University of Plymouth

PEARL **https://pearl.plymouth.ac.uk**

04 University of Plymouth Research Theses 01 Research Theses Main Collection

2018

A Distributed Service Delivery Platform for Automotive Environments: Enhancing Communication Capabilities of an M2M Service Platform for Automotive Application

Glaab, Markus

http://hdl.handle.net/10026.1/11249

http://dx.doi.org/10.24382/623
University of Plymouth

All content in PEARL is protected by copyright law. Author manuscripts are made available in accordance with publisher policies. Please cite only the published version using the details provided on the item record or document. In the absence of an open licence (e.g. Creative Commons), permissions for further reuse of content should be sought from the publisher or author.

University of Plymouth

A Distributed Service Delivery Platform for Automotive Environments: Enhancing Communication Capabilities of an M2M Service Platform for Automotive Application

by

Markus Glaab

A thesis submitted to the University of Plymouth in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Computing, Electronics and Mathematics

March 2018

Copyright Statement

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the author's prior consent.

Abstract

A Distributed Service Delivery Platform for Automotive Environments: Enhancing Communication Capabilities of an M2M Service Platform for Automotive Application

Markus Glaab

The automotive domain is changing. On the way to more convenient, safe, and efficient vehicles, the role of electronic controllers and particularly software has increased significantly for many years, and vehicles have become software-intensive systems. Furthermore, vehicles are connected to the Internet to enable Advanced Driver Assistance Systems and enhanced In-Vehicle Infotainment functionalities. This widens the automotive software and system landscape beyond the physical vehicle boundaries to presently include as well external backend servers in the cloud. Moreover, the connectivity facilitates new kinds of distributed functionalities, making the vehicle a part of an Intelligent Transportation System (ITS) and thus an important example for a future Internet of Things (IoT).

Manufacturers, however, are confronted with the challenging task of integrating these ever-increasing range of functionalities with heterogeneous or even contradictory requirements into a homogenous overall system. This requires new software platforms and architectural approaches. In this regard, the connectivity to fixed side backend systems not only introduces additional challenges, but also enables new approaches for addressing them.

The vehicle-to-backend approaches currently emerging are dominated by proprietary solutions, which is in clear contradiction to the requirements of ITS scenarios which call for interoperability within the broad scope of vehicles and manufacturers. Therefore, this research aims at the development and propagation of a new concept of a universal distributed Automotive Service Delivery Platform (ASDP), as enabler for future automotive functionalities, not limited to ITS applications. Since Machine-to-Machine communication (M2M) is considered as a primary building block for the IoT, emergent standards such as the oneM2M service platform are selected as the initial architectural hypothesis for the realisation of an ASDP. Accordingly, this project describes a oneM2M-based ASDP as a reference configuration of the oneM2M service platform for automotive environments.

In the research, the general applicability of the oneM2M service platform for the proposed ASDP is shown. However, the research also identifies shortcomings of the current oneM2M platform with respect to the capabilities needed for efficient communication and data exchange policies. It is pointed out that, for example, distributed traffic efficiency or vehicle maintenance functionalities are not efficiently treated by the standard. This may also have negative privacy impacts. Following this analysis, this research proposes novel enhancements to the oneM2M service platform, such as application-data-dependent criteria for data exchange and policy aggregation. The feasibility and advancements of the newly proposed approach are evaluated by means of proof-of-concept implementation and experiments with selected automotive scenarios. The results show the benefits of the proposed enhancements for a oneM2M-based ASDP, without neglecting to indicate their advantages for other domains of the oneM2M landscape where they could be applied as well.

Contents

List of Figures ..... **XII**

List of Tables ..... **XVI**

Author's Declaration ..... **XVII**

Acknowledgement ..... **XXI**

1 Introduction and Overview **1**

1.1 Focus of this Research 1
1.2 Aims and Objectives 2
1.3 Method Overview 3
1.3.1 High-Level Methodology 3
1.3.2 Low-Level Methodology 4
1.4 Contributions 5
1.5 Published Papers 6
1.6 Thesis Outline 7

2 Review of the Automotive Environment **9**

2.1 Functional Domains 9
2.1.1 In-Vehicle Infotainment 10
2.1.2 Advanced Driver Assistance Systems towards Highly Automated Driving 12
2.1.3 Intelligent Transportation Systems 14
2.2 Automotive Software and System Landscape 16
2.2.1 Computing Components 17
2.2.2 Communication Technologies 19
2.3 Domain Characteristics and Challenges 21
2.3.1 Programmable Vehicle 22
2.3.2 Distributed Functionalities 23
2.3.3 Reuse of Functionalities 24
2.3.4 Life Cycle and Innovation Cycle 25
2.3.5 Variants and Configurations 26
2.3.6 Heterogeneity and Mixed Criticality 27
2.3.7 Cost Model 27
2.3.8 Development Process 28
2.3.9 Conclusion 29
2.4 Software (and System) Architecture Design and Methodology 29
2.4.1 Software and System Architecture 30
2.4.2 Problem Space 31
2.4.3 Solution Space 31
2.4.4 Architecture Design 32

2.4.5 Platform-Based Design ... 33
2.4.6 Architecture Analysis ... 36
2.4.7 Conclusion ... 36
2.5 Related Work ... 37
2.5.1 Basic Approaches to Improve Integration of Heterogeneous Functionalities ... 37
2.5.2 AUTOSAR ... 38
2.5.3 GENIVI ... 41
2.5.4 OEM Vehicle-to-Backend Platforms ... 42
2.6 Problem Statement ... 42
2.7 Summary ... 44

3 The Distributed Automotive Service Delivery Platform Concept ... 46

3.1 Concept ... 46
3.1.1 Principles ... 47
3.1.2 Enabling Architecture: A Distributed Automotive Service Delivery Platform ... 48
3.2 Scenarios ... 49
3.2.1 Extended Floating Car Data ... 49
3.2.2 Vehicle Maintenance / Fleet Management ... 51
3.2.3 Enhanced Navigation ... 51
3.3 Criteria for the Distribution of Automotive Functionalities ... 53
3.3.1 Application Type ... 55
3.3.2 Data flow ... 55
3.3.3 Performance ... 57
3.3.4 Application Lifecycle and Data Update Frequency ... 58
3.3.5 Quality of Service ... 59
3.3.6 Assessment of Selected Functionalities ... 60
3.3.7 Discussion ... 63
3.4 Viewpoints and Derived Requirements ... 64
3.4.1 Future Vehicle-to-Backend Platforms ... 64
3.4.2 The Vehicle as Part of an Internet of Things ... 65
3.4.3 Towards an (Automotive) Embedded Internet ... 67
3.5 Summary ... 68

4 An Automotive Service Delivery Platform Based on the oneM2M Service Platform ... 70

4.1 Introduction to M2M, oneM2M ... 71
4.1.1 M2M Communication ... 71
4.1.2 oneM2M Standard ... 72
4.1.3 Related Work ... 73
4.2 M2M as Initial Hypothesis ... 74
4.2.1 M2M aims Interoperability ... 75

4.2.2 M2M Aims Network Efficiency ... 77
4.2.3 M2M Considers the Automotive Domain ... 78
4.3 Functional Architecture ... 78
4.3.1 Layers and Entities ... 79
4.3.2 Reference Points ... 80
4.3.3 Common Services Functions ... 81
4.3.4 Domains ... 86
4.3.5 Nodes ... 86
4.3.6 Configurations ... 87
4.4 Service/Resource Model and Technologies ... 88
4.4.1 Service-Orientation, Service-Oriented Architecture ... 88
4.4.2 REST, RESTful Architecture ... 89
4.4.3 Resources and Resource Structure ... 90
4.4.4 Methods ... 93
4.4.5 Protocol Stack and Bindings ... 94
4.5 Communication and Data Exchange Mechanisms ... 95
4.5.1 Principles ... 96
4.5.2 Request/Response ... 97
4.5.3 Announcement ... 98
4.5.4 Subscribe/Notify ... 98
4.6 The oneM2M-based Automotive Service Delivery Platform ... 101
4.6.1 Reference Configuration ... 102
4.6.2 Basic Communication Scenarios ... 103
4.7 Summary ... 110

5 Analysis of Current Data Exchange Capabilities of the oneM2M Service Platform ... 112

5.1 Condensed ASDP Scenario ... 113
5.1.1 Node Configuration ... 113
5.1.2 Application Entities ... 114
5.2 Principles ... 116
5.2.1 Filter Requirements vs. Capabilities ... 116
5.2.2 Filter Positions ... 118
5.3 Detailed Analysis of oneM2M Data Exchange Capabilities ... 120
5.3.1 Application Data Handling ... 120
5.3.2 Existing Filtering Capabilities for Exchange of Application Data by Use of the Subscribe/Notify Mechanism ... 123
5.3.3 Analysis with ASDP scenario ... 135
5.4 Shortcomings of the Current oneM2M Data Exchange and Their Impacts ... 137
5.4.1 No Application-Data-Dependent Criteria for Data Exchange ... 138
5.4.2 No Aggregation of Subscriptions ... 139

5.5 Summary ... 140

6 Proposal of Novel Data Exchange Capabilities for the oneM2M Service Platform ... 141

6.1 Proposed Enhancements and Derived Requirements ... 141
6.1.1 Enhancement 1: Application-Data-Dependent Notification Criteria for Data Exchange with Subscribe/Notify Mechanism ... 141
6.1.2 Enhancement 2: Aggregation of Subscriptions ... 142
6.2 Approach ... 143
6.2.1 XML and XSD to Enable Transparent Application Data ... 143
6.2.2 Complex Event Processing to Enable Application-Data-Dependent Notification Criteria for Data Exchange With Subscribe/Notify Mechanism ... 144
6.2.3 Aggregation of Subscriptions Enabled Through Criteria Unification ... 147
6.3 Alternative Architectural Approaches ... 148
6.3.1 Approach 1: Within oneM2M AE Layer ... 150
6.3.2 Approach 2: Within oneM2M CSE layer ... 153
6.3.3 Excursion: Towards (Full) Semantic Interoperability in oneM2M ... 155
6.3.4 Assessment ... 158
6.4 Summary ... 161

7 Prototype Implementation ... 163

7.1 Technologies ... 163
7.1.1 Eclipse OM2M Project ... 164
7.1.2 Esper CEP Engine ... 165
7.2 Building Blocks ... 166
7.2.1 Content Decoder ... 167
7.2.2 Esper Event Adaptor ... 167
7.2.3 Subscription and Notification Criteria Aggregator ... 168
7.2.4 Esper Statement Adaptor ... 169
7.3 Experiments and Estimations ... 170
7.3.1 Setup and Test Strategy ... 170
7.3.2 Experiment 1: Remaining fuelRange with Single-Steps Provider, Variant 1 ... 190
7.3.3 Experiment 2: Remaining fuelRange with Decimal-Steps Provider ... 192
7.3.4 Experiment 3: Remaining fuelRange with Single-Steps Provider, Variant 2 ... 194
7.4 Concluding Considerations ... 196
7.5 Summary ... 198

8 Conclusions and Future Work ... 199

8.1 Achievements of the Research ... 199
8.2 Limitations ... 201
8.3 Suggestions for Future Work ... 202

Bibliography **204**

Abbreviations **226**

List of Figures

Figure 1.1: Concept and architecture development as cyclic process (adopted from Masak, 2009) 3

Figure 2.1: Functional Domains 10

Figure 2.2 Intelligent Transportation System scenarios (Source: ETSI TR 102 638, 2009) 14

Figure 2.3: Automotive software and system landscape 17

Figure 2.4: The platform-based design approach (Natale & Sangiovanni-Vincentelli, 2010) 35

Figure 2.5: The AUTOSAR software architecture (AUTOSAR, 2013) 39

Figure 3.1: Basic architecture of the distributed automotive software platform 47

Figure 3.2: Extended Floating Car Data Scenario 50

Figure 3.3: Vehicle Maintenance / Fleet Management Scenario 51

Figure 3.4: Enhanced Navigation Scenario 52

Figure 3.5: Abstract model for distributed functionalities 54

Figure 3.6: Qualitative computational and memory requirements assessment criteria 57

Figure 3.7: Qualitative application lifecycle and data update frequency assessment criteria 58

Figure 3.8: Qualitative tolerable delay and bandwidth requirements assessment criteria 59

Figure 3.9: Qualitative caching or resubmission possible/reasonable assessment criteria 60

Figure 4.1: Limited interoperability between components, vendors, and domains through heterogeneous abstraction levels, interfaces, and technologies 75

Figure 4.2: Improved interoperability between components, vendors, and domains through unified abstraction levels, interfaces, and technologies 80

Figure 4.3: Common Service Functions in the Common Services Entity (Source: oneM2M TS-0001, 2015) 81

Figure 4.4: Configurations supported by the oneM2M service platform (Source: oneM2M TS-0001, 2015, p. 19) 87

Figure 4.5: Resource tree structure example of a CSEBase 93

Figure 4.6: Current oneM2M protocol stack for Mca, Mcc, Mcc' reference points 95

Figure 4.7: Generic oneM2M configuration for communication and data exchange considerations 97

Figure 4.8: Sequence Diagram of Subscribe/Notify example 99

Figure 4.9: Functional architecture of the reference configuration of a oneM2M-based Automotive Service Delivery Platform 103

Figure 4.10: Sequence diagram of subscription setup within transparent intermediary node scenario 104

Figure 4.11: Sequence diagram of notification within transparent intermediary node scenario 105

Figure 4.12: oneM2M Node with back-to-back functionality: Realisation of different views on the same node 107

Figure 4.13: Sequence diagram of subscription setup within back-to-back intermediary node scenario 108

Figure 4.14: Sequence diagram of notification within back-to-back intermediary node scenario 109

Figure 5.1: Condensed ASDP scenario 113

Figure 5.2: Data set provided by an AE and required subset by other AEs 116

Figure 5.3: Data set provided/required by AEs, filter selection and resulting loss or overhead 117

Figure 5.4: Data set provided/required by AEs, increased filter selection and resulting overhead 117

Figure 5.5: Dataflow with possible filter positions between a distributed AE scenario 118

Figure 5.6: Resource structure of container and contentInstance 121

Figure 5.7: Combined filter for notifications 124

Figure 5.8: Compound resource structure of container, contentInstance, subscription, eventNotificationCriteria, and schedule 125

