A M2M-based Automotive Service Delivery Platform for Distributed Vehicular Applications

M.Glaab[1,2], W.Fuhrmann[1], J.Wietzke[1] and B.V.Ghita[2]

[1]In-Car Multimedia Labs, Faculty of Computer Science,
University of Applied Sciences Darmstadt, Darmstadt, Germany
[2]Centre for Security, Communications and Network Research,
Plymouth University, Plymouth, United Kingdom
e-mail: markus.glaab@h-da.de

Abstract

Modern vehicles are connected with the Internet through a range of wireless cellular network technologies. This provides the basis for many novel applications and use-cases towards intelligent vehicles, offering enhanced vehicle safety, traffic management, and driver and passenger convenience capabilities. The connectivity further enables new distributed software architectures that can provide solutions to existing challenges in the field of automotive software engineering. This paper introduces the approach of an Automotive Service Delivery Platform, based on the Machine-to-Machine Communication Service Architecture. Experiences and findings are presented, gained by implementing core elements of the proposed architecture prototypically. This implementation makes apparent the range and capabilities of current Machine-to-Machine Service Architecture and describes remaining challenges to facilitate efficient distributed automotive services.

Keywords

Machine-to-Machine Communication, Automotive Software Engineering, Resource-Oriented Architecture, Service Delivery Platform, Data Opaqueness

1. Introduction

Driven by the recent progress of Consumer Electronics (CE) devices like smartphones and tablets, the customers' demands for functionality, customisability, and connectivity of their In-Car Multimedia (ICM) system is continually growing. Besides, in the context of Intelligent Transportation Systems (ITS), governments, standards development organisations (SDO), and engineers have been spending lots of research to enhance the traffic safety and efficiency for many years.

Connecting vehicles with the Internet is the foundation for these visions, and the number of cars, equipped with GPRS, UMTS, and LTE hardware, already increases. Although further advancements within wireless cellular network technologies, network protocols, and automotive embedded hardware are necessary, adequate automotive software (SW) architectures and platforms are the key to let vehicles become an integrated part of the Internet and to make them *intelligent* or *smart*. Use-cases and applications of former non-automotive domains must be integrated and formed into a homogenous overall system (Bauer, 2010). But the automotive

industry, and e.g. the consumer electronics and communication industry have different performance or safety requirements, as well as lifespan and innovation cycles. For instance, vehicle models are usually produced seven to eight years and they have to be maintainable for at least 15 years after the purchase. In contrast, the lifecycle of hardware, e.g. CPUs, is less than five years (Broy et al., 2007). More frequently, and even during the production phase, OEMs may want to integrate new software features (e.g. of other vehicle series). This might already be influenced by the innovation cycle of CE-software, but the latter is even shorter (Shimizu, 2004). Many social network smartphone-applications, such as Facebook and Spotify, are updated within days to a few weeks. In contrast, the software of the vehicle is only updated during service in a garage. Although the general mechanisms do exist to perform Over-The-Air (OTA) updates, the current automotive software architectures are not adequate to implement this securely with respect to possible side-effects and compatibility (Pretschner et al. 2007). This is intensified by the huge number of possible variants of a vehicle, not only regarding HW and SW revisions, but also with respect to configuration possibilities for car equipment. The issues of today's automotive SW engineering is also emphasised by this fact: While the functionality from one vehicle generation to the next in many sub-domains only differs by 10% due to enhancements and changes, more than 90% of the software is rewritten (Broy, 2006). This is caused by low level, hardware specific code that is hard to change or port (Broy, 2006). Even though the software-related challenges can be largely solved, hardware limitations continue to exist. Due to the harsh environment of the automotive domain, with wide temperature and humidity ranges, and special requirements on shock resistance, specialised embedded hardware has to be used. They are usually less powerful, compared to CE, and usually more expensive. However, it can be assumed, that the implementation of new applications during lifetime into a "traditionally designed headunit", where all functionality is truly installed, will be limited by hardware constraints – similar to todays' CE, hardware upgrades or replacements must be taken into account.

