Service Oriented Middleware for Automotive Applications and Car Maintenance

Guido Gehlen, Erik Weiß
RWTH Aachen University,
Communication Networks
Kopernikusstr. 16, 52074 Aachen
{guge, erw}@comnets.rwth-aachen.de

André Quadt
RWTH Aachen University,
Research Institute for Operations Management
Pontdriesch 14/16, 52062 Aachen
qu@fir.rwth-aachen.de

***Abstract*— The After Sales Market of the Automotive Industry is an important market in Europe. The changes in the block exemption regulation by the European Union in 2002 open up new business opportunities in this market by stipulating the automotive industry to offer the information they offer to their dealers also to independent organizations.**

Based on the new requirements of the block exemption regulation the IST-MYCAREVENT project will develop new applications and services, which can be seamlessly and securely accessed by mobile clients. To cope with the challenge of a heterogeneous and mobile environment, a communication and application middleware has to be developed. One proposal is based on the Service Oriented Architecture (SOA) and Web Service technologies.

This paper introduces a middleware for vehicular applications in common, directed to the support of the roadside assistance, the workshops, and the driver in a breakdown situation.

I. Introduction

The After Sales Market of the Automotive Industry has become a very important market. The introduction of innovative mobile applications will enable new ways of working and collaboration among car manufacturers, workshops, road assistance services and the customer. In wake of the new developments in block exemption regulations the different service providers have the right to access different kinds of repair information, training material and tools, which earlier have only been provided to franchised dealers and garages of the respective vehicle manufacturers.

The IST-MYCAREVENT project aims at developing an innovative system which utilizes the state-of-the-art technologies to support the mobile user and worker in this area and to establish profitable business models to foster this important market sector. This paper discusses the gain of current and near future technologies in the mobile communication and application domain of MYCAREVENT.

Automotive applications in common and especially the MYCAREVENT applications are one of the most challenging applications for mobile communication systems and mobile application designers. The mobility level varies from stationarity to velocities of 100 km/h and more. Different mobile and wired communication system could be used, like e.g. an Ethernet connection in the workshop, GSM/GPRS, UMTS, or WLAN on the road, or many communication systems in parallel.

The computing device environment is very heterogeneous, since there are various car manufactures, independent and manufacture dependent roadside assistants and the drivers themselves. Each of them has different devices. The car equipment ranges from cars without any computing device to cars with a diagnostic system, navigation system, and entertainment system. The roadside assistant may be equipped with one or more diagnostic tool, a laptop, or a Personal Digital Assistant (PDA). The driver may have a private PDA, a smartphone, or a mobile phone.

Taking these requirements into account, this paper introduces a middleware which will provide application developers a flexible and device/OS independent platform to develop services and applications for automotive applications in the car maintenance sector.

II. MYCAREVENT

MYCAREVENT will develop and implement new applications and services, which can be seamlessly and securely accessed by mobile devices. They will provide manufacture specific car repair information according to the problems identified by the Off-/On-Board-Diagnostic systems. This information needs to be presented in a multi-lingual way. The mobile worker needs to access information stored within the manufacturers IT system. Thus, he must interact with the service portals of the car manufacturers and independent services while performing the repair of the car. MYCAREVENT will foster the aims of the new block exemption and will support the worker in coping with the increasing complexity of cars and their repair procedures. The project will also provide 'Telematic' support for car drivers with self-services in situations where a little advice or a software fix is all that is needed.

Figure 1 illustrates the different domains of the MYCAREVENT project which are using mobile communication systems and mobile applications on the left-hand side and the stationary domains on the right side. Between these domains a mobile communication link, the so called INtelligent COnnection (INCO), connects the mobile environment with the stationary one. The INCO represents a virtual link which could

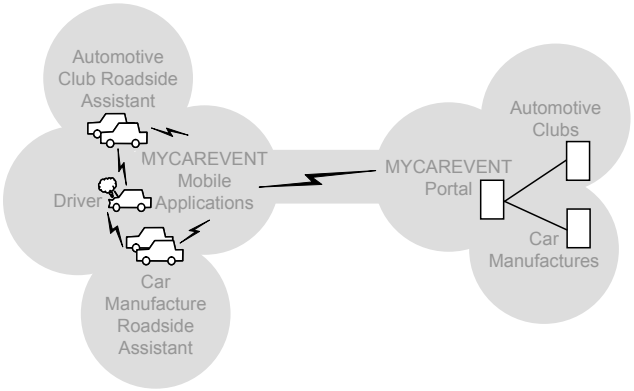

Fig. 1. Overview of the mobile service world of MYCAREVENT

be realized by one or more real communication links, like a GSM/GPRS, UMTS, or WLAN link.