Figure 5.9: Concatenation and interaction of three filter aspects of subscribe/notify mechanism 126

Figure 5.10: Compound resource structure of container, contentInstance, subscription, eventNotificationCriteria, and schedule including relations of the notification criteria 127

Figure 5.11: Compound resource structure of container, contentInstance, subscription, eventNotificationCriteria, and schedule including content-related notification criteria 130

Figure 5.12: Compound resource structure of container, contentInstance, subscription, eventNotificationCriteria, and schedule including time-related notification schedule criteria 132

Figure 5.13: Application data exchange example of ASDP scenario with current oneM2M data exchange capabilities 136

Figure 6.1: Enhanced data exchange capabilities as concatenation of new filter capabilities with existing filter capabilities 143

Figure 6.2: Flowchart for aggregation of subscriptions at local CSE 148

Figure 6.3: Flowchart for aggregation of subscription at transit CSE 148

Figure 6.4: Design space for Mca interface 149

Figure 6.5: Concatenation and interaction of new application-data-dependent filter aspect with existing three filter aspects of subscribe/notify mechanism for alternative architectural approach 1 150

Figure 6.6: Enhanced data exchange with alternative architectural approach 1 151

Figure 6.7: Concatenation and interaction of new application-data-dependent filter aspect with existing three filter aspects of subscribe/notify mechanism for alternative architectural approach 2 153

Figure 6.8: Enhanced data exchange for alternative architectural approach 2 154

Figure 7.1: Component diagram of enhanced OM2M prototype showing modified and new CORE plugins 164

Figure 7.2: Integration of enhanced data exchange capabilities for subscribe/notify within CSE layer of oneM2M service platform 166

Figure 7.3: Experiment setup 170

Figure 7.4: Usage of Simple REST Client as AEPrototype showing the creation/registration of a VehicleDataProvider application at the CSE/NSCL 172

Figure 7.5: Usage of OM2M Web Interface as Monitor showing the evaluation of the resource tree after the creation/registration of a VehicleDataProvider application at the CSE/NSCL 173

Figure 7.6: Sequence Diagram of Experiment Bootstrap 174

Figure 7.7: Sequence Diagram of Experiment Execution and Result Retrieve 180

Figure 7.8: Comparison of the number of notifications for VehicleDataSubStd and VehicleDataSubEnh during fuelRange decrease according to the conditions of Experiment 1 192

Figure 7.9: Comparison of the number of notifications for VehicleDataSubStd and VehicleDataSubEnh during fuelRange decrease according to the conditions of Experiment 2 194

Figure 7.10: Comparison of the number of notifications for VehicleDataSubStd and VehicleDataSubEnh during fuelRange decrease according to the conditions of Experiment 3 196

List of Tables

Table 2.1: ETSI basic set of applications related to traffic safety (ETSI TR 102 638, 2009, p. 18) 15

Table 2.2: ETSI basic set of applications related to traffic efficiency (ETSI TR 102 638, 2009, p. 18) 15

Table 2.3: ETSI basic set of applications related to traffic efficiency (ETSI TR 102 638, 2009, p. 18) 16

Table 2.4: Selected vehicle-internal communication technologies (Sagstetter, 2016; Talbot & Ren, 2009; W. Zimmermann & Schmidgall, 2014) 19

Table 3.1: Possible data flows between vehicle and backend and their degree of limitation regarding increased requirements 56

Table 3.2: Assessment of criteria for the distribution of selected automotive applications 62

Table 4.1: Selection of oneM2M resource types (oneM2M TS-0001, 2015) 92

Table 4.2: Supported operations between originator and receiver entity 94

Table 6.1: Examples of beneficial automotive application-data-dependent notification criteria and their EPL statement representation. 146

Table 7.1: Mapping of selected oneM2M terms to ETSI M2M terms 165

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee.

Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.

Word count of main body of thesis: 62298

Date 04.03.2018

Signature Markus Glaab

To my family

Acknowledgement

This research project would not have been possible without the support and contribution of many people, and the organisational facility, namely the University of Applied Sciences, Darmstadt, DE, in cooperation with the Centre for Security, Communication and Network Research, Plymouth University, UK.

I want to especially thank my Director of Studies, Prof Dr Woldemar Fuhrmann. Thank you for many intensive and fruitful discussions, professional advice, personal support and guidance throughout this project. Secondly, I want to thank my supervisor Prof Dr Joachim Wietzke for his constant and outstanding support and advice. In particular, I want to express many thanks for providing the foundation and environment for the development of the central contributions of this research by means of heading the ICM labs at the University of Applied Sciences, Darmstadt, DE. Moreover, I would like to thank my supervisor Dr Bogdan V. Ghita for his invaluable help and guidance for this research project in general and contribution to publications in particular.

Many thanks also go to my fellow researchers from the former ICM research group, namely Mrs Bettina Kurz-Kalweit, Mr Sergio Vergata, Mr Clemens Fischer, Mr Tobias Holstein, and most notably Dr Andreas Knirsch and Mr Pierre Schnarz: I thank you for your friendship, interesting and helpful discussions, and incredible support in particular during the final stages of this research project.

Furthermore, I would like to thank Mr Daniel Ebanja, Dr Benjamin Heckmann, Mr Björn Bär, Mr Stefan Benkner, and Dr Jakob Schelbert for their valuable inputs to this research.

Many thanks to Carole Watson and the team of the graduate school administration for your help and support regarding administrative issues.

Finally, I would like to thank my lovely wife Katharina for her patience, encouragement and support throughout the last years. Additionally, I would like to thank my sister Sabrina, my parents, parents-in-law, and friends for their help and support.

1 Introduction and Overview

During the last decades, the relevance of electronics and software embedded in cars has increased exponentially. Starting in the late 1960s and the beginning of 1970s, the first kilobytes of software were introduced, hidden from the passengers' perception, to control the ignition of the engine (Venkatesh Prasad, Broy, & Krueger, 2010). Nowadays, premium cars have more than 100 million lines of code distributed over up to 100 Electronic Control Units (ECUs). At the same time, the role of software has changed. While at the beginning software only supports hardware-enabled features, it now "[increasingly replaces] functions previously performed by hardware, thereby replacing hardware [or it] explicitly augments new hardware-enabled features" (Venkatesh Prasad et al., 2010). Moreover, today a significant number of functionalities is realised purely with software and this will continue to increase. Consequently, software plays a major role for the latest achievements in the area of safety, comfort, performance, and ecology of the vehicles (Charette, 2009). Hardung et al. stated that today "electronics make 90% of the innovations [of a vehicle], 80 % out of that in the area of software" (Hardung, Kölzow, & Krüger, 2004). Sophisticated graphical displays and touchscreens replaced hardware buttons for vehicle control and moved software in the focus of drivers' vehicle perception. Hence, "software enables [Original Equipment Manufacturers (OEMs)] and suppliers to tailor systems to particular customers' needs. In other words, software can help differentiate between cars." (Pretschner, Broy, Kruger, & Stauner, 2007). In this regard, software nowadays is also a distinctive competitive advantage.

Cars recently became connected with the Internet by use of wireless cellular networks. This extends the automotive software and system landscape beyond the vehicle internals now including also external entities, such as servers in the backend. The connected vehicle is the enabler for a huge amount of novel and beneficial functionalities, e.g., within the areas of In-Vehicle Infotainment and Advanced Driver Assistance Systems. Moreover, it is the foundation for the vision of vehicles becoming an integral part of an Intelligent Transportation System, which provides increased traffic safety, traffic efficiency, and comfort to its users.

1.1 Focus of this Research

The opportunities of connected cars are huge with regard to the increase of comfort, safety, and efficiency. However, they also cause and require a massive change in the automotive domain

and used technologies to materialise (Broy, 2006a). Automotive manufacturers and engineers require now in addition to mechanical and electrical/electronic expertise also profound knowledge in computer science, distributed software architectures, and recently in addition to vehicle-internal network technologies also knowledge in wireless cellular network technologies and server infrastructure. The existing and historically grown software and system architectures are not good enough to enable the realisation of the future automotive functionalities (Broy, 2006a). Enhanced architectures and platforms are required, which are developed interdisciplinary to address the challenges of connected vehicles.

Due to this, the focus of this research are enhanced vehicle-to-backend platforms, that enable the realisation of the functionalities associated with future connected vehicles. This includes the analysis of these distributed functionalities to identify requirements to the vehicle-to-backend platforms. Furthermore, it requires the consideration of suitable architectures, building blocks, and technologies for the platform approach.

1.2 Aims and Objectives

The aim of this research is to propose an enhanced vehicle-to-backend platform architecture, which facilitates the realisation of functionalities related to connected vehicles, as for example those in the context of Intelligent Transportation Systems, Advanced Driver Assistance Systems and In-Vehicle Infotainment Systems. Thereby, this research envisages a holistic view on the software and system architectures of vehicle-to-backend platforms, reflecting the wide range of involved domains, such as automotive, telecommunication, and Internet, as well as disciplines, such as system architects, software architects, or data engineers.

In more detail, this leads to the following objectives:

1. To perform a comprehensive review and analysis of the current automotive environment by means of driving functional domains, software and system landscape as well as domain characteristics and challenges.
2. To develop a concept for a novel vehicle-to-backend service platform enabling the delivery of future distributed automotive software functionalities.
3. To investigate the appropriateness of an M2M service platform (oneM2M TS-0001, 2015; oneM2M TS-0004, 2015) as enabler for the concept previously developed by means of architectural analysis and selected automotive scenarios.
4. To propose enhancements and an architectural approach that improve the suitability of an M2M service platform for automotive application, in case that the investigation has identified shortcomings of the M2M service platform.
5. To proof the feasibility of proposed enhancements for the M2M service platform through a prototype implementation.

1.3 Method Overview

The methodologies applied to this research reflect its focus on concepts and architectures. They can be divided into a high-level methodology that describes the overall structure of this research and different low-level methodologies that are used to develop single research artefacts or contributions.

1.3.1 High-Level Methodology

The high-level methodology of this research reflects the iterative or cyclic nature of concept and architecture development, which means that the transition from the problem space to the solution space is traversed several times at different abstraction levels (see Figure 1.1) (Masak, 2009).

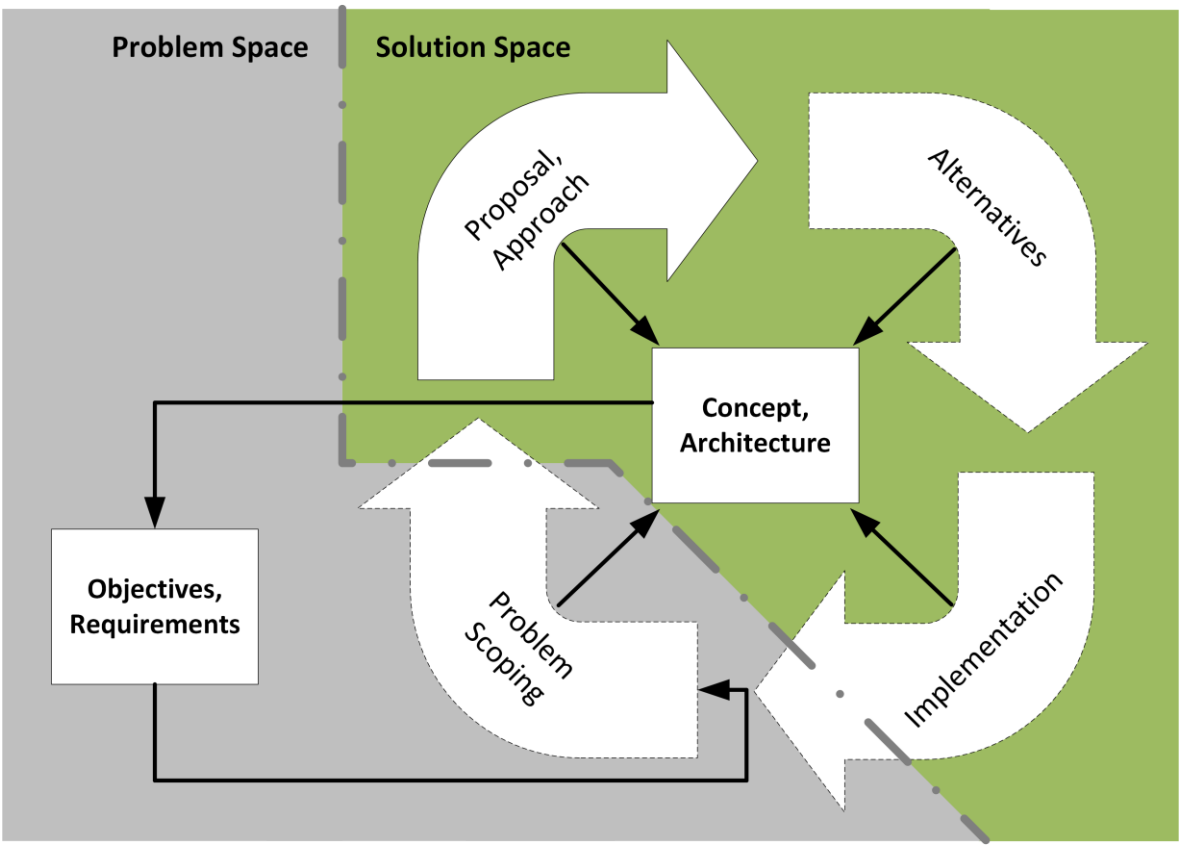

Figure 1.1: Concept and architecture development as cyclic process (adopted from Masak, 2009)

During this process, the problem space is built from objectives and requirements. This "problem" in general might become scoped, followed by a proposal or approach as to how this problem could be solved. In many cases, several solutions are possible which is why the evaluation and selection of alternatives might be necessary which in turn finally lead to the implementation. All the named activities influence the concept or architecture "under development". It has to be considered that conceptual decisions, as well as architectural design decisions overlap with the problem space and the solution space because each decision widens

the problem space and narrows the solution space (Masak, 2009). This is because each decision typically leads to additional requirements and less design space for the solution (Masak, 2009). New requirements, and the fact that this cyclic process is traversed several times on different abstraction levels, is the reason why it might become necessary to investigate related concepts or work at different stages of the research and this thesis, simply because new requirements have arisen, possibly at a different abstraction levels. However, all activities may not be present during all cycles. Particularly the scoping and the investigation of alternatives might be of minor significance at different abstraction levels or at different maturity levels of the solution (i.e., concept or architecture).