Facing these challenges, a new architecture paradigm, including a substantial proportion of automotive applications implemented outside the vehicle as services residing on OEM servers (Glaab et al., 2010) might be valuable and constitutes the basis for the presented architecture. It is introduced in Section 2 in more detail, concluded with the motivation of an Automotive Service Delivery Platform (ASDP). Section 3 discusses architecture fundamentals, if such ASDP should be realised based on the M2M Service Architecture. Finally, the applicability of M2M is evaluated within section 4 by discussing results and experiences, gained during prototypical implementation of core elements at the ICM lab. Section 5 briefly summarises the findings of the paper and provides an outlook on future work.

2. Towards an Automotive Embedded Internet

As indicated, offloading of automotive functionalities ("intelligence") to servers is a promising approach for the next generation of automotive software architectures. Figure 1 shows the general architecture, and involved components/domains: The OEM headunit is the central component of the vehicle, functionally connecting the displays, sensors, actuators, etc. The headunit is connected via wireless cellular networks with an OEM server, located within the Internet domain ("Cloud"). This in

turn might connect to 3rd party servers. Accordingly, "the clash" of different domains, with quite heterogeneous requirements, lifecycles, etc., should occur at the OEM server, where it is anticipated that they can be mitigated more easily.

This is expected to reduce the hardware requirements for the headunit. Furthermore customising and adding of new functions during the lifetime of the vehicle does not raise the hardware requirements of the headunit as much, as integrated approaches. But, not every automotive application is suitable to be transformed to a web service in the same way. Criteria are needed for profound decision whether functions should (still) be realised within the vehicle, or if it is advantageous to transfer them as a service on a web server (Glaab et al., 2011). In particular resulting requirements against the wireless access networks have to be considered, because they can be treated as the bottleneck of this approach. Finally, it has to be reflected that vehicles can transit areas with no coverage. Consequently the remaining functionality during connectivity-loss has to be well-considered. However, since many of the aforementioned future automotive functionalities need data connectivity anyway, it can be expected that the number of suitable applications for cloud-based realisation increase above average.

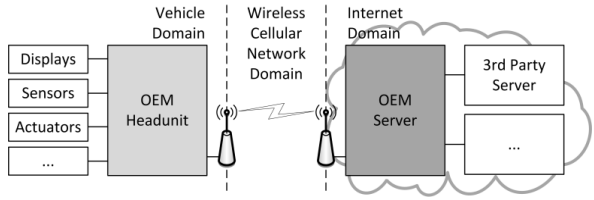

Figure 1: General architecture of distributed automotive software platform

Automotive software engineers need an end-to-end (E2E) solution, which extends their design space for the implementation of applications from the vehicle to the OEM server. We name this architecture an Automotive Service Delivery Platform (ASDP). According to our approach it should meet the following requirements: It shall offer appropriate open and standardised interfaces, and a modular design to enable re-use of common functions, following a Service-Oriented-Architecture (SOA). Resulting communication capabilities shall offer appropriate mechanisms for designing and controlling of data flows, and to prevent network-misalignments with respect to functional split and scalability. It shall offer capabilities for mediation and adaption of services. Since future vehicles, as part of an ASDP, will consume and also expose services, they should not only be treated as connected, but as an integrated part of the Internet towards an (automotive) Embedded Internet (Wu et al., 2011).

3. An Automotive Service Delivery Platform based on the M2M Service Architecture

Machine-to-Machine Communication (M2M), also known as Machine-Type-Communication (MTC), has been selected as the technology for realising an ASDP, while meeting the above listed requirements. There is no complete M2M architecture

defined at the moment, as the preferred route has been to define M2M as a collection of functionality blocks, developed by different research institutes, companies and SDOs in particular European Telecommunications Standards Institute (ETSI) (ETSI, 2013a), 3rd Generation Partnership Project (3GPP) (3GPP, 2013) and Open Mobile Alliance (OMA) (OMA, 2013). The main standardisation activities have been bundled in oneM2M (oneM2M, 2013) for global harmonisation since July 2012.