All use cases in MYCAREVENT will be differentiated into three Pilot areas. Pilot Area 1 deals with the manufacture specific car maintenance. Either a roadside assistant of the car manufacture will be send to the breakdown scene or the driver is still able to drive his vehicle to the franchised workshop and will it have repaired there. Opposed to this, Pilot Area 2 deals with the cases where an independent roadside assistant will arrive at the broken down car or an independent workshop will conduct the repair job. Both Pilot Areas are together illustrated in figure 2.

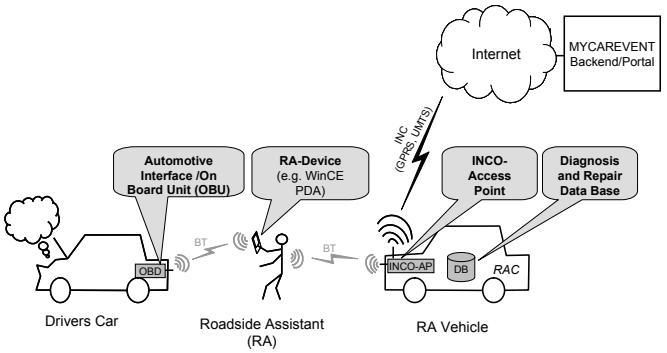

Fig. 2. Pilot Area 1 and 2

The roadside assistant (RA) is equipped with a mobile device and might be able to read the diagnostic data from the damaged car. Otherwise, he will type the symptoms (visible damages, but also smells, noises etc.) of the car into his interactive application. All data is transmitted over the INCO to the MYCAREVENT portal, which analysis the data and triggers further events, like the submission of an electronic and interactive repair instruction or circuit diagram.

If either the damaged car or the driver himself have a long range communication device, error messages or a symptom description can be transmitted from the driver resp. the driver's car to the MYCAREVENT portal. This first information about the situation of breakdown can be used to trigger the right event for the roadside assistant or may be used to instruct the driver on how to repair the car on his own. This is valid for trivial repairs and covered in MYCAREVENT by Pilot Area 3. If the detected fault could not be recovered by the driver, a roadside assistant can be sent to the scene and equipped in respect of the special demands of the damaged car.

As mentioned above, the Focus Area 3 deals with a driver self-help scenario, see figure 3. If the driver or the car detects a fault of the car, he advices to transmit a fault report to the MYCAREVENT portal. Alternatively, if the car is not equipped with communication and diagnosis capabilities, the driver can send a symptom description to the portal. For Pilot Area 3, the system decides, after the fault/symptom analysis, to start the drivers selfhelp and prepares a repair instruction for the driver.

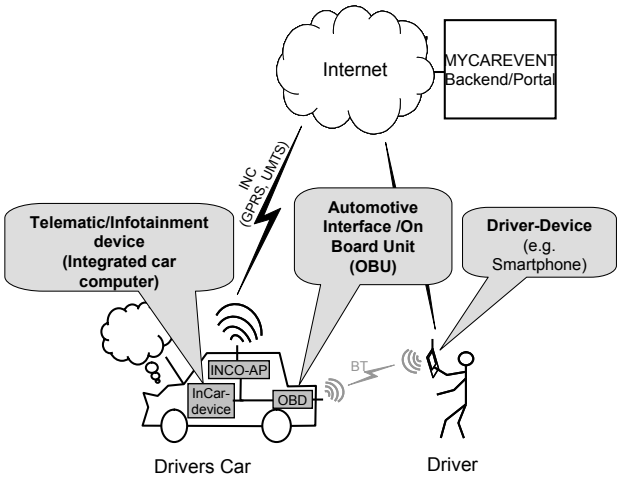

Fig. 3. Focus Area 3 ...

III. Automotive Middleware Architecture

To cope with the challenge of this mobile and heterogeneous environment, a middleware is proposed which lean against the Service Oriented Architecture (SOA), see section IV. The main elements in this architecture are the services, which are atomic software components with a well defined and network addressable interface.

The development process of the mobile applications is geared to the Model Driven Architecture (MDA), see section V. The MDA iteratively maps an application model to a specific platform until the lowest level platform (here the operating system) is reached. In our middleware proposal the first iteration maps the application model to the SOA in general, then to Web Services, and in the last step to a Java, .Net Compact Framework (CF), or Symbian OS for mobile devices, as illustrated in figure 6.

