after an emergency takes place*.

Element needed	Location	Description
Event Provider Bundle 1	Vehicle, Ambulance	Generates an event with the GPS position to be sent periodically to the agent through the Communicator Bundle.
Event Provider Bundle 2	Vehicle	Throws an event if an emergency situation occurred (i.e.: airbag is activated).
Rules	Vehicle agent	***If*** *event type is emergency* ***then*** *it searches for nearest agents of type ambulance sending them a help message*
Rules	Ambulance agent	***If*** *message type is "help request"* ***then*** *sends an event to a mobile device with the required information.*

4 CONCLUSIONS

Nowadays, car industry spends a lot of effort to develop new software systems, as well as new standard and development models. According to Stavros Stefanis - director of IBM's Embedded Systems Lifecycle Management - in 2010 the 90% of vehicle innovations will be related with software. According to this idea, the development of (*i*) new communication systems between vehicles or vehicles and infrastructures and (*ii*) the development of new telematic services will be necessary in the near future.

The main contribution of the present work is the improvement in the administration of all the information that current cars are generating within a centralized system, which manages and stores this information. The purpose of the system is not only to improve the communication between vehicles or vehicle-infrastructure, but also to provide services to other applications that need it.

The future work is focused in the development of new services in the context of the CTAG (*Technological Automotive Centre in Galicia*) active projects. Examples of these applications are: remote diagnosis, remote software updates, modification of engine settings, navigation, theft tracking, automatic log book, car pooling and traffic information. In all of this application areas, MATRIX will play a crucial role using all the information generated from this centralized system.

ACKNOWLEDGEMENTS

We are deeply indebted to the personnel at the Technological Automotive Centre in Galicia (CTAG), who made this research work possible by providing useful information of state-of-the-art technologies and allowing us access to all kind of facilities.

REFERENCES

ANFAC, 2006. *Vehicle registration Statistics.* Retrieved November 2, 2006 from http://anfac.com/global.htm.

Apache Software Foundation, 2006. *Apache Felix.* Retrieved November 7, 2006 from http://cwiki.apache.org/FELIX/index.html.

AUTOSAR, 2006. *Automotive Open System Architecture*. Retrieved September 20, 2006 from http://www.autosar.org.

Car2Car, 2006. *Car 2 Car Communication Consortium*. Retrieved November 1, 2006 from http://www.car-to-car.org.

CE, 2003. *Paper to the European Council and Parliament about information and communication technologies bringing the car security and services*. Brusels, 15.9.2003. COM(2003) 542 final (SEC(2003) 963).

Chen, K., Gong, L., 2001. *Programming Open Service Gateways with Java Embedded Server Technology.* Addison-Wesley Professional.

DGT, 2006. *Spanish Traffic Department*. Retrieved October 10, 2006 from http://www.dgt.es.

Equinox, 2006. *Eclipse Equinox.* Retrieved November 2, 2006 from http://www.eclipse.org/equinox.

e-Safety, 2006. *Europe's Information Society*. Retrieved November 9, 2006 from

Friedman-Hill, E., 2003. *JESS in Action.* Manning.

http://ec.europa.eu/information_society/activities/esafety/index_en.htm

JADE, 2006. Java Agent DEvelopment framework. Retrieved September 15, 2006 from http://jade.tilab.com/.

Knopflerfish, 2006. *Knopflerfish – Open Source OSGi*. Retrieved October 10, 2006 from http://www.knopflerfish.org.

MISRA, 2006. *The Motor Industry Software Reliability Association*. Retrieved 11 September, 2006 from http://www.misra.org.uk.

OSGi, 2006. *Open Services Gateway initiative*. Retrieved October 28, 2006 from http://www.osgi.org/.