3.1. Functional Architecture

The current ETSI M2M Service Architecture (ETSI, 2013b) specifies three types of components: *M2M Device* (D), *M2M Gateway* (G), and *M2M Network* (N). D and G are located inside the *M2M Device (and Gateway) Domain* and are connected by using wireless access networks to the *Network Domain*, where N is located. This allows hierarchical structures, where several D connect to one G and several G connect to one N, which emphasises the need for scalability in the context of millions of devices. With respect to an ASDP, the D is the vehicle and N is the OEM Server. A Gateway is currently not part of the considerations, as the vehicle has been decided to be M2M-compliant and it hence is able to connect directly to N.

In contrast to the currently widespread silos of vertically integrated applications, which are caused by the strive for the ultimate "killer application" (Wu et al., 2011), M2M has been developed as an open, horizontal, and hence more universal, integration platform. Thus, the *Service Capability Layer* (SCL), including the *Service Capabilities* (SCs), has been introduced within every M2M component, in order to encapsulate functions that are to be shared by many *M2M applications* (xA), which thus should only contain the business logic. Currently 11 SCs are proposed (ETSI, 2013b): *Application Enablement* (xAE), *Generic communication* (xGC), *Reachability, Addressing, and Repository* (xRAR), *Communication Selection* (xCS), *Remote entity management* (xREM), *SECurity* (xSEC), *History and data retention* (xHDR), *Transaction management* (xTM), *Compensation broker* (xCB), *Interworking proxy* (xIP), and *Telco operator exposure* (xTOE). The x is a placeholder for the component, in which SCL the SC is implemented. If the xAE, for example, is located on the D, it is called DAE, a xGC within the Network is called NGC, etc.

ETSI has defined four reference points (interfaces): *dIa* (vertical between DSCL and DA), *mIa* (vertical between NSCL and NA), *mId* (horizontal between D and N), and *mIm* (horizontal between N and N). Figure 2 depicts the compounded functional architecture, instantiated according to our proposed M2M-based ASDP.

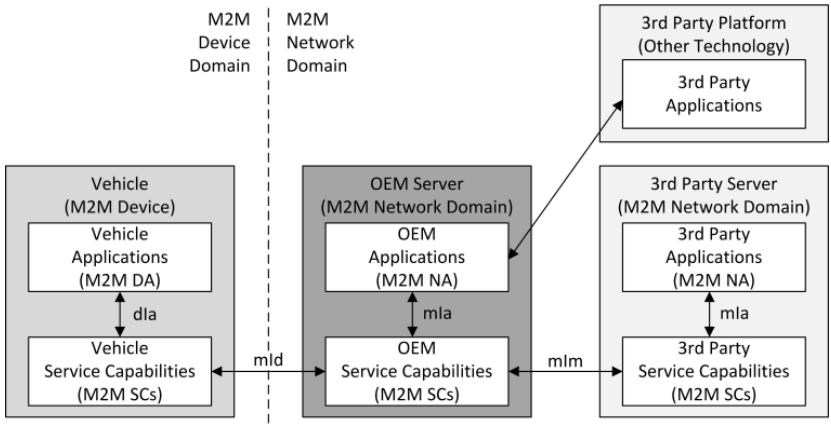

Figure 2: Compound functional Architecture of an M2M-based ASDP Resource Organisation and Management

Intrinsic to the M2M Service Architecture are that the resource organisation and related management procedures are following the RESTful architectural style, as defined in Fielding (2000). This style is particularly suitable for M2M communications (Pautasso et al., 2008).

A generic, hierarchical structured, *Resource Tree* is located inside each SCL for collaboration and exchange of applications, data, and SCs on the D, G, and N. It "describe[s] how the different resources relate to each other [, and it is introduced to] improve the overall system performance through the use of minimal structured data." (Boswarthick et al., 2012, p.127). The subtree within Figure 3 indicates the general structure of the *Resource Tree*. Several resources, e.g. data *containers* recur on different levels of the *Resource Tree*, which is used to model their scope.