The SOA could be realized by different technologies, like CORBA, DCOM, or XML Web Services. The middleware is supposed to use XML Web Services, since there is a big support by the IT industry, e.g. Microsoft, IBM, and Sun by integrating XML Web Service functionalities into their

development and runtime environments. Another advantage of XML Web Services is the link to the application ontology by using standardized XML data formats and linking the exchanged XML messages to the standardization in terms of referencing the corresponding XML schema.

In general, the proposed middleware is divided into two layers, a communication layer and a service/application layer, see figure 4. The communication layer provides a secure and reliable link using different mobile communication systems. The service layer on top mainly manages the provision, publication, and discovery of services. The Device Management, Digital Rights Management, Context Management, Authentication/Authorization, and Location Management are additional features.

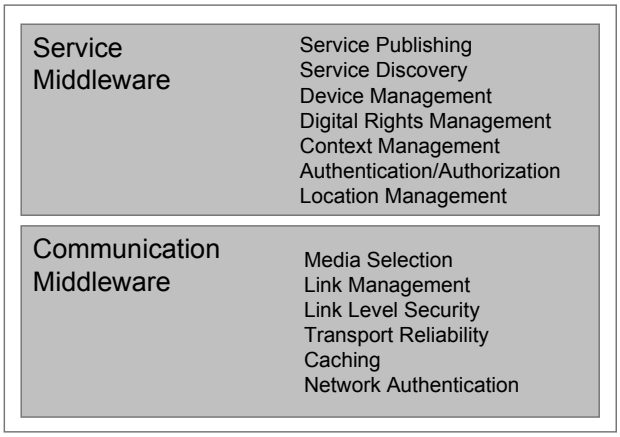

Fig. 4. The proposed MYCARVENT middleware segmentation

The separation of the middleware into two layers has different advantages. The development of the middleware will be simplified, since the communication experts and service experts can develop independently. The middleware will be scalable, since one layer could be replaced without changing the other layer, e.g. a full featured communication layer can be replaced by a cheaper lightweight communication layer. This could be an important criteria for a later product development.

To realize this scalable two layer approach, the interface between the communication and the service layer has to be defined clearly. We propose a well standardized and widely used IP based interface. This enables et al. the distribution of both middleware parts on different devices if necessary.

The interface of the service middleware to the application should also be well defined to enable application developers to implement applications without a deeper knowledge of the middleware. We propose a Web Service interface, since all technologies are open and platform independent and several development environments are supporting Web Services.

Some examples are given, to clarify the approach:

A) The communication and service middleware is deployed on the device of the roadside assistant, e.g. a laptop or a PDA. This solution is cost efficient, but may lacks in performance and useability, since only standard consumer hardware is used.

B) The communication middleware is separated on an embedded device inside the car and the service middleware on a different device (laptop or PDA). Thus, the communication device can be adapted to the car environment, e.g. by integrating advanced antennas within the car.

IV. Services Oriented Architecture

Object oriented software architecture is hierarchically structured to build complex and reusable software. On the lowest level, functionalities are encapsulated in an object. A set of interacting software objects is collected into a component. The Service Oriented Architecture (SOA) [1] consequently extends this hierarchical structure to distributed systems. The interaction of services does not take places in one application on one device, but services reside on different heterogeneous systems and collaborate over communication systems. Services have the following characteristics [2]:

- Services are self contained and modular
- Services are discoverable and dynamically bound
- Services stress interoperability
- Services are loosely coupled, reduction of artificial dependencies to a minimum
- Services have a network-addressable interface
- Services have coarse-grained interfaces in comparison to finer-grained interfaces of software components and objects
- Services are composable

These characteristics should make the distributed system as simple as possible, but no simpler. In other words, the SOA is an architectural style to achieve loose coupling among interacting software agents and to minimize their artificial dependencies.

The SOA defines three roles, a Service-Requestor (R), Service-Provider (P), and a Service-Broker (B). A software agent interacting with other software agents plays one or more roles. They communicate in the way as depicted in figure 5.