This methodology is combined with the platform-based design approach (Sangiovanni-Vincentelli & Martin, 2001) which is introduced in more detail in Section 2.4.5. At this point, it shall only be noted that the cyclic process is not only performed top-down, but also includes bottom-up considerations, such as a platform candidate that is investigated as an architectural solution.

In more detail, the following cycles of this process are performed within this research:

- Based on the review of the automotive environment that includes characteristics and challenges, fundamental objectives are derived. These are scoped to a certain subset of the overall automotive software and system landscape, namely vehicle-to-backend platforms. Based on this, the problem space for the concept "under development" is further detailed which finally leads to the concept of a distributed Automotive Service Delivery Platform as solution to the selected scope of the automotive challenges.
- This concept again impacts the problem space where further objectives or requirements can be derived to implement it.
- Then, according to the Platform-Based Design approach (see Section 2.4.5), the oneM2M service platform is introduced as candidate architectural solution to the developed concept. It is comprehensively analysed which leads to the identification of shortcomings.
- The identified shortcomings are the starting point for a proposal of architectural enhancements for which requirements are derived. In advance to the prototype implementation, the general approach is described, and architectural alternatives are discussed.

1.3.2 Low-Level Methodology

The low-level methodologies of this research refer to the development of single research artefacts or contributions, such as concepts, proposals, and approaches. Thereby the main research methodology applied is conceptual analysis as a basis concept formulation for software and system architectures (cf. Glass, Ramesh, & Vessey, 2004; Ramesh, Glass, & Vessey, 2004). This is primarily performed on the level of abstract concepts, computing elements, or

algorithm/protocols. Besides, proposed enhancements are empirically evaluated by use of concept implementation (proof-of-concept) (Ramesh et al., 2004; Segal, 2003). The selected methodologies match with those predominantly applied in the related disciplines (i.e. computer science and software engineering) (Dodig-Crnkovic, 2002; Glass et al., 2004; Ramesh et al., 2004)

1.4 Contributions

Concerning the focus of this research (see Section 1.1) and related aims and objectives (see Section 1.2), contributions are made. These are the main contributions of this research:

- The detailed analysis of characteristics and challenges of the automotive environment with regard to the advancements in in the area of In-Vehicle Infotainment, Advanced Driver Assistance Systems, and Intelligent Transportation Systems.
- The revelation of the gap of a common vehicle-to-backend platform as basis for the realisation of the functionalities and expectations, associated with the connected vehicle.
- The development and proposal of the distributed Automotive Service Delivery Platform concept as a novel approach for vehicle-to-backend platforms
- Criteria for the qualitative assessment of automotive applications that enable analysis of suitable decompositions and distributions of functionalities within a distributed Automotive Service Delivery Platform.
- Identification of key architectural design decisions of the oneM2M service platform that proved its general suitability as enabler for the distributed Automotive Service Delivery Platform concept.
- The description of a reference architecture for a oneM2M-based distributed Automotive Service Delivery Platform.
- The identification of shortcomings with regard to the data exchange capabilities of the oneM2M service platform which may decrease the network efficiency of the distributed functionalities.
- The development of novel enhanced data exchange capabilities for the oneM2M service platform that increase the network efficiency and thus improve the suitability of the oneM2M service platform as enabler for the distributed Automotive Service Delivery Platform concept.
- The proof the applicability of the proposed enhanced data exchange capabilities through prototype implementation together with experiments and estimations that show the increased network efficiency based on a typical automotive scenario.

1.5 Published Papers

At the time of writing, seven papers have been published that cover significant parts of this research:

1. Markus Glaab, Woldemar Fuhrmann, Joachim Wietzke, and Bogdan V Ghita. A New Architectural-Approach for Next Generation Automotive Applications. In *Proceedings of the Sixth Collaborative Research Symposium on Security, E-Learning, Internet and Networking (SEIN2010)*, pages 11–18, Plymouth, United Kingdom, November 2010. ISBN 978-1-84102-269-7

2. Markus Glaab, Woldemar Fuhrmann, and Joachim Wietzke. Entscheidungskriterien für die Verteilung zukünftiger automotiver Anwendungen im Kontext vernetzter Fahrzeuge. In *Mobilkommunikation 2011 - Technologien und Anwendungen - 16. ITG-Fachtagung*, Osnabrück, Germany, 2011.

3. Markus Glaab, Woldemar Fuhrmann, and Joachim Wietzke. The need for Transparent Data within M2M Service Capability Layer. Presented at the 9th KuVS NGSDP Expert Talk - 9. KuVS Fachgespräch on Next Generation Service Delivery Platforms: "Machine to Machine Communications Platforms, Applications and Standards", Berlin, Germany, April 2014.

4. Markus Glaab, Woldemar Fuhrmann, and Joachim Wietzke. Transparent Data for the M2M Service Capability Layer: Benefits and Approaches. In *21st International Conference on Telecommunications: Cooperation for a United World (ICT2014)*, Lisbon, Portugal, May 2014. IEEE Communications Society. http://doi.org/10.1109/ict.2014.7575094

5. Markus Glaab, Woldemar Fuhrmann, Joachim Wietzke, and Bogdan Ghita. A M2M-based Automotive Service Delivery Platform for Distributed Vehicular Applications. In *Tenth International Network Conference (INC2014)*, pages 35–45, Plymouth, United Kingdom, July 2014. ISBN 978-1-84102-373-1

6. Markus Glaab, Woldemar Fuhrmann, Joachim Wietzke, and Bogdan Ghita. Enhanced Data Exchange Capabilities for M2M Applications. In *Sixth International Conference on Internet Technologies & Applications 2015 (ITA 15)*, pages 160–164, Wrexham, North Wales, United Kingdom, September 2015. https://doi.org/10.1109/itecha.2015.7317388

7. Markus Glaab, Woldemar Fuhrmann, Joachim Wietzke, and Bogdan Ghita. Toward Enhanced Data Exchange Capabilities for the oneM2M Service Platform. *IEEE Communications Magazine*, 53(12), 2015. http://doi.org/10.1109/mcom.2015.7355583

1.6 Thesis Outline

Chapter 2 starts with the presentation of the automotive environment which is the foundation of this work. It provides a comprehensive review of the state of the art of automotive software and systems landscape. Therefore, first the functional domains In-Vehicle Infotainment, Advanced Driver Assistance Systems, and Intelligent Transportation Systems are described that drive the developments. All those functionalities must be integrated to a homogeneous overall system whose physical components and the communication technologies are afterwards described. Followed by the domain characteristics and challenges, it describes the automotive landscape from the technical and organisational perspective. Then, software and system design and methodology as foundation for the subsequent chapters is introduced. It includes the introduction of terminology as well as discussion of problem and solution space, the role of architectures, architecture design and analysis, and the Platform-Based Design methodology. Finally, related work can be crystallised on and located within the automotive landscape unveiling the gap, namely the absence of universal vehicle-to-backend platforms, which is where this research aims to deliver its contributions.

The concept of a universal and standardised distributed Automotive Service Delivery Platform and its principles as a solution to the identified gap is introduced in Chapter 3. The consideration of three related scenarios, and criteria for the distribution of functionalities between the vehicle and the backend contribute to three different viewpoints on the proposed concept. These viewpoints are the basis for the derivation of requirements to detail the problem space.

Chapter 4 begins with a general introduction to Machine-to-Machine Communication. Its similarities with the previously described concept with respect to problem space (i.e. challenges and requirements) motivate the consideration of the oneM2M service platform as initial architectural hypothesis. Afterwards, the oneM2M service platform is introduced in more detail and its fundamental design decisions are unveiled and analysed. This enables architecture analysis and reasoning to which degree the oneM2M Service Platform can fulfil the identified requirements of the distributed Automotive Service Delivery Platform concept. Although aspects have been identified that need more detailed consideration, such as the communication mechanisms and data exchange capabilities, the oneM2M service platform approved its general appropriateness as enabler for the distributed Automotive Service Delivery Platform. In this regard, this concept is finally mapped to the oneM2M service platform referred to as the reference configuration for a oneM2M-based Automotive Service Delivery Platform.

Since the data exchange capabilities are of central importance for the efficient applicability of the oneM2M service platform as enabler for the distributed Automotive Service Delivery Platform concept, Chapter 5 provides the detailed analysis. For this reason, first a condensed Automotive Service Delivery Platform scenario is introduced. Subsequently, the detailed analysis of existing data exchange capabilities of the subscribe/notify mechanism according to the current version of the oneM2M standard is performed, starting with the discussion of data

exchange principles. This analysis identifies shortcomings that typically result in decreased network efficiency and privacy of distributed functionalities.

To address the identified shortcomings, Chapter 6 proposes novel enhancements for the data exchange capabilities of the subscribe/notify mechanism of oneM2M. It starts with the derivation of requirements from the proposed enhancements. Afterwards, approaches to realise the enhancements are introduced. Finally, two possible architectural alternatives for the implementation of the enhancements within the oneM2M service platform are assessed. The selected architectural alternative is the basis for the prototypical implementation as proof-of-concept.

In Chapter 7, the prototypical implementation of the enhanced data exchange capabilities is presented. For this purpose, the used technologies are introduced followed by the description of the building blocks, which are introduced to implement the enhanced data exchange capabilities. Afterwards, setup and execution of experiments is described in detail which are the foundation for three experiments and related estimations that prove the advancements of the proposed enhancements. Finally, this is put into context with respect to network efficiency and performance considerations, as well as qualitative attributes.

This leads to the conclusion and the summary of the achievements of this research, followed by limitations and possible future work, which is the content of Chapter 8.

2 Review of the Automotive Environment

This research has its motivation and foundation in the evolution of the automotive domain and related challenges. In this regard, this chapter aims to provide a comprehensive background of the automotive environment. It starts top-down with a review of the three automotive functional domains that currently primarily drive the automotive developments, including their historical background, status quo, and anticipated future functionalities. Afterwards, the automotive system landscape is considered bottom-up, by means of the components and internal and external communication technologies, which in its entirety build the landscape in which the applications from the functional domains should be integrated. The emergence of the functional domains together with the software and system landscape facilitate two things: on the one hand, they support the understanding of the characteristics and challenges within the automotive domain, which are presented in Section 2.3. On the other hand, they enable the categorisation and localisation of related work regarding software architectures and platforms, which is content of Section 2.4. All considerations leading to the problem statement, presented in Section 2.6, that is the starting point for further research.

2.1 Functional Domains

Several functional domains drive today's automotive software and system development. This section introduces the three main functional domains with respect to software and connectivity, namely: In-Vehicle Infotainment (IVI), Advanced Driver Assistance Systems (ADAS), and Intelligent Transportation Systems (ITS), see Figure 2.1.

Initially separated, the borders between those domains in many cases become blurred today. In any case, those functional domains clash inside the vehicle or while considering all vehicular usage scenarios, which generates a great deal of challenges for vehicular software and system development. However, since the introduction of ever-increasing functionalities is no end in itself but aims various benefits for the driver and passengers, the three main functional domains are subsequently introduced with respect to their history as well as their status quo and anticipated future developments.

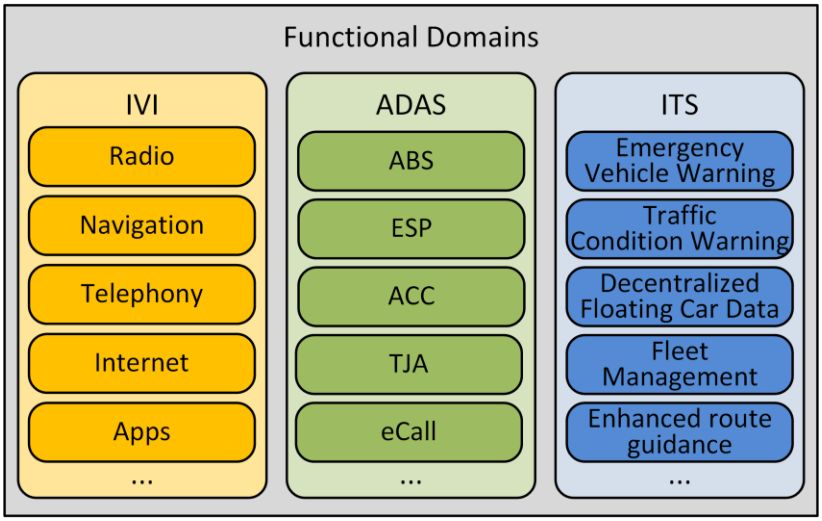

Figure 2.1: Functional Domains

2.1.1 In-Vehicle Infotainment

In 1932, Blaupunkt introduced the first car radio in Europe (Blaupunkt, 2017). Since then, especially within the last decade, the car radio has evolved to a sophisticated In-Vehicle Infotainment system (IVI), also referred to as Automotive Infotainment System or In-Car Multimedia System (ICM-System), which offers a continuously growing number of functionalities to the passengers. Herein, the term infotainment is a portmanteau between information and entertainment. In the automotive context – in contrast to some critical discussions in the context of journalism – it can be understood more literally as a system that provides both information (e.g., of the car or traffic situation) and entertainment capabilities (e.g., radio or audio/video-player). The term multimedia emphasis that content can be presented with different media types, such as textual information, animations, and audio outputs.

With the integration of satellite navigation, television and video-player capabilities into car radios, colour displays and Graphical User Interfaces (GUI) were introduced. The functionalities are controlled through hard/soft keys, push-rotary buttons, or increasingly touch screens/pads, gestures, and voice commands. These powerful input and output capabilities of IVI systems, together with the increasing number of overall vehicular functionalities that need to be displayed and controlled, paved the way for the IVI system to become the main Human-Machine Interface (HMI) for non-driving functions of the vehicle.

Accordingly, current IVI systems are also used to display car maintenance information, interactive car manuals, and to enable the setup of diverse car functions, such as air-conditioning, driving modes, and driver assistance systems. Accenture identified in their study "Perspectives on In-Vehicle Infotainment Systems and Telematics" the following three main trends impacting IVI systems: eco-efficiency, security and safety, and comfort (Accenture, 2012). In that context, the demands for IVI systems are influenced by Telematics (Telecommunication and Information Technology) use cases. Since vehicular Telematics are

also related to ITS, this is one example where the IVI functional domain has overlaps with other domains. However, ITS will be discussed in more detail in a separate subsequent section (see Section 2.1.3).