The *Resource Tree* is mapped to Uniform Resource Identifiers (URIs) and manipulated via the RESTful reference points *dIa*, *mIa*, *mId*, and *mIm* by the four basic CRUD methods, the so-called "verbs": *CREATE*, *RETRIEVE*, *UPDATE*, *DELETE* and might be extended through *NOTIFY* and *EXECUTE* (ETSI TS 102 690 2013). These methods can be mapped to the RESTful application layer protocols, most likely HTTP (Hyper Text Transfer Protocol). Recently the Constraint Application Protocol (CoAP) was developed, which is especially designed for the RESTful communication of very limited electronics devices (Shelby et al., 2013; Bormann et al., 2012), but it might also be valuable for an ASDP.

4. Evaluation and Discussion

Core elements of an M2M-based ASDP have been prototypically implemented. To evaluate the capabilities of the current ETSI M2M Service Architecture and its applicability to an ASDP, as envisaged within our research, exemplary use-cases have been mapped according to the basic architecture presented above. Three of them are briefly presented below, but for the ease of understanding, the further evaluation and discussion is basically continued with the first use-case.

1. Floating Car Data (FCD) / Extended Floating Car Data (XFCD)
FCD describes vehicles that are used as driving sensors, periodically reporting at least their current location together with the timestamp to the OEM server. The trigger for the reporting might be time-related, distance-related, or a combination of both. The OEM server aggregates and analyses the data from all vehicles. Appropriate traffic algorithms and models enable the detection of traffic jams and average travelling times, which in turn can be used for enhanced route guidance purposes. XFCD may transmit additional data to the OEM server, such as rain sensor values, light sensor and hazard lights status, outside temperature, or vehicle dynamics data gained from active driver assistance systems. This data enables advanced inference of traffic safety and efficiency on a specific route. For instance, icy roads, heavy rain, or emergency braking can be determined and propagated to other vehicles that are approaching the relevant area.

2. Vehicle Maintenance
Modern vehicles have variable service intervals, depending on their usage, which is monitored over time, to estimate when thresholds are exceeded and service is necessary. Besides, various sensors and check routines may detect individual component failures. These are currently only locally stored using a fault recorder and manually readout at the car service station. M2M should enable use-cases, where relevant data can be submitted to the OEM server periodically, or event-/failure-/based. The gathered data may be subsequently used to initiate a separate business process of contacting the vehicle owner, discuss necessary service amounts, and arrange workshop dates, etc. Furthermore, it might be used for quality management and product improvements.

3. Enhanced Navigation, Social Driving, Intelligent Vehicles
Assuming the addition of an online navigation system, which calculates the routes on the OEM server, to the ASDP. In such a scenario, the server knows the destination, the route, and maybe even upcoming trips (through access to an online calendar). Additionally with XFCD and vehicle maintenance, the system also has information about the current tank level, remaining distance, and average economy. Based on these data, combined with statistical analysis of historical gas price data regarding cities, day, time, it can provide optimal suggestions for intermediate refuelling stops. The importance of such use-cases increases especially regarding electro mobility, where charging stops may require more comprehensive planning, with respect to minimised range, charging time, power plant capabilities etc.. Enhancing the calculation with these additional constraints requires only service advancements on the OEM server and no vehicular software updates, or considerable improved wireless access network capabilities. Equally, a car-pooling service can be added just by connecting it to the OEM server, since the necessary data (e.g. driving destination, route, and current position) is already available, because of a basic application like the navigation service.

4.1. Evaluation

The M2M Service Architecture in general facilitates the transparent transport of data between a D (vehicle) and an N (OEM server). DA and NA never exchange data directly, but via their local SCLs. With the “Announcement” and “Subscription”

mechanisms of the ETSI M2M Service Architecture it is possible to fully or selectively transfer data from D to N and vice versa through the *Resource Tree*, SC(L)s, and RESTful communication. As a result a somehow vehicle' ("vehicle stub") arises within the NSCL, and an OEM server' ("OEM server stub") within the DSCL. Therefore local applications (lA) can consume data out of their local SCL (lSCL), originally generated by a remote application (rA), which spans the design space for data exchange, distribution, storage, wireless access network requirements, and connectivity handling.