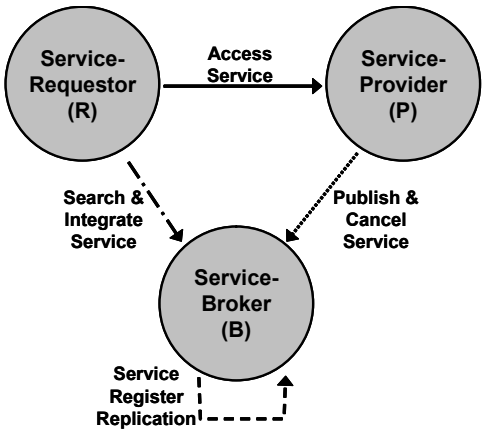

Fig. 5. The Service Oriented Architecture (SOA) roles and relations

Providers publish their services to a service registry (service-broker). More than one service-broker within the environment have to replicate their service registries (dashed arrow). Requesters use the Broker to search for services and integrate them by accessing the service description (dashed-dotted arrow). This description includes all information needed to access the service and is used to generate a service-proxy object. The service-proxy represents the remote service, i.e. all published remote service methods are methods of the local proxy object. This architecture bridges the native messaging inside the client environment to the platform independent messaging in the SOA environment. The process of proxy object generation maps the platform independent description into a real software object for the client's system environment.

To achieve high interoperability, all SOA entities have to use a common language for service description, messaging and service registration. The Extensible Markup Language (XML) is such an appropriate common language. On the basis of XML the World Wide Web Consortium (W3C) has specified a middleware framework, called Web Services, following the SOA. For messaging the Simple Object Access Protocol (SOAP) [3], [4] is used. It is based on standard internet protocols like HTTP or other arbitrary protocols like the Wireless Application Protocol (WAP) [5] or the Block Extensible Exchange Protocol (BEEP) [6]. The SOAP envelope is structured in XML. Interfaces are described in a XML subset, the so called Web Service Description Language (WSDL) [7]. This description includes all the information needed in order to invoke service methods from other nodes.

V. Model Driven Architecture

As we have mentioned, mobile distributed applications and the underlying middleware have to be platform independent and cross-platform interoparable. This takes the heterogeneity of mobile devices and their operating system and programming languages into consideration. The MDA [8], specified by the Object Management Group (OMG), is intended to support platform independency, interoperability and productivity.

In the past the abstraction level of software development has been raised by moving from assembly languages to third generation and object oriented languages like C++, C#, and Java. The MDA is a software development architecture, which raises the level of abstraction by using modelling languages as programming languages. Besides, the abstraction level of models can iteratively be reduced, starting with a high abstract model and ending with an implementation model.

Within the MDA different developers, like application developers, platform experts, and domain experts, build platform-independent models that are processed by model compilers (generators). A platform-independent system model can be mapped to different platforms and systems, or different parts of the system could be mapped to varying platforms. At every mapping to a platform, the level of abstraction is reduced.

The MDA differentiates between three viewpoints on a system, a computation independent viewpoint, a platform independent viewpoint and a platform specific viewpoint. Consequently, the system has three different models, the Computation Independent Model (CIM), the Platform Independent Model (PIM), and the Platform Specific Model (PSM), which represent a view of the system from the corresponded viewpoint.

A system can be modelled by more than one PIM and PSM. Each of these models is related to a specific platform. The platform is a set of subsystems and technologies that provide a coherent set of functionality.

The MDA is illustrated by designing a mobile distributed application with SOA, Web Services and Java2 Micro Edition (J2ME). The mobile distributed application can be modelled in four steps, as depicted in figure 6. The first PIM of the mobile distributed application is deducted from a CIM including the Business Model and the Domain Model. This high level PIM specifies the overall architecture, the communication mechanisms and the algorithms independent of a special platform.

This high level PIM is mapped using the SOA into a PSM based on the SOA. At the same time, this PSM is the PIM for the next step. This step will map the PIM of the mobile SOA application to the PSM of a Web Service Platform.

In the last step, this model is mapped to the final PSM on a J2ME platform. It is the lowest level model since this model can be mapped into executable code and deployed on the devices.

Each model can be used to start a new branch to an alternative platform, e.g. the PIM of the mobile SOA application can be also mapped to a Common Object Request Broker Architecture (CORBA) platform, or the PIM of the mobile Web Service application can be mapped into a .Net platform for MS Smartphones. The PIM will remain unaffected, only a new mapping in terms of a model compiler have to be developed by platform experts.

In the proposed MYCAREVENT middleware the MDA can be used to align the development of applications and services for many heterogenous devices on top of the two layered middleware approach. A PIM will be mapped in a first step to the MYCAREVENT middleware and in a last step to a specific device OS.

VI. Conclusion

This paper describes the IST-MYCAREVENT objectives in respect to the mobile communications and applications and a proposal of a middleware. The middleware copes with the requirements of the heterogeneous and mobile automotive environment. For the design of the middleware and applications the SOA and MDA is proposed.