Moreover, mobile phones have become connected with the IVI systems through Bluetooth or cable connections. Initially, this should enable hands-free telephony by use of the vehicular sound system and microphone. Additionally, this connection can enable the usage of on-board antenna (e.g., on the vehicle roof) instead of the built-in antenna of the mobile phone, to enhance the link quality with the mobile cellular network. Nowadays, in some cases, the mobile or smartphone connections are also used to provide Internet connectivity to the IVI system. In contrast, particularly high-end IVI systems use in-vehicle cellular modems for Internet connectivity independently from connected mobile phones. Those IVI systems might also share their Internet connectivity by means of a vehicle-internal Wireless Local Area Network (WLAN) hotspot for other devices brought-in by the passengers.

No matter what connection type is used, current premium IVI systems are connected with the Internet. This, for instance, is used to receive extended traffic information, to enhance the navigation screen with dynamically loaded satellite picture overlays, or to enable location-based searches and Internet radio or music streaming.

Coinciding with the progress of IVI systems, mobile phones have experienced a substantial evolution towards “smart” phones, changing the whole mobile devices landscape seen previously. Performance improvements of mobile hardware, increasingly powerful mobile networks together with advanced mobile operating systems and integrated application stores paved the way for the emergence of huge ecosystems (Basole & Karla, 2011). The millions of “apps” available enable the comprehensive customisation of the smartphone functionalities, according to the users’ wishes. These developments in the Consumer Electronics (CE) domain already have impacts on design and range of functions of IVI systems. Gryc and Johnson found in (2011) that “car buyers now expect automakers to keep up with the frantic pace of development maintained by consumer electronics, and in particular the ever-increasing library of apps and services available on smartphones“. Lastly, also the provision of social networks to vehicles is an interesting use case (Lequerica, Garcia Longaron, & Ruiz, 2010).

Indeed, two different solutions currently find their way individually or in combination to IVI systems, providing passengers functionalities already known from their smartphone ecosystems:

1. **OEM-specific app stores or services**
   Some OEMs have started to integrate their own app stores or to provide their own services to their vehicles. Often popular apps from smartphone ecosystems, such as Pandora, Spotify, and Facebook are also made available for the car (Johanning & Mildner, 2015). This might be done either by providing an OEM-specific Software

Development Kit (SDK) that enables app developers to port their apps to the vehicle ecosystem by themselves, or the OEM itself takes over this task.

2. **Deep smartphone integration in the IVI system**
   With this approach, the smartphone remains the system which hosts, runs, and hence brings in the apps into the IVI system. The latter only provides the output capabilities (e.g., screen, speakers) and input capabilities (e.g., touchscreen, voice commands). Currently, particularly three technologies exist that enable this kind of smartphone integration, namely: MirrorLink, Apple CarPlay, and Android Auto (Johanning & Mildner, 2015). Depending on the actual technology, apps are displayed in the vehicle with the same or a specific layout. However, usually the apps have to implement particular functionalities, which is why typically only a subset of all smartphone apps is available by means of those technologies. The available smartphone apps may duplicate or even make obsolete vehicle-internal functionalities. For example: Google Maps may substitute an IVI-internal navigation system, and music or radio streaming apps, such as Spotify, or Pandora, may substitute the traditional FM/DAB-radio or CD/MP3-player.

Finally, it shall also be noted here that OEM may additionally provide apps or web-frontends for the customer that enable the remote access of vehicle functionalities or data. For example, driving destinations and routes can be sent to the car in advance of a trip or current fuel level, average economy, and remaining range, climate control, and positioning information are retrievable.

2.1.2 Advanced Driver Assistance Systems towards Highly Automated Driving

The continuous enhancement of traffic safety has been a professed goal of automotive researchers, engineers, politicians, and society for many decades. The invention of safety cells, seatbelts, multilayer glasses in the 1960s poses the starting point with passive safety mechanisms which are aimed for crash mitigation purposes. Today, passive safety mechanisms still play an important role and are still being improved, and this will continue to be the case for several years to come.

Nevertheless, since Bosch introduced the electronic Antilock Brake System (ABS) as one of the first active safety mechanisms in 1978, the focus has started to shift towards crash prevention up to the final goal of crash-free driving (Siebenpfeiffer, 2014). The ABS was followed by Traction Control Systems (TCS) which are combined and enhanced to Electronic Stability Programs (ESP). Looking at the emergence of these functions, the intervention timespan before an accident may occur increases. This leads to the development of Advanced Driver Assistance Systems (ADAS), which nowadays aim to support the driver in more and more situations. As part of this development, the vehicles are becoming equipped with an increasing number of sensors, such as visual sensors (e.g., Ultrasonic, Radar, Lidar, Camera).

These facilitate functionalities, such as Forward Collision Warning (FCW), Lane Departure Warning (LDW), Blind Spot Detection (BSD), Traffic Sign Recognition (TSR), and Driver Monitoring (Cacilo et al., 2016).

The concrete driver assistance could be realised in the form of warnings, such as the display of messages or warning tones, which is the reason why ADAS also have influence on the IVI system and particularly (functional) safety requirements to the latter (Schneider & Nett, 2014). Furthermore, the driver could be warned haptically, e.g., through steering wheel vibrations or belt tightening. While the previously-named functionalities FCW, LDW, BSD, and TSR are aimed to providing driver assistance in terms of support and warnings, several functionalities are introduced that assist actively, similar to ESP. For example: An Adaptive Cruise Control (ACC) can actively control the lateral dynamic of a vehicle by means of retaining the distance to preceding vehicle depending on the current speed (within the system boundaries[1]). In contrast, a Lane Keeping Assistance (LKA) can actively control the longitudinal vehicular dynamics through active steering interventions (within the system boundaries[1]).

Recently, the combination of such single functionalities facilitated more enhanced functionalities that ultimately shall lead to Highly Automated Driving (HAD) and Autonomous Driving (AD). For example: ACC and LKW can be combined to a Traffic Jam Assistant (TJA) or Traffic Jam Chauffeurs (TJC) that enable lateral and longitudinal control of the vehicle during traffic jams on motorways. TJAs are typically one of the first HAD functionalities, since in this situation relatively small differential speeds between the vehicles, the fact that everyone drives in the same direction on a motorway, as well as the absence of traffic lights, etc., ease the complexity of the driving task and hence ease its development and introduction.

Until the vision of autonomous driving with zero accidents becomes reality, the effective rescue of casualties remains a significant task. Thus, the automatic emergency call (eCall) that in case of an accident autonomously calls the responsible Public Safety Answering Point (PSAP), i.e. an emergency call centre, is another important functionality of future vehicles. To speed up the rescue, the eCall will transmit the current position of the car while additionally establishing a voice connection to the PSAP. Effective 31 March 2018, the European Commission has regulated that this functionality must be incorporated in every newly manufactured vehicle (European Parliament, Council of the European Union, 2015). The eCall can also be considered as another step towards an Intelligent Transportation System:

[1] Typical system boundaries are the nominal performance of the sensors used to detect preceding vehicles, as well as defined/configured boundaries. The latter for ACC could be the maximum allowed deceleration, which typically is less than the maximum possible deceleration, at least for certain speed ranges. For LKA, the defined boundaries may be, in addition to others, the maximum steering wheel angle depending on the speed.

2.1.3 Intelligent Transportation Systems

Another perspective on functionalities associated with connected vehicles constitute (cooperative) Intelligent Transportation Systems[2] (ITS), also referred to as Transportation Cyber-Physical Systems (T-CPS) (Al-Fuqaha, Guizani, Mohammadi, Aledhari, & Ayyash, 2015). For several years, they have been subject of research and standardisation activities, addressing partial aspects of the overall vision (cf. COMeSafety, 2013; simTD-Consortium, 2013). Figure 2.2 shows various ITS scenarios at a glance, e.g., crash avoidance, fleet management, and travel assistance.

Figure 2.2 Intelligent Transportation System scenarios (Source: ETSI TR 102 638, 2009)

There are different viewpoints on ITS and their role within smart city considerations (Naphade, Banavar, Harrison, Paraszczak, & Morris, 2011). This research utilises the European ITS Communication Architecture, whereas ITS applications can generally be categorised in "Traffic Safety", "Traffic Efficiency", and "Value-Added Services" (Bechler et al., 2010; Kosch et al., 2009). This section is intended to briefly introduce the background of each category and to present related applications according to the ETSI "Basic Set of Applications" (ETSI TR 102 638, 2009).

Traffic Safety

While first efforts to increase traffic safety were about mitigating the consequences of a crash (see Section 2.1.2), active safety technologies to completely avoid accidents are being introduced exponentially towards the vision of accident-free driving (Eskandarian, 2012,

[2] Also known as: Intelligent Transport Systems

p. 710). The connection of vehicles with each other (Car2Car) and with infrastructure, e.g., traffic lights (Car2Infrastructure) is considered a major step towards this vision. Besides completely new applications, Car2X (see Section 2.2.2) can further increase the potential of existing active and passive safety systems. Table 2.1 names the basic set of "Active road safety" applications, according to ETSI (ETSI TR 102 638, 2009, p. 18).

Application Class	Application	Use Case
Active road safety	Driving assistance - Cooperative awareness	Emergency vehicle warning
		Slow vehicle indication
		Intersection collision warning
		Motorcycle approaching indication
	Driving assistance - Road Hazard Warning	Emergency electronic brake lights
		Wrong way driving warning
		Stationary vehicle – accident
		Stationary vehicle – vehicle problem
		Traffic condition warning
		Signal violation warning
		Roadwork warning
		Collision risk warning
		Decentralized floating car data - Hazardous location
		Decentralized floating car data - Precipitations
		Decentralized floating car data - Road adhesion
		Decentralized floating car data - Visibility
		Decentralized floating car data - Wind

Table 2.1: ETSI basic set of applications related to traffic safety (ETSI TR 102 638, 2009, p. 18)

Traffic Efficiency

Mobility is a central property of our modern society, but a growing number of traffic incidents occur and inhibits efficient and reliable travelling. Traffic jams are extremely costly, and the higher fuel consumption burdens the environment. Hence improving traffic efficiency is an objective not only for the driver but also for society. ITS should enable advanced traffic efficiency scenarios. Table 2.2 lists the basic set of applications within the class "Cooperative traffic efficiency", according to ETSI (ETSI TR 102 638, 2009, p. 18).

Application Class	Application	Use Case
Cooperative traffic efficiency	Speed management	Regulatory / contextual speed limits notification
		Traffic light optimal speed advisory
	Cooperative navigation	Traffic information and recommended itinerary
		Enhanced route guidance and navigation
		Limited access warning and detour notification
		In-vehicle signage

Table 2.2: ETSI basic set of applications related to traffic efficiency (ETSI TR 102 638, 2009, p. 18)

Value-Added Services

Further use cases are subsumed as "Value-Added Services". This category shows the most overlaps with the future IVI-System functions. Hence, the context (i.e., functional domain) in which these applications are introduced is not fixed. In more detail, two application classes of the ETSI Basic Set of Applications, namely: "Cooperative local services" and "Global Internet services" (ETSI TR 102 638, 2009, p. 18), can be assigned to "Value-Added Services". These are listed in Table 2.3.

Application Class	Application	Use Case
Cooperative local services	Location based services	Point of Interest notification
		Automatic access control and parking management
		ITS local electronic commerce
		Media downloading
Global Internet services	Communities services	Insurance and financial services
		Fleet management
		Loading zone management
	ITS station life cycle management	Vehicle software / data provisioning and update
		Vehicle and RSU data calibration

Table 2.3: ETSI basic set of applications related to traffic efficiency (ETSI TR 102 638, 2009, p. 18)

2.2 Automotive Software and System Landscape

To implement the previously described IVI, ADAS, and ITS functionalities, they must be integrated somewhere into the automotive software and system landscape. In a modern connected vehicle, this automotive software and system landscape constitutes of a highly distributed system, including many different computing components that are interconnected by several wireline and wireless communication technologies. Figure 2.3 provides a schematic overview and indicates those components that are typically in OEM ownership, which means that their design and range of functions can be directly controlled.

While the actual component and communication technologies configuration varies, functionalities of a connected vehicle in general can get:

- "Built-in" into on-board equipment.
- "Brought-in", e.g., through the connection of CE devices.
- "Beamed-in", if they are realised off-board, e.g., on server infrastructure in the backend and connected within the vehicle (Venkatesh Prasad et al., 2010).

A more detailed look on the typical computing components and communication technologies of the automotive software and system landscape is presented below.