Since vehicular sensor data is the foundation for many automotive-related applications, it should be made available to DAs and NAs within the ASDP. Accordingly, a DA "Vehicle Data Provider" has been introduced, to make location data (e.g. latitude, longitude, height, heading, speed) and sensor data (e.g. water temperature, oil temperature, service status, tank level, average economy) available in the local resource tree through tailored data containers such as XML. For this reason, the application may gets the vehicle data e.g. from an external source (such as a Controller Area Network fieldbus (CAN-bus)), and processes it accordingly. Besides, a NA "Floating Car Data" is introduced, to implement the FCD business logic. Figure 3 provides an architectural view of the resource structure and data flow.

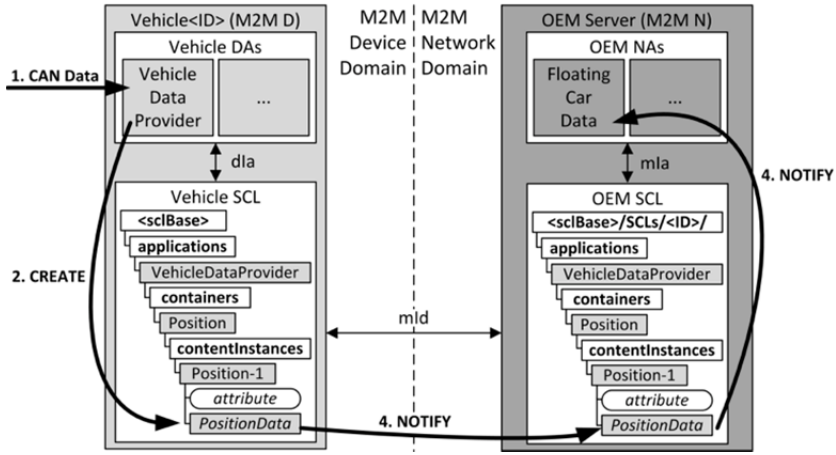

Figure 3: VehicleDataProvider-DA to FloatingCarData-NA Data-Flow Example

It is assumed, that all bootstrap and registration procedures of DA, DSCL, NSCL, and NA are successfully completed. Further, it is assumed, that necessary subscriptions from NA to NSCL, and NSCL to DSCL, are successfully created. Accordingly, the basic steps of a *VehicleDataProvider-DA* to *FloatingCarData-NA* data-flow are:

1. The *VehicleDataProvider-DA* reads the vehicle data (position) by use of its CAN-bus interface.

2. The position data is added to the resource tree, using a CREATE call of an appropriate *contentInstance: Position-1*.

3. Because of the existing NSCL-DSCL subscription, a NOTIFY from the DSCL to the issuing NSCL resource is sent, which contains the actual representation of the *contentInstances* resource.

4. Because of the existing NA-NSCL subscription, a NOTIFY from the NSCL to the issuing *FloatingCarData-NA* resource is sent, which contains the actual representation of the *contentInstances* resource.

The *FloatingCarData-NA* can process the vehicle's *PositionData* sample together with prior samples of this vehicle, and other samples of other vehicles, considering the respective positions and timestamps to fulfil the objectives of the FCD use-case.