Acknowledgment

The work presented has been motivated by IST-MYCAREVENT[1] project. The authors would like to thank Prof. B. Walke and C. -H. Rokitansky of ComNets for their help and support, as well as all partners of the project.

[1]Information available at http://www.mycarevent.com

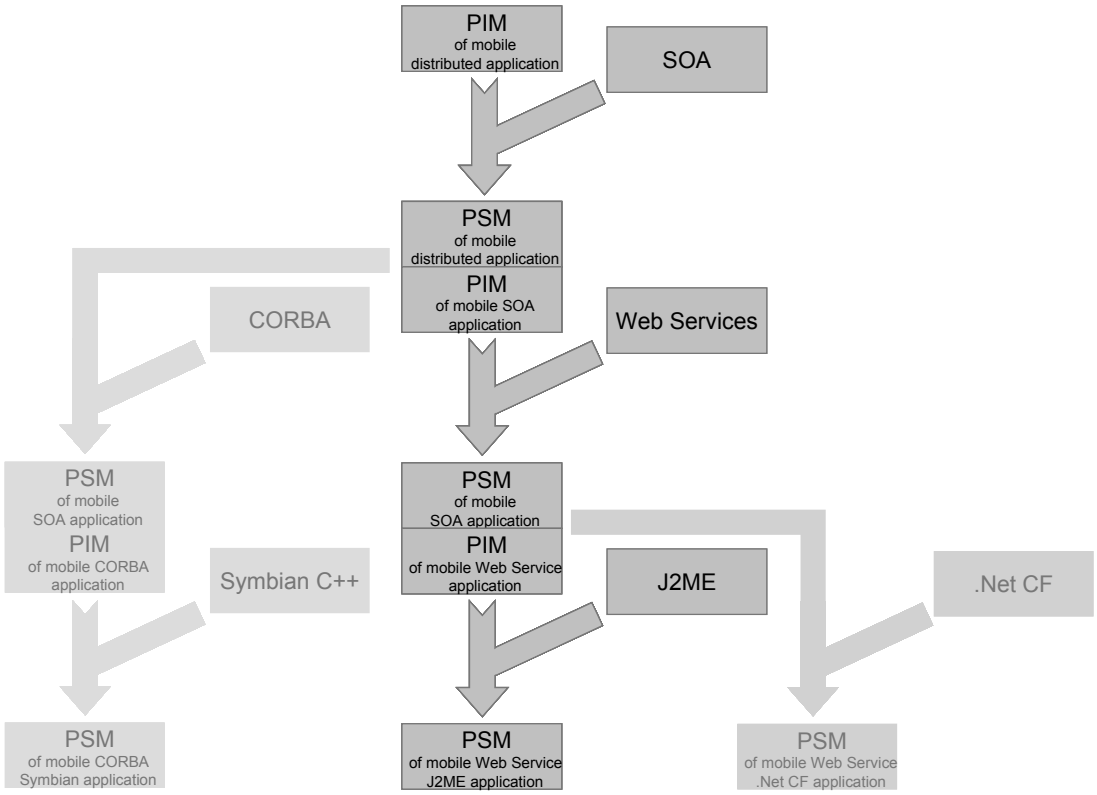

Fig. 6. MDA approach of the middleware design

REFERENCES

[1] "Web service architecture," Published on the internet. Available at http://www.w3c.org, 2004.

[2] J. M. Govern, S. Tyagi, M. Stevens, and S. Mathew, *Java Web Service Architecture*. Morgan Kaufmann, 2003.

[3] N. Mitra, "Soap version 1.2 part 0: Primer," Published on the internet. Available at URL http://www.w3.org/TR/soap12-part0/, June 2003.

[4] T. Clements, "Overview of soap," Published on the internet. Available at URL http://developer.java.sun.com/developer/technicalArticles/xml/webservices/, Jan. 2002.

[5] G. Gehlen and R. Bergs, "Performance of mobile web service access using the wireless application protocol (wap)," in *Proceedings of World Wireless Congress 2004*. San Francisco, USA: University Aachen, Communication Networks, 05 2004, pp. 427–432.

[6] M. Rose, "Rfc-3081: Mapping the beep core onto tcp," Published on the internet. Available at http://www.ietf.org/rfc/rfc3081.txt, Mar 2001.

[7] R. C. et al., "Web services description language (wsdl) version 1.2," Published on the internet. Available at URL http://www.w3.org/TR/wsdl12, June 2003.

[8] J. Mukerji and J. Watson, "Overview and guide to omg's architecture," Published: www.omg.org/mda, 2003.