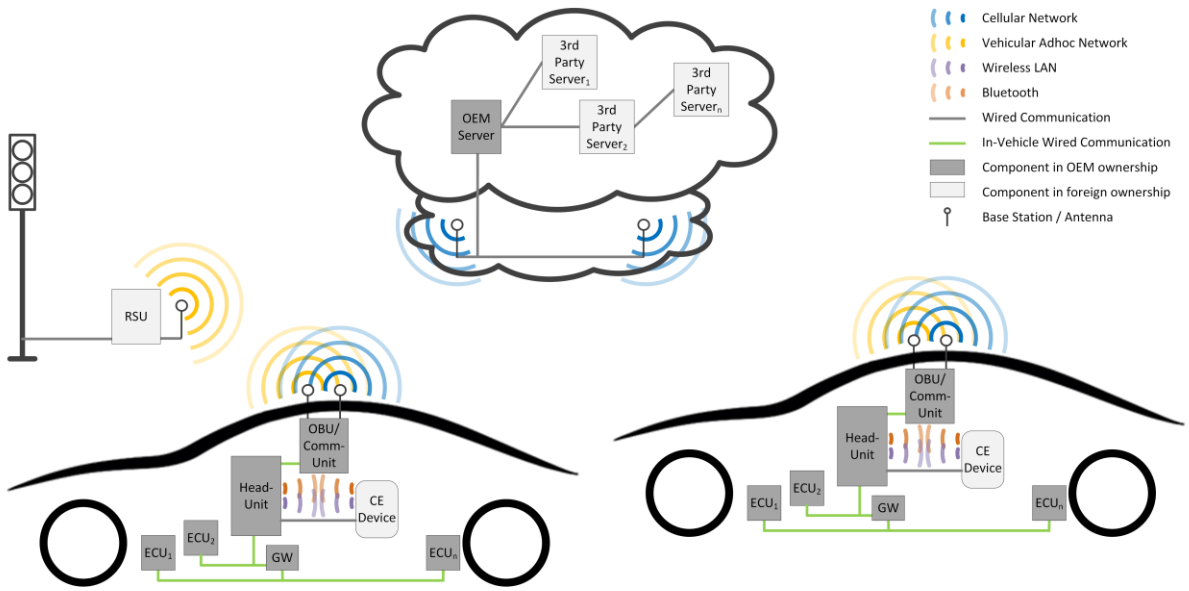

Figure 2.3: Automotive software and system landscape

2.2.1 Computing Components

Computing components within the vehicle are referred to as embedded systems:

> *"Embedded systems are information processing systems embedded into enclosing products." (Marwedel, 2010)*

Accordingly, the software running on these embedded systems is called embedded software:

> *"Embedded Software is software integrated with physical processes." (E. A. Lee, 2006)*

Depending on the context, and if the inter-relationship between different embedded systems and their physical environment should be emphasised, the term Cyber-Physical System (CPS) is used, which can be defined as follows:

> *"Cyber-Physical Systems (CPS) are integrations of computation and physical processes" (E. A. Lee, 2007)*

The automotive system landscape of a modern vehicle consists of various computing components. While their actual design and concrete vehicle setups vary, the basic types of units are described below:

Electronic Control Unit

The Electronic Control Unit (ECU) denotes an embedded system used in the automotive domain. Modern vehicles have a high number of various ECUs that are typically tailored to their specific task. For example, there are ECUs particularly dedicated to engine control, brake control, door control, which are responsible for physical control of the related vehicular components. Automotive ECUs must be designed for the harsh environment in which they are operated such as wide ranges of temperature and humidity, shock resistance, and Electro-

Magnetic Compatibility (EMC). The computational and memory capabilities of many ECUs are still very limited, e.g., 8 Bit processors with 8 MHz, and few KB memory (Sagstetter, 2016). A special form of an ECU is an On-Board Unit (OBU), also referred to as Vehicle Communication Gateway (VCG). They constitute the in-vehicle counterpart for external communication parties such as Road-Side Units (RSUs) and servers in the backend.

Head-unit

Another specialised ECU is the "so called" head-unit which in modern vehicles replaces the former car radio. It is the main computing hardware of the IVI system (Alt, 2009, p. 7; Wietzke & Tran, 2005). Although it is still an embedded system, it is nowadays a powerful multimedia Car-PC, typically equipped with large (touch-)screens to display or control the navigation system, music player, FM/DAB radio, as well as car status and configuration (see Section 2.1.1). Smethurst found in (2010) that "this device is the most complex component in a modern automobile by software volume. In a high-end device, approximately 70% of the total code in a car will be in that single device."

Consumer Electronic Devices

Section 2.1.1 already dealt with the history and motivation that lead to the integration of CE devices, in particular smartphones and tablets, into the automotive system landscape. For some time, this integration has gone beyond hands-free telephony: Several technologies exist that facilitate the display output of the CE device on the in-vehicle display, the audio input and output with vehicular microphone and speakers, and the control of the CE device, e.g., by use of the touchscreen. Thus, functionalities (i.e., apps) from the CE device are being integrated into the vehicle's IVI system and control concept which is also referred to as terminal mode (Bose, Brakensiek, & Park, 2010; Bose, Brakensiek, Park, & Lester, 2011). This could optionally include an adoption of the apps to the in-vehicle screen and vehicular context. Examples are, in addition to others, Apple "iOS CarPlay" (Apple, 2016), Google "Android Auto" (Android Auto, 2016), or "MirrorLink" (Car Connectivity Consortium, 2016).

Backend Server

Modern vehicles are connected with the Internet. This is driven, e.g., by enhanced IVI systems (see Section 2.1.1) and to facilitate updates or upgrades of functionalities Over-The-Air (see Sections 2.3.1 and 2.3.4). Not least the eCall as mandatory functionality of vehicles latest by 31 March 2018 (see Section 2.1.2) will further increase the number of vehicles connected with the Internet, since this functionality requires the equipment with wireless cellular network hardware anyway. As a result, external entities are becoming an integrated part of the automotive software and system landscape of a connected vehicle. In addition to RSUs and others, in particular backend server facilities are gaining importance here. In this regard, the objectives and technologies related to cloud computing (cf. Baun, Kunze, Nimis, & Tai, 2009) have found their way to the vehicular domain too, which is the reason why such backend servers are sometimes also referred to as "cloud server" or just "cloud". From the point of view of an

OEM, these servers basically can be controlled by themselves (i.e., OEM server), or by 3rd parties (i.e., 3rd party server). However, considering different cloud server architectures, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Software as a Service (SaaS), this labelling is not a statement regarding the actual operator of the server or infrastructure, but relates to the authority that provides the functionalities or services by means of this server.

2.2.2 Communication Technologies

The components of the automotive software and systems landscape are inter-connected by means of various in-vehicle and external communication technologies. The most important are introduced in the following:

Internal

Various vehicle-internal communication technologies exist to connect the ECUs, OBU, and head-unit with each other and with sensors or actuators. Today, the most common are the Controller Area Network (CAN), Time-triggered CAN, Local Interconnect Network (LIN), FlexRay, Media Oriented System Transport (MOST), and Automotive Ethernet. Besides physical media (fibre vs. copper), topologies, and costs, they differ with regard to maximum bit rate and messaging (event-triggered or time-triggered). The latter impacts their applicability for real-time applications (Kopetz, 2011).

	CAN	TTCAN	FlexRay	LIN	MOST	Automotive Ethernet
Max. bit rate	1 Mbps	10 Mbps	10 Mbps	19.2 Kbps	150 Mbps	100 Mbps
Messaging	Event-triggered	Time-triggered/ Event-triggered	Time-triggered/ Event-triggered	Time-triggered	Time-triggered/ Event-triggered	Event-triggered

Table 2.4: Selected vehicle-internal communication technologies (Sagstetter, 2016; Talbot & Ren, 2009; W. Zimmermann & Schmidgall, 2014)

External

The connected vehicle is equipped with external communication capabilities. These can be classified as wireless infrastructure-based and wireless ad hoc communication technologies. They are introduced below.

Wireless Infrastructure-based Communication

Wireless cellular networks, or Public Land Mobile Networks (PLMNs), are the most common infrastructure-based wireless communication technologies. They are particularly suitable for the connection of vehicles with the Internet, since the various cell sizes support an appropriate scalability between coverage and capacity, and because they are able to deal with high mobility speeds as they occur with vehicles. The evolution of digital wireless cellular networks, starting

with 2nd Generation (2G), referred to as Global System for Mobile Communications (GSM), and continuing with 3rd Generation (3G), referred to as Universal Mobile Telecommunications System (UMTS), further progressing to the 4th Generation (4G), referred to as Long Term Evolution Advanced (LTE-A), all of the mentioned enabled significant advances (M. R. Bhalla & Bhalla, 2010; Sauter, 2015). The currently common LTE-A, in detail 3GPP release 10, theoretically supports up to 3 Gbps downlinks, and 1.5 Gbps uplinks (Wannstrom, 2013). In the future, 5th Generation (5G) will support even higher peak data rates up to 25 Gb/s with 3GPP release 13 (J. Lee et al., 2016). Furthermore, 5G also brings, e.g., reduced latency (one-way ~ 1ms with 3GPP release 14), multicast services, and potentially even direct device-to-device (D2D) communication that would provide an alternative approach for Car2Car communication (Hoymann et al., 2016; J. Lee et al., 2016; Mumtaz, Huq, & Rodriguez, 2014; Tehrani, Uysal, & Yanikomeroglu, 2014).

However, PLMN evolution is not only about radio interface enhancements – the concept of Mobile Edge Computing (MEC) describes infrastructure advancements which is considered as one of the key technologies towards 5G (Beck, Werner, Feld, & Schimper, 2014; Y. C. Hu, Patel, Sabella, Sprecher, & Young, 2015): With MEC, Over-The-Top (OTT) services, such as automotive services initially operated in the OEM backend, can now be operated at the edge of the network (e.g., base station). The advantages of such distributed service deployment compared to pure backend deployment are, e.g., reduced latency, increased overall network efficiency, and thus increased scalability of the automotive application. MEC complements efforts, such as Network Function Virtualization (NFV) and Software-Defined Networking (SDN) (Rost et al., 2016). NFV is about the virtualisation of network functions that, e.g., enables their relocation to and the usage of standard servers. SDN separates the control and data plane, which enables increased network programmability, e.g., through centralisation of control. To summarise, MEC, NFV, and SDN are approaches that contribute to 5G's ability to provide the necessary Quality of Service (QoS) for the functionalities and use cases (see Sections 2.1 and 3.3.5). In addition, vehicles might also be equipped with standard Wireless Local Area Network (WLAN) IEEE 802.11a/b/g/n (Mohammad, Rasheed, & Qayyum, 2011) modules to connect to public and private Hotspots during vehicle standstill, but they are of minor importance in the context of this research.

Vehicular Ad Hoc Communication

Wireless ad hoc communication networks in the context of cars are referred to as Vehicular Ad hoc NETworks (VANETs) which represent a specialisation of Mobile Ad hoc NETworks (MANETs) for automotive (Abdalla, Abu-Rgheff, & Senouci, 2007; Mohammad et al., 2011; Picone, Busanelli, Amoretti, Zanichelli, & Ferrari, 2015). VANETs are being developed for the purpose of decentralised dynamic (ad hoc) multi hop communication of cars with other cars

(Car2Car[3]) and with infrastructure, e.g., RSUs or traffic lights (Car2Infrastructure[4]). These communication scenarios are aggregated by the term Car2X[5] communication.

Currently, there are several activities toward standardisation, such as within the context of IEEE 802.11 as IEEE 802.11p WAVE (Wireless Access in Vehicular Environments) (Jiang & Delgrossi, 2008; Mohammad et al., 2011). IEEE 802.11p is basically a modification of 802.11a for vehicular purposes. It uses the Dedicated Short Range Communication (DSRC) frequency spectrum. IEEE 802.11p defines the Physical Layer and it is supplemented by upper layer IEEE 1609 standards (Faezipour, Nourani, Saeed, & Addepalli, 2012; Jiang & Delgrossi, 2008).

VANETs have their foundation in ADAS and ITS scenarios (Dar, Bakhouya, Gaber, Wack, & Lorenz, 2010), hence are predominately discussed together with time-critical, safety-related, and locally limited use cases such as "Collision Warning", which require very short delays (Kosch et al., 2009). In contrast, wireless cellular networks, are discussed mainly for enhanced IVI use cases and superior communication scenarios with a higher tolerable delay, e.g., "Internet Connectivity" or "Traffic Management". However, both technologies are not restricted to the mentioned scenarios. There do exist solutions to provide Internet connectivity over VANETs (Festag, 2014; Sandonis, Soto, Calderón, & Urueña, 2016). Besides, evolved wireless cellular networks may provide small enough packet delay to make safety-related use cases feasible (cf. simTD-Consortium, 2009, pp. 60-62). Particularly the introduction of 5G wireless cellular networks with D2D capabilities and the MEC concept (see above) can enable a significant shift here (Seo, Lee, Yasukawa, Peng, & Sartori, 2016). Nevertheless, it is not part of this research project to define the borderline between those two communication techniques, and both will continue to coexist and complement each other. Nonetheless, this research focuses on wireless cellular networks and hence also on use cases, which these days are considered as reasonable with such connections.

2.3 Domain Characteristics and Challenges

In today's vehicles, software plays a major role for the overall range of functions. Already in 2007, premium cars had 2000 software-based functions (Broy, Kruger, Pretschner, & Salzmann, 2007). Therefore, current vehicles have become software-intensive systems:

> *A **software-intensive system** is any system where software contributes essential influences to the design, construction, deployment, and evolution of the system as a whole. (IEEE Computer Society, 2007)*

[3] Also known as: Car-2-Car, C2C, Vehicle2Vehicle, Vehicle-2-Vehicle, V2V

[4] Also known as: Car-2-Infrastructure, Vehicle2Infrastructure, Vehicle-2-Infrastructure, V2I

[5] Also known as: Car-2-X, Vehicle2X, Vehicle-2-X, V2X

The development of these automotive software-intensive systems is a very complex task. Important domain characteristics and challenges are discussed in the following.

2.3.1 Programmable Vehicle

Today's vehicles already have many sensors, actuators and ECUs. This fact in itself, but particularly in their inter-connection through in-vehicle networks, facilitate new functionalities just by introducing new software, making the vehicle increasingly a "programmable vehicle". Some examples are:

- The availability of outside temperature information can be used to automatically switch on the exterior mirrors heating if the temperature is below a specified value.
- The measurement of the rain sensor can be used to automatically trigger the rear wiper when the reverse gear is engaged.
- The right exterior mirror can be automatically lowered to show the curbs, if the reverse gear is engaged to ease exact parking.
- The monitoring of the rotational speeds of each wheel that was initially introduced for anti-lock braking purposes can be used to implement an indirect Tire Pressure Monitoring System (TPMS). The indirect TPMS software can utilize the fact that tires with a lower inflation have a smaller diameter which in turn causes a higher rotational speed.
- A steering wheel angle sensor has been introduced as part of the ESP system. A software, the purpose of which is to evaluate the angle values over a time period, can be used to determine driver's fatigue. This considers the fact that studies showed a correlation between steering behaviour and fatigue, e.g., corrections used to be made less often but frequency increases when the driver is tired (Krajewski, Sommer, Trutschel, Edwards, & Golz, 2009; Takei & Furukawa, 2005).
- Modern adaptive headlight/high-beam levelling systems can use map data from the vehicular navigation system in addition to built-in sensors such as a camera (Winner, Hakuli, & Wolf, 2012, p. 1064). Whereas the camera can control the adaptive lightning with respect to oncoming and surrounding traffic, the map data enables, e.g., predictive bend lightning, adaptive levelling at intersections and roundabouts, and exact high-beam activation/deactivation on city boundaries.
- The vehicle route out of the navigation system might be useful to estimate upcoming wireless access network coverage and capacity to facilitate enhanced caching strategies for music streaming (Protzmann, Massow, & Radusch, 2014).

These examples show that new functionality does not necessarily need the "installation of an additional box" and that the programmable vehicle is becoming a reality. This generates new technical and business opportunities for functional upgrades after purchase of the car, over and beyond common bug fixes or map updates. Indeed, some OEMs have already started to

integrate a kind of application store into their IVI system or offer own services, such as enhanced traffic information.

The programmable vehicle also facilitates functional upgrades beyond IVI applications. An impressive example for this was given by Tesla Motors, who recently provided an autopilot functionality for U.S. $2,500 as Over-The-Air (OTA) update to their Model S owners (Tesla Motors, 2016).

Finally, the programmable vehicle is another aspect conducing to the development that the functionalities of the vehicle will no longer remain static during its lifetime, which is discussed in the subsequent Section 2.3.4 in more detail.

2.3.2 Distributed Functionalities

At the beginning of automotive ECU development, there was a "one box-one function paradigm" (Natale & Sangiovanni-Vincentelli, 2010). It easily facilitated separation of concerns, and the integration of new functionalities was realised by just adding another "box" (i.e., an ECU). This "has rapidly led to a proliferation in the number of ECUs" (Natale & Sangiovanni-Vincentelli, 2010). Although vehicle-internal networks with bus-topologies (see Section 2.2.2) between ECUs were introduced to reduce wiring complexity and overall cabling amount as well as costs and weight, this approach is being faced with further limitations. These are not only necessary space, weight, and power requirements for all these ECUs – it is in particular the fact that the number of distributed functionalities is rising.

The previous section already provided some examples for advanced software functionalities that are distributed across ECUs. Further, fundamental, apparently simple functionalities such as the locking system confirm the distributed nature of today's automotive software (cf. I. H. Krüger, Nelson, & Prasad, 2004): At its beginning, the vehicle locking system was a purely mechanical system, where a key was used to unlock the door or trunk. The next step was the introduction of an electrified central locking system, where the key could unlock/lock all doors and the trunk at once. Modern vehicles have a radio-controlled central locking system where the successful unlocking or locking of the vehicle might be signalled through the indicator lights and the horn. Upon unlocking, key-individual settings might be applied such as seat and mirror positions and climate or tuner configuration. Further, the vehicle may automatically lock the car doors and trunk when is being driven above walking speed to protect passengers against attacks. This, however, requires that the vehicle should not only unlock all doors and the trunk at the end of the trip, but particularly due to safety considerations, it must automatically unlock the doors in case of an accident (of sufficient severity) so as not to hinder rescue operations. Latest connected vehicles already facilitate the locking/unlocking of the vehicle through smartphone applications of the OEM and via service telephone call to the OEM hotline, e.g., when a key gets lost. The list could be extended, e.g., with connections to the car alarm, engine immobilizer, coming home and leaving home exterior lights, etc. However, already the facts

listed indicate the highly distributed nature of a significant number of vehicular functionalities. Pretschner et al. named a number of 18 ECUs over which the overall central locking system is distributed (Pretschner et al., 2007).

The "one box-one function paradigm" cannot be continued simply because of the ever-increasing number of functionalities that would lead to too many ECUs. Besides, it is unrewarding in view of the number of ever-increasing distributed functionalities. Hence, the integration density in terms of functions per ECU is increased, whereas the number of ECUs lastly remains at about 100 (Schäuffele & Zurawka, 2013). Moreover, different dimensioning of future ECUs and adopted software engineering practices that lead to functionalities that are more hardware-independent could enable an even higher integration density and the significant reduction of the overall number ECUs. Recently, OEMs have started to think about only very few ECUs that are less tailored but more general and powerful, which thus would consequently reflect the already named shifts in domain characteristics and challenges. One example is Audi's "zentrales Fahrerassistenzsteuergerät"[6] (zFAS) (Audi AG, 2016b; Johanning & Mildner, 2015).

The highly distributed nature of an increasing number of vehicular functionalities leads to many cross dependencies between different functionalities and ECUs. However, this necessary and intentional interaction of functions, also referred to as features, also causes "phenomena like unintentional feature interaction" (Pretschner et al., 2007). This term originates in the telecommunication domain and expresses the fact that "the system behaviour as a whole does not satisfy the separate feature specifications" (Zave, 1993). Broy noticed in (2006a): "So far, the understanding of these feature interactions between the different functions in the car is insufficient." With future connected vehicles, the distribution of functionalities is widened including external entities such as server of the OEM or third party or other vehicles or infrastructure in terms of Car2X functionalities. This is expected to intensify these challenges, although related architecture principles, such as service-orientation, may provide adequate means to address them. However, it is a task for future automotive software architectures and related development processes to provide solutions to this challenge.

2.3.3 Reuse of Functionalities

Similar to equal parts strategies in the context of mechanics and electronics, functionalities provided by software are mostly not exclusively developed for a single vehicle series. Hence, reusability is a target. Reusability begins inside a single vehicle, where sensors as well as software functionalities should only exist once (Broy, Reichart, & Rothhardt, 2011). It extends to considering programmable vehicles (see Section 2.3.1), where it might be envisaged to roll out new functionalities also to existing vehicles from the same or even different vehicle series

[6] Translation: One central control unit for driver assistance

(Broy et al., 2011). Finally, reuse of functionalities should be possible across different vehicle generations (Broy et al., 2011).

Hereby, reusability is not only a functional goal. With respect to quality considerations related to the maturity level of (often highly complex) software functionalities, recurring implementations of the same functionalities must be avoided. Broy stated in 2006: "In many sub-domains the functionality from one car generation to the next is only changed and enhanced by 10 % while more than 90 % of the software is rewritten. The reason is a low level, hardware specific implementation, which makes it difficult to change, adopt, and port existing code." (Broy, 2006a). In the end, all named aspects regarding insufficient reuse of functionalities results in enormous costs. Broy raised the following calculation in (Broy et al., 2007): The development costs for all electronic parts of the vehicle are 300 million Euros and more of which one to two thirds is related to software. Raising the reuse to 50% to a next car generation could save up to 100 million Euros of development costs (Broy et al., 2007). This is more than the potential savings of highly optimised software functionalities for individual hardware (Broy et al., 2007) which is a typical characteristic of today's automotive software and system development.

Although the automotive industry might have achieved improvements on the software reuse percentage since 2006, against the background of the envisaged future functionalities and resulting changes within the vehicular software landscape, reusability remains a key requirement and challenge.

2.3.4 Life Cycle and Innovation Cycle

A typical development of a new vehicle takes about three years until Start of Production (SoP) (Schäuffele & Zurawka, 2013, p. 21). Afterwards, the vehicle is usually produced for approximately seven years and even the final cars of the production period should be able to be maintained for at least 15 years after the purchase (Broy et al., 2007). This leads to a total lifecycle of one vehicle generation of at minimum of 20-25 years.

In contrast, the lifecycle of hardware, e.g., processors, is typically less than five years (Broy et al., 2007), which requires new hardware revisions of ECUs. This means that "already after the first 3 years of production 20 to 30 percent of the ECUs in the car typically have to be replaced due to discontinued ECUs." (Broy, 2006a) This affects the vehicle production as well as maintenance, i.e., ECU replacement due to individual failure.

Consumer Electronics devices, e.g., smartphones, "are launched, upgraded, and sometimes even rendered obsolete in months" (Gryc, 2011). The typical age of a smartphone is about 2-3 years in Germany. Considering the average age of vehicles in Germany of 9 years, this leads to an average of 3-4 smartphones that one vehicle owner wants to use with Bluetooth hands-free kit of the vehicle, etc. Considering additional drivers and owners of the vehicle and its intended

overall lifecycle, the number of smartphones (and mobile operating system generations) could be two-digit.

Although many basic vehicular functionalities, such as speedometer, FM-radio, etc. could be expected to for the most part remain constant, an increasing number of overall functionalities might change: With respect to the programmable vehicle, more frequently, and even during the production phase, OEMs may want to integrate new software features e.g., of other vehicle series (see Section 2.3.1). Particularly within the IVI domain, frequent updates could be expected, especially if application stores become available, similar to the CE domain. Many smartphone-applications such as Facebook and Spotify are updated within days to a few weeks (Shimizu, 2004). If those applications or related ecosystems in general are being made available to the IVI system, related life cycles and innovation cycles hit (at least parts of) the automotive domain, generating new challenges, for example:

Until recently, the software of the vehicle was only updated during service in a garage and those updates were mainly related to map data updates and fixing of bugs and less related to new or upgraded functionalities. However, future connected vehicles and related functionalities will become updated OTA and hence independent of the garage. This raises technical and organisational challenges: Although the general mechanisms do exist for OTA updates, and though it is a common practice for CE products, this is a comparatively new requirement for the automotive domain. Hence, from the technical point of view, most automotive software so far has not been designed for OTA updates. Besides, "lifecycle management of software in cars is in its early stage" as Broy found in (2006a). Accordingly, in addition to technical advancements, organisation must be achieved.

2.3.5 Variants and Configurations

Henry Ford is credited with the statement "Any customer can have a car painted any colour that he wants so long as it is black". This was at the early 1900s. More than 100 years later, this has fundamentally changed. These days, vehicle buyers of course can choose between the outside colour of their car: they can also choose the interior colour, materials, engine, transmission, wheels, light system, as well as infotainment, navigation, driver assistance systems, connectivity features, etc. Considering that a premium car has about 80 electronic fittings, with respect to customer individualisation and country specifics, just a binary decision yes or no leads to 2^{80} variants (Broy et al., 2007). It is obvious that the number of variants must be restricted. However, the number of variants and configurations still remaining when considering only customer individualisation and country specifics is huge.

Besides, additional variants and configurations originate because of necessary ECU changes during the production period (see Section 2.3.4), which leads to an updated hardware and software version. Thus, this not only leads to a new overall system status for the vehicle series, but also causes mixtures because of necessary replacements of defect ECUs of existing vehicles

for maintenance reasons. Here additional variants that will occur due to “normal” software updates for bug fixing or optimisation (e.g. due to IVI application stores) have not even considered.

An eye-opening observation is mentioned by Broy: He tells of studies showing that up to 50% of the ECUs that are replaced at the garage because of a problem reported by the vehicle owner, are technically error-free (Broy et al., 2007). The new ECU, which typically has also a different (i.e., new) hardware and software version, often fixes the problem. More enhanced diagnosis and repair capabilities could be expected to support the proper failure identification and elimination at the garage (Broy et al., 2007). However, this example shows that the source of the difficulties primarily originates at another level: It is because of “ill-designed or incompatible software” (Broy et al., 2007).

To summarise, the challenge is a comprehensive variants and configuration management that covers all its technical and organisational aspects for automotive software and systems development during the whole lifecycle. Considering the envisaged future vehicular functionalities with accelerated innovation cycles, it could be expected that variants management will gain further significance.

2.3.6 Heterogeneity and Mixed Criticality

Within the vehicular context, diverse domains and their intrinsic organisational and technical characteristics, requirements and challenges now come together. Previously, they used to occur elsewhere only in isolation from each other (S. Bauer, 2010; Pretschner et al., 2007). Their integration into a homogeneous overall system is another challenging task for automotive software and system development. Besides the already-named heterogeneity with respect to life and innovation cycles, the functionalities also differ with regard to safety-criticality, leading to mixed critical systems (Burns & Davis, 2013). In this regard, functionalities that used to have no safety relevance may get safety relevant in the vehicular context. Examples could be HMI functionalities of an IVI system, which have functional safety requirements (ISO 26262, 2011) according to an certain Automotive Safety Integrity Level (ASIL) level, e.g., on essential warning messages (Gieraths, 2014).

2.3.7 Cost Model

Due to the harsh environmental conditions for in-vehicular hardware, with wide temperature and humidity ranges, and special requirements on shock resistance and voltage, specialised embedded hardware has to be used. Due to this reason, the automotive hardware is usually more expensive compared to, e.g., to CE hardware.

Moreover, “[t]he automotive industry operates in a highly competitive mass market with strong cost pressure” (Pretschner et al., 2007). This puts the cost of a single unit into focus (unit-based cost model), in particular because automotive is a “business of scale” (Pretschner et al., 2007).

Here, the production costs are more important than the development costs (Broy, 2006b). The possibility of decreasing the production costs of a component by a few Euros (or even cents) proposes the saving of several million Euros considering the number of units per year over the whole production period. In contrast, the increase of development costs of several ten or hundred thousand Euros is less significant.

Considering that the production costs of software are negligible because they are basically the copying of software, the unit-based cost model substantially impacts the software development: It used to be attractive to spend significant efforts (and money) to the optimisation of software functionalities with the aim of reducing the hardware requirements, because this enables the usage of cheaper hardware, which decreases the production costs. Since this software optimisation typically is achieved by less abstraction or generalisation and more hardware-specific optimisations, this has a negative impact on the reusability of software functionalities, cf. Section 2.3.3. However, because the relevance of software regarding functionalities and development costs increases, reusability becomes more and more important, and to enable even faster time-to-market (see Section 2.3.4), it is anticipated that the relation of hardware production to software development costs changes: This means, while the importance of software for the vehicle as well as the efforts and costs for software development significantly increasing, highly-optimised hardware to decrease production costs is becoming less important. Instead, standardised, more powerful hardware is becoming advantageous, since it reduces complex and costly software adaptions and eases reusability of software components, which again reduces costs.

2.3.8 Development Process

The development process of a vehicle is characterised by a high division of labour and its multidisciplinary. As Broy found in (2006a): “The mechanical engineers worked hard for over 100 years to make the various sub-systems in cars in their development and production quite independent. This facilitates independent development and production of the sub-parts […]” But, with the rising amount of electronics and software, the high division of labour nowadays also poses challenges: Since vehicular functionalities are increasingly distributed (see Section 2.3.2), OEMs not only remain the assembler of parts, but now have to integrate them (Broy et al., 2007). Moreover, nowadays OEMs in several situations additionally occur as Tier-2 to the Tier-1, for example, if they provide not only requirements but also HMI models to the Tier-1 that develops the IVI system. The term Tier-n refers to the supply chain distance to the OEM, which is the top of it. Typically, the Tier-n+1 supplies to a Tier-n, and a Tier-1 supplies to the OEM (cf. Fürst & Bunzel, 2015; Sangiovanni-Vincentelli & Di Natale, 2007).