4.2. Discussion

Modern vehicles produce several hundred megabyte of sensor data per second: For instance, location data, changes with 1-4 Hz, but frequency can go as high as 20 Hz in the case of speed, due to device capabilities, system design, or statutory provisions. Considering the presented use-cases, often only few, but maybe particular, data samples are needed. Accordingly, in theory, only few values must be transferred between D and N, what even enables some use-cases with respect to wireless cellular networks constraints and thousands of vehicles. But, in its current release, ETSI has been decided that the information, passed between M2M applications (e.g. DAs and NAs), is a black box – opaque – to the M2M platform (ETSI, 2013b). Currently, the *contentInstance* resource shall contain the attributes *contentSize*, *creationTime*, *lastModifiedTime*, and the *content* itself. The latter additionally can be described by the attribute *contentType*, according to MIME-Type definitions (Freed and Borenstein, 1996a; Freed and Borenstein, 1996b), where appropriate. But these attributes are only meta data and do not change that the actual content is not transparent to the M2M platform. Accordingly the *filterCriteria* resource, which indicates that e.g. the *SUBSCRIPE/NOTIFY* can be filtered in more detail, can only address meta data of the *contentInstance*. For example time-related criteria like *ifModifiedSince*, *ifUnmodifiedSince*, *createdAfter*, and *createdBefore* or content-size-related attributes like *sizeFrom*, *sizeUntil* can be used. Besides descriptive attributes like *searchString* are available (ETSI, 2013c). On one hand this seems to be a reasonable, central design decision to build a horizontal service platform and prevent building another vertical silo solution, and it might be sufficient and tolerable if e.g. only a very limited temperature sensor, providing one sample per minute, is connected. But on the other hand, this leaves tedious tasks ahead if a very complex device like a vehicle should be integrated, which offers a vast amount of complex data that might chance several times per second. Advantageous automotive data *subscriptions*/*filterCriteria* like

- "Provide speed, current position, and timestamp of the car every 100 m"
- "Send notification if water temperature is above 90 °C"
- "Provide speed, position, outside-temperature, rain sensor value, timestamp in case of an ABS or ESP control intervention"

are not feasible according to the current M2M release. Hence, to realise specific use-cases, vehicle data filtering can/must already be performed within the M2M

application layer. Thus, as a workaround, data might be provided not common, but specific, e.g. within an appropriately filled data container *FloatingCarData*, on which the existing meta data *filterCriteria* are sufficient. Since it can be expected that the data acquisition rules are not fixed and may change often during runtime, depending on running applications, current driving region, traffic status, wireless access network utilisation, etc., extensions for dynamic configuration might be valueable. However, this again causes vertically integrated and isolated silo solutions on top of the common M2M Service Architecture.

M2M Service Architecture specifies values like *delayTolerance*, and *minimalTimeBetweenNotifications*, which indicate that timing-constraints can be defined for notification and data transmission. This should facilitate some network optimisations with respect to the number of packets and the reduction of overhead in case of aggregation of several packets (Lo et al., 2013). In the context of complex M2M-devices, like vehicles, which potentially offer a vast amount of information, semantics support – data-awareness – for *SUBSCRIPTION/NOTIFY* below the M2M application layer, promises an important bandwidth-reduction, while retaining most flexibility. This could also facilitate data-mediation functions on D and N, which prevent the multiple transmission of the same information, caused by independent, disjoint applications.

5. Conclusions

Connectivity and appropriate distributed automotive software platforms are the foundation for future vehicles to facilitate an additional level of inference, prediction, and responsiveness, to be perceived as *intelligence*. Such intelligent vehicles will offer enhanced user experience and evolve the activities associated with driving towards an Intelligent Transportation System, with increased traffic safety and efficiency. To enable this vision, this paper proposes an M2M-based Automotive Service Delivery Platform, facilitating many automotive applications implemented headunit-external on OEM servers. The ETSI M2M Service Architecture has been selected, because it specifies a horizontal service platform, with common Service Capabilities, interfaces, and resource-based, RESTful, communication on Resource Trees and it has been designed to offer an end-to-end integration solution for many different domains, including automotive. We presented experiences, gained during prototypical implementation of core elements according to the current M2M release, and evaluated the capabilities on the basis of representative use-cases, in particular Extended Floating Car Data. We noticed that by now, the M2M layer only helps to unify communication with and management of devices, to achieve decoupling between applications and devices, and to handle heterogeneous access network technologies. But, as discussed, ETSI M2M Service Architecture is currently not very efficient for complex devices, offering many information at a high rate, like vehicles. In order to support this, enhancements that make data understandable (transparent) to the M2M platform are necessary and will be subject of further research.

6. References

3GPP (2013), 3rd Generation Partnership Project (3GPP), http://www.3gpp.org/. (Accessed 10 April 2013)

Bauer, S. (2010), "Das vernetzte Fahrzeug – Herausforderungen für die IT", *Informatik-Spektrum*, Vol. 34, No. 1, pp. 38–41.

Bormann, C., Castellani, A.P. and Shelby, Z. (2012), "CoAP: An Application Protocol for Billions of Tiny Internet Nodes", *Internet Computing*, Vol. 16, No. 2, pp. 62–67.

Boswarthick, D., Elloumi, O. and Hersent, O. (2012), "M2M Communications: A Systems Approach", John Wiley & Sons, ISBN: 978-1-119-99475-6

Broy, M. (2006), "Challenges in Automotive Software Engineering", *Proceedings of the 28th International Conference on Software Engineering (ICSE)*, Shanghai, China, pp. 33–42.

Broy, M., Krüger, I.H., Pretschner A. and Salzmann C. (2007), "Engineering Automotive Software", *Proceedings of the IEEE*, Vol. 95, No. 2, pp. 356–373.

ETSI (2013a), "ETSI - M2M", http://www.etsi.org/index.php/technologies-clusters/technologies/m2m. (Accessed 10 April 2013)

ETSI (2013b), "Machine-to-Machine communications (M2M); Functional Architecture", *European Telecommunications Standards Institute*, TS 102 690, V.2.1.1.

ETSI (2013c), "Machine-to-Machine communications (M2M); mIa, dIa and mId interfaces", *European Telecommunications Standards Institute*, TS 102 921, V.2.1.1.

Fielding, R.T. (2000), "Architectural Styles and the Design of Network-based Software Architectures", *Doctoral dissertation*, University of California.

Freed, N. and Borenstein, N. (1996a), "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", IETF.

Freed, N. and Borenstein, N. (1996b), "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", IETF.

Glaab, M., Fuhrmann W., Wietzke J. and Ghita B.V. (2010), "A New Architectural-Approach for Next Generation Automotive Applications", *Proceedings of the Sixth Collaborative Research Symposium on Security, E-Learning, Internet and Networking (SEIN2010)*, Plymouth, United Kingdom, pp. 11–18.

Glaab, M., Fuhrmann, W. and Wietzke, J. (2011), "Entscheidungskriterien für die Verteilung zukünftiger automotiver Anwendungen im Kontext vernetzter Fahrzeuge", *Mobilkommunikation 2011 - Technologien und Anwendungen - 16. ITG-Fachtagung*, pp. 149-154.

Lo, A., Law, Y. and Jacobsson, M. (2013), "A cellular-centric service architecture for machine-to-machine (M2M) communications", *Wireless Communications, IEEE*, Vol. 20, No. 5, pp. 143–151.

OMA (2013), "Open Mobile Alliance", http://openmobilealliance.org/. (Accessed 11 April 2013.

oneM2M (2013), "oneM2M", http://onem2m.org. (Accessed 16 April 2013)

Pautasso, C., Zimmermann, O. and Leymann, F. (2008), "Restful web services vs. 'big' web services: making the right architectural decision", *Proceeding of the 17th international conference on World Wide Web (WWW2008)*, ACM, pp. 805-814.

Pretschner, A., Broy, M., Kruger, I.H. and Stauner, T. (2007), "Software engineering for automotive systems: A roadmap", *Future of Software Engineering (FOSE'07)*, pp. 55–71.

Shelby, Z., Hartke, K., and Bormann, C. (2013), "Constrained Application Protocol (CoAP)", *CoRE Working Group*, IETF.

Shimizu, N. (2004), "Analysis of Automotive Telematics Industry in Japan", *Doctoral dissertation*, Massachusetts Institute of Technology.

Wu, G., Talwar, S., Johnsson, K., Himayat, N. and Johnson, K.D. (2011), "M2M: From Mobile to Embedded Internet", *Communications Magazine, IEEE*, Vol. 49, No. 4, pp. 36–43.