Besides derived necessities for improved technical solutions (e.g., software and system architecture, development tools, modelling techniques), also new development processes and collaboration models between OEM and Tier-1s are necessary. Finally, these challenges (respectively the improvement of them) are strongly interrelated with skills of employees. Broy

found in (2006a): “In a time of only 30 years the amount of software related development activities went from 0 to 30 or even 40 %. If we assume that an engineer works about 35 to 40 years in industry, it is obvious that the companies were not able to gather sufficient competencies quickly enough”.

2.3.9 Conclusion

The previous sections discussed characteristics and challenges of today’s automotive software and system development. These challenges must be addressed on different levels: This includes organisational dimensions such as development processes (e.g., agile approaches) and collaboration models (e.g., between OEMs and Tier-1s). Furthermore, technology improvements can contribute to advancements: for example, new communication technologies and System on Chip (SoC) that support advanced ECUs.

However, the keys to dealing with these challenges and to enable the implementation and integration of heterogeneous functionalities associated with future vehicles (see Section 2.1) are advanced software and system architectures and platforms. Moreover, considering their complex and costly development, the connected and distributed nature of future functionalities and objectives such as reusability, such architectures should be independent from a particular vehicle series and even independent from one OEM as well. Therefore, also the architecture design and methodology are important. They are introduced in the following section.

2.4 Software (and System) Architecture Design and Methodology

In general, systems are built to implement the intended business or functional goals. According to Kossiakoff et al., a system can be defined as follows:

> *“A system is a set off interrelated components working together toward some common objective” (Kossiakoff, Sweet, Seymour, & Biemer, 2011).*

This general definition facilitates its applicability in several contexts, including large-scale distributed software systems or even smaller software components that might be considered as a system.

However, within the automotive context, the term ‘system’ is typically used when considering the whole vehicle. Furthermore, the term system, as used in the automotive domain, has a stronger hardware dependence. This also relates to the V-Model, and V-Model XT (Höhn & Höppner, 2008), which are common development models in automotive: Here, the upper part of the V is related to system, whereas the lower part of the V is related to software. Consequently, within the automotive domain and in the context of embedded devices, a distinction between system and software is established, although both could be considered as a system in the sense of its definition. This distinction is also manifest in the term software-intensive system, to which modern vehicles belong (see Section 2.3).

Nevertheless, since this research focuses on distributed systems, namely the vehicle and its connection to backend server facilities, both aspects of the system (i.e., software and hardware) have to be considered. Although often referring to software (architecture) in the following, it shall be noticed that there might also be impacts on the system (architecture).

The intention of the following sections is to explain the role of software (and system) architecture with respect to the achievement of the challenges, respectively derived objectives. This includes understanding of the problem space and the solutions space, as well as methodology for architecture design and evaluation.

2.4.1 Software and System Architecture

As discussed in Section 2.3.1, software is of central importance for the fulfilment of the objectives of today's vehicles, or – in other words – the software is a fundamental aspect of the solution to the given problem. In this regard the architecture of a system (or software) is the key and foundation "to build systems that satisfy requirements" (Bass, Clements, & Kazman, 2012, p. 64). Or as Medvidovic et al. stated in (Medvidovic & Taylor, 2010): "At the heart of every well-engineered software system is a good software architecture".

There are several definitions of the term (software or system) architecture. Two are considered as particularly appropriate and thus shall be used for this project. The first definition is provided by the Institute of Electrical and Electronics Engineers (IEEE):

> *"[The architecture is the] fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution. (IEEE Computer Society, 2007)*

In addition to others, the statement about the principles or the foundation of an architecture that shall also guide the design and evolution, is considered as important aspect. Another appropriate definition is provided by Bass et al.:

> *"The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both" (Bass et al., 2012, p. 4)*

In contrast to others, both definitions waive the adjectives "early" or "major" about the design decisions being made. Bass et al. argues that depending on the development methodology, design decisions might not be early decisions (Bass et al., 2012, p. 4). Further, they say that at the time when design decisions are being made, the grade of their implications might typically not be known yet (Bass et al., 2012, p. 4). However, Bass et al. does not exclude that design decisions might be early and major decisions – they just do not declare that as necessity (Bass et al., 2012, p. 4). Furthermore, this definition emphasises that the software architecture of a system is not only the "a priori" outcome of the design process but also enables "a posteriori"

reasoning about the system. Due to this fact, architectural analysis provides a means for evaluation of a system.

2.4.2 Problem Space

This project is driven by the automotive domain. The current automotive software and systems development landscape, including, e.g., driving functional domains (see Section 2.1), computational components (see Section 2.2.1), communication technologies (see Section 2.2.2), and domain characteristics and challenges (see Section 2.3). All these aspects have impacts on the problem space.

Requirements engineering is the discipline which generates ("shapes") the problem space for the system to be developed. Requirements can be divided into functional requirements and non-functional requirements (also known as quality attributes). Another perspective on requirements is their relevance for the software (and system) architecture. These subsets of the overall requirements are called Architecturally Significant Requirements (ASR), defined as:

> *"An architecturally significant requirement (ASR) is a requirement that will have a profound effect on the architecture – that is, the architecture might well be dramatically different in the absence of such a requirement." (Bass et al., 2012, p. 291)*

Harrison et al. found: "A particular challenge of quality attributes is that because they tend to be system-wide characteristics, system-wide approaches are needed to satisfy them; these approaches are defined at the system architecture level and not the component level." (N. B. Harrison & Avgeriou, 2010). Hence, non-functional requirements are often ASRs, but they are not necessarily congruent since also functional requirements could be ASRs.

Several other aspects of the automotive context might be immutable, or at least are out of scope for this research project. Herein these are, for example, physical network technologies, certain network protocols, as well as organisational considerations, e.g., division of labour, or product lifecycles. These are transferred to the problem space as constraints, whereby:

> *"A constraint is a design decision with zero degrees of freedom. That is, it's design decision that's already made" (Bass et al., 2012, p. 64)*

Accordingly, a constraint has also a direct counterpart in the solution space that is its design decision.

2.4.3 Solution Space

The solution space "spans" the space for the identification of the solutions to the requirements of the problem space. Within the solution space, functional requirements are mapped to functional attributes and non-functional requirements are mapped to quality attributes (Raja, Iqbal, & Ihsan, 2005). The counterpart of the ASRs in the problem space is the software (and

system) architecture in the solution space. The software architecture is the sum of its architectural design decisions being made, whereas an architectural design decision is:

> *"A description of the set of architectural additions, subtractions and modifications to the software architecture, the rationale, and the design rules, design constraints and additional requirements that (partially) realize one or more requirements on a given architecture." (Jansen & Bosch, 2005)*

Architectural design decisions have impacts on the problem space and the solution space, since they extend the problem space (due to the fact that every design decision typically leads to new requirements) and constrain the solution space (Masak, 2009).

2.4.4 Architecture Design

> *"Design is a process in which a given specification of desired functions and a set of constraints are transformed into a description of an artefact that fulfils the requirements and satisfies the constraints." (Zhang, 2008)*

In this regard, architecture design is the process of transforming the problem space into the solution space.

However, neither requirements nor architectural artefacts are orthogonal to each other. Both, functional and non-functional requirements have interdependencies, and architectural design decisions as well have interdependencies among themselves and with the requirements (Raja et al., 2005). This requires perpetual decisions about architectural trade-offs during the development of the architecture, which are finally trade-offs between ASRs respective to what extent they are fulfilled. For example, increasing the modifiability through reduced coupling, e.g., by means of an additional intermediate layer, might lead to a decreased performance (Bass et al., 2012).

This indicates the challenge and complexity of architecture development. Hence, "designers have been looking for ways to capture and reuse hard-won architectural knowledge" (Bass et al., 2012, p. 203). An emerging approach is the usage of architectural patterns and tactics. It makes use of the fact, even though every software and system development at first sight might be recognized as unique, it usually unveils several similarities to other software systems at a closer observation (with an appropriate abstraction level). This means: Other systems might already contain knowledge about a solution to a similar problem which could be discovered and used. More precisely:

> *"A pattern documents a recurring problem-solution paring within a given context" (Buschmann, Henney, & Schmidt, 2007, p. 4).*
>
> *"The documentation comprises a common structure (i.e. components and connectors) and behaviour, and the clear presentation of its characteristics (i.e. benefits and*

liabilities, respectively satisfied functional and non-functional requirements)" (Bass et al., 2012; Buschmann et al., 2007; N. B. Harrison & Avgeriou, 2010).

Well-known patterns are, for instance, solutions to object-oriented design (Gamma, Helm, Johnson, & Vlissides, 1994), the Service-Oriented Architecture (SOA) pattern or the publish/subscribe pattern (Bass et al., 2012; Coulouris, Dollimore, & Kindberg, 2012). Patterns do also have interdependencies and overlaps, and accordingly a pattern implements several tactics, whereas:

"A tactic is a design decision that aims to improve one specific design concern of a quality attribute" (N. B. Harrison & Avgeriou, 2010)

For example: A tactic to improve the quality attribute "performance" is to increase available resources (Bass et al., 2012).

In this regard, both patterns and tactics, facilitate the reuse of "proven good design structures" (Bass et al., 2012) which can support the development of a system architecture that satisfies the envisaged functional and non-functional requirements. Hence, patterns and tactics shape the architecture (or the solution space), although tactics usually do this at a smaller scale.

Jansen and Bosch stated in (2005): "Although the achievements of software architecture are formidable, some problems still remain. The complexity, high costs of change, and design erosion are some of the fundamental problems of software architecture. We believe these problems are partially due to knowledge vaporization." (Jansen & Bosch, 2005). In this regard, architectural design decisions, trade-offs and principles need to be documented in an appropriate way to facilitate architectural evolution "in harmony with the existing design decisions" (Jansen & Bosch, 2005). According to the IEEE Std 1471-2000, "[…] an [architectural description] shall include the rationale for the architectural concepts selected. An [architectural description] should provide evidence of the consideration of alternative architectural concepts and the rationale for the choices made. […]" (IEEE Standards Association, 2000). This is also necessary to master architectural erosion (Strasser et al., 2014; Vogel et al., 2009, p. 410).

2.4.5 Platform-Based Design

After discussing the principles of architecture design, a systematic methodology is needed. Basically, there are two possible approaches for architecture design and software development methodology (SEBoK authors, 2015):

- Top-down approach: It starts with the problem (i.e., the functionality to be realised) that is considered at its highest abstraction level. From this, the problem is decomposed one by one into smaller problems (or requirements) until the problem is holistically specified. Afterwards, single building blocks that address sub-problems can be built and integrated, which in their entirety solve the initial overall problem.

- Bottom-up approach: It starts with the consideration of the available building blocks (i.e., the sub-solution) which are combined, expanded and evolved until they meet the overall requirements (i.e., the problem).

Strictly selecting one of these two opposing approaches as architecture design methodology is typically not appropriate, considering their individual drawbacks: Purely top-down approaches provide maximum flexibility to finding and building the solutions. Nevertheless, this flexibility may lead to "unimplementable requirements" (Horowitz et al., 2003). Particularly within the context of software-intensive systems with increasing functional complexity, this methodology is often too expensive with respect to time-to-market and costs. Moreover, with respect to connected vehicles or ITS, which shall be interoperable and cooperative, individual solutions that are a natural result of a strictly top-down approach, are a contradiction. In contrast, a purely "bottom-up design [approach] often results in a mess" (Horowitz et al., 2003). This statement of Horowitz et al. emphasises the fact that bottom-up approaches often lack a structured process and due to the absence of adequately decomposed problem statements or requirements, a clear view on the necessary development direction is missing.

The Platform-Based Design approach (PBD) is a "middle-out" or "meet-in-the-middle" approach, intending to combine the advantageous of top-down and bottom-up approaches, while at the same time minimising their individual drawbacks. In the context of PBD a platform is defined as follows:

> *"[A] platform is designed to be a library of components that can be assembled to generate a design at that level of abstraction. This library not only contains computational blocks that carry out the appropriate computation but also communication components that are used to interconnect the computational components" (Natale & Sangiovanni-Vincentelli, 2010).*

This generic definition of a platform facilitates its utilisation on several abstractions levels of an embedded system. In this regard, this model can be instantiated for hardware platforms, network platforms, and software platforms. Furthermore, each platform typically can be refined, e.g., a network platform itself can be considered as a stack of several platforms, such as the different layers of the Open Systems Interconnection (OSI) reference model. Similarly, the software platform can be refined to several layers and respective interfaces or Application Programming Interfaces (APIs), such as a Portable Operating System Interface (POSIX) provided by the Operating System (OS), a middleware layer, and an application layer.

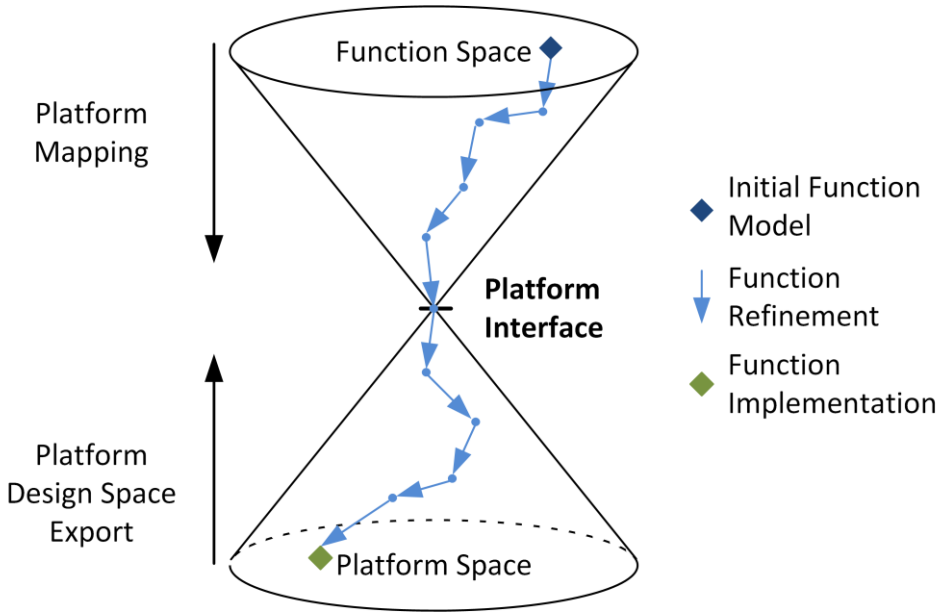

Figure 2.4: The platform-based design approach (Natale & Sangiovanni-Vincentelli, 2010)

Figure 2.4 illustrates that the function design space with the PBD approach is restricted by and clearly specified through the respective platform interface. At such interface, the higher level of abstraction is mapped to a lower level of abstraction. Both, the constraint design space at the interface and the exact mapping is illustrated by the narrow tip of the two cones. This illustration also indicates that interfaces are typically designed to hide details (principle of information hiding), hence the actual exchange between both layers of abstraction is also narrowed/limited by design-choice. The actual mapping process from the initial function model to the function implementation is performed by several refinements (i.e., design decisions) that reduce the available design space to zero, which is the mapping of the functionality onto the platform (interface), or in other words, the implementation of the functionality by use of the platform's interface capabilities provided. This illustrates that a platform constraints the overall available solution space which facilitates the transition from federated to integrated solutions, as well as interoperability and shorter time-to-market (Marwedel, 2010; Natale & Sangiovanni-Vincentelli, 2010). Besides, a "[…] platform effectively decouples [e.g.] the application development process (the upper triangle) from the architecture [(i.e., platform) development or] implementation process (the lower triangle)" (Sangiovanni-Vincentelli, 2002). In this regard, PBD also contributes to maintainability and reusability of both platforms and (application layer) functionalities.

The PBD approach has recently gained importance. For example, Gajski et al. found that "[most] designers today use some kind of meet-in-the-middle methodology […] in order to take advantage of the benefits of both bottom-up and top-down methodologies" (Gajski, Abdi, Gerstlauer, & Schirner, 2009, p. 411). As discussed in Section 2.5.2, the principles of the PBD approach are already being applied in the automotive domain by the AUTOSAR development methodology (Natale & Sangiovanni-Vincentelli, 2010). Last but not least, PBD is an

advantageous approach for the development of embedded systems and network platforms (Carloni, De Bernardinis, Pinello, Sangiovanni-Vincentelli, & Sgroi, 2005).

2.4.6 Architecture Analysis

The previously described relationship between the software and system architecture, ASRs, quality attributes, patterns, and tactics also generates means for "a posteriori" architecture analysis. According to Bass et al., "architecture not only imbues systems with qualities, but it does so in a predicable way" (Bass et al., 2012, p. 28). This idea of utilising high level architectural design descriptions to predict the quality of a software-intensive system basically goes back to Parnas's work "On the Criteria To Be Used in Decomposing Systems into Modules" in (1972).

In general, the detection of certain patterns within an architecture enables predictions about the functionalities and quality attributes of the software architecture. Further, modifications of patterns, their specific combination, additions or subtractions can be used to identify tactics that allow inference about design and quality attribute trade-offs (ISO/IEC/IEEE, 2011; Zhu, Babar, & Jeffery, 2004).

Furthermore, scenarios can be used to facilitate architecture analysis early in the design process (Ionita, Hammer, & Obbink, 2002; Kazman, Abowd, Bass, & Clements, 1996). One example is the Architectural Trade-off Analysis Method (ATAM) (Kazman, Klein, & Clements, 2000). The architectural analyses, performed in this research, based on the ideas related to scenario-based architecture analysis and ATAM.

2.4.7 Conclusion

Being aware of the fundamentals of software (and system) architecture design is the foundation for understanding the methodology for the development of the concept and proposed solutions and enhancements that are discussed in the following chapters.

The software and system architecture of a system is essential with regard to the capability of the system to meet functional and non-functional requirements. These requirements together with constraints are shaping the problem space. A subset of these requirements are architecturally significant requirements which have significant impact on the architecture respectively which should be addressed by it. The solution space contains the solutions to the problem space including the software (and system) architecture which is the sum of the architecture design decisions being made. However, considering that typically each architectural design decision results in new requirements, it extends the problem space and constraints the solution space. This again indicates the iterative nature of architecture development (see Section 1.3).

Architecture design is the process where the problem space (i.e., requirements and constraints) is transformed into the solution space (i.e., software and system architecture and further solution artefacts). Architecture design is a challenging task, since architectural design decisions typically have interdependencies which require trade-offs. Architecture principles, patterns, and tactics provide means to document and reuse solutions to given problems and thus contribute to the development of adequate architectures. Not least, these relationships, e.g., between ASRs, architecture, patterns, and tactics, facilitate as well the "a posteriori" architecture analysis. The consideration of architectural design decisions and trade-offs enable the prediction of system qualities and hence the analysis of architecture suitability.

Methodologies such as the Platform-Based Design approach support the architecture design. Instead of a purely top-down or bottom-up approach it uses a "meet-in-the-middle" approach which aims to combine their advantages. As discussed in Section 2.4.5, PBD is particularly suitable for the development of embedded systems and network platforms which is the reason why it is used in this project.

2.5 Related Work

The Section 2.3 emphasised the relevance of enhanced software and system architectures for the development of future connected vehicles. This section introduces related work with regard to common architectures or basic approaches that contribute to the integration of functionalities at different locations in the automotive landscape (cf. Figure 2.3).

2.5.1 Basic Approaches to Improve Integration of Heterogeneous Functionalities

The "traditional" and most obvious approach for integrating new functions into vehicles is to physically integrate (to install) the related source-code into the ECUs or head-unit. Several approaches exist that provide improvements that support the integration of new functionalities in continuation of this "traditional approach".

Knirsch et al. are proposing the utilisation of multicore hardware for structuring of heterogeneous IVI software in (2010). An appropriate structuring and defined core bindings of selected operating system (OS) tasks are proposed to be used for resource management to achieve a deterministic behaviour, even on high load situations on multicore hardware (Knirsch, Schnarz, & Wietzke, 2012; Knirsch, Wietzke, Moore, & Dowland, 2011). The structuring is performed above a single (POSIX-conformant) operating system.

Vergata et al. recommend the use of a Virtual Machine Monitor (VMM) (also known as Hypervisor) on top of modern head-unit hardware (i.e., System on Chip (SoC)) with virtualisation support (Vergata, Knirsch, & Wietzke, 2012; Vergata, Wietzke, Schütte, & Dowland, 2010). The necessary structuring of the software system is enabled, and the integration is performed by use of Virtual Machines (VMs) with different Guest Operating

Systems (GOSs). In the future, even Asymmetric Multiprocessing (AMP) system approaches are discussed, where no indirections are introduced through a VMM apart from a few individual hardware modules which still have to be shared across OSs (Wietzke, 2012).

The named approaches provide fundamental building blocks to improve primarily the integration of heterogeneous functionalities on head-units. Nevertheless, their transferred to other ECUs is possible, and the utilisation of these approaches in higher-level architectures might be meaningful.

2.5.2 AUTOSAR

Vehicles are physically assembled from individual parts which are delivered from different suppliers, producing these parts according to the specified requirements. This method and division of labour has been applied to ECU development including software and hardware. As a result, the entire software used to be hardware-dependent and supplier-specific or even series-specific. From the supplier perspective, this makes reuse across different vendors, series as well as hardware difficult, which hampers the innovativeness and leads to high development efforts and costs. From the OEM perspective, this also negatively influences reusability of features across vehicle generations and series, and it complicates variant management, which, as previously summarised, is time-consuming and costly.

To overcome these issues, leading OEMs and supplier[7] founded the AUTOSAR (AUTomotive Open System ARchitecture) consortium in the fall 2003. The identified shortcomings in former automotive software development for ECUs are reflected by the following nine project objectives (AUTOSAR, 2014), which are the work order for all activities:

- AUTOSAR shall support the transferability of software. (RS_PO_00001)
- AUTOSAR shall support the scalability to different vehicle and platform variants. (RS_PO_00002)
- AUTOSAR shall support a broad variety of functional domains. (RS_PO_00003)
- AUTOSAR shall define an open architecture for automotive software. (RS_PO_00004)
- AUTOSAR shall support the development of dependable systems. (RS_PO_00005)
- AUTOSAR shall support sustainable utilization of natural resources. (RS_PO_00006)
- AUTOSAR shall support the collaboration between various partners. (RS_PO_00007)
- AUTOSAR shall standardize basic software functionality of automotive ECUs. (RS_PO_00008)
- AUTOSAR shall support applicable international automotive standards and state-of-the-art technologies. (RS_PO_00009)

[7] Nowadays the core partners are (AUTOSAR, 2013): Bayerische Motoren Werke AG, Robert Bosch GmbH, Continental AG, Daimler AG, Ford Motor Company, General Motors Holding LLC, Peugeot Citroën Automobiles S.A., Toyota Motor Corporation, Volkswagen AG.

Architecture

To achieve these objectives, besides others, AUTOSAR defines a layered software architecture (see Figure 2.5): It introduces three layers on top of the ECU-Hardware. Bottom-up, there is a hardware-oriented Basic Software (BSW) layer, hosting, e.g., Microcontroller Abstraction, ECU Abstraction, and Standard Software such as Operating System, common Services, and Communication. On top of that layer, the AUTOSAR Runtime Environment (RTE) is placed, following a middleware approach. This enables the overlying Software Components (SW-C) within the AUTOSAR software layer, e.g., Application Software Components, Sensor-Actuator Software Components, Parameter Software Components, and Composition Software Components to become/remain hardware independent.

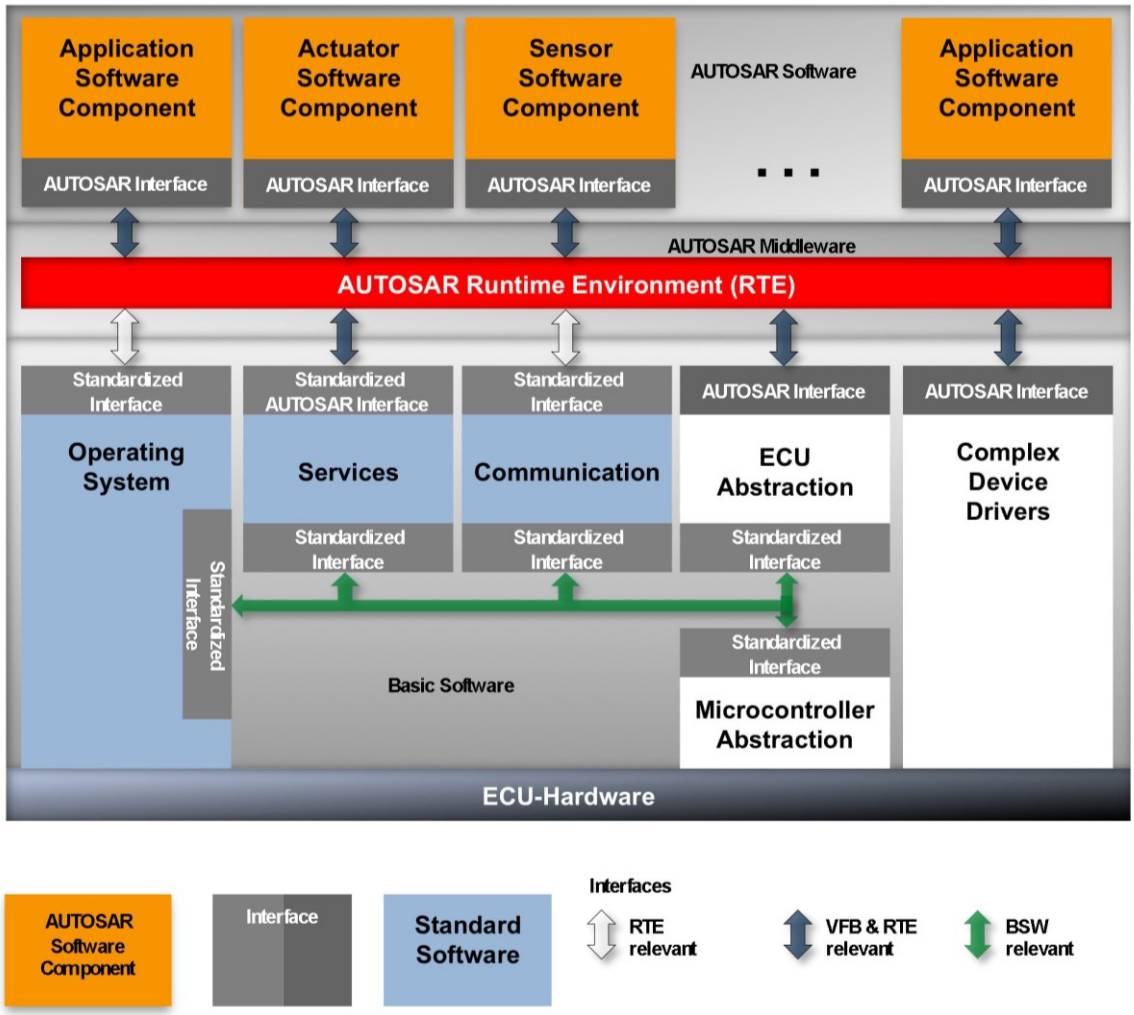

Figure 2.5: The AUTOSAR software architecture (AUTOSAR, 2013)

Further, AUTOSAR defines three types of interfaces (AUTOSAR, 2015a, p. 76): The "Standardized Interface" is a standardized API on programming language level (typically C) which is used between Components of the Basic Software Layer. The second interface type is the "AUTOSAR Interface" which "is independent [of] a specific programming language, ECU or network technology" and defines the information exchanged between SW-C and BSW

modules (AUTOSAR, 2015a, p. 76). Finally, there is a third type named "Standardized AUTOSAR Interface", which is a specific variant from the "AUTOSAR Interface", typically used for standardized AUTOSAR Services, provided by the AUTOSAR Basic Software.

Both "AUTOSAR Interface" and "Standardized AUTOSAR Interface" specify ports, providing different communication capabilities for interworking of components and services: Client-server, Sender-receiver, Parameter Interface, Non-volatile Data Interface, Trigger Interface, and Mode Switch Interface. Each port is available as a Provider Port, a Receiver Port or a Provider and Receiver Port. These ports, together with the RTE middleware layer facilitate abstraction of SW-C from specific communication technologies and hardware (see Section 2.2). This also hides ECU boundaries, which relates to the requirements of transferability of software (Fürst & Bunzel, 2015).

Besides syntactical specification of the ports, data types and semantics must also be specified for SW-C interworking. AUTOSAR supports these formal specification through the appropriate templates (AUTOSAR, 2015b). However, since these templates still require coordination during the development process, the AUTOSAR consortium has begun to collaboratively identify common software functionalities (or SW-C), for which they provide the full specification of the external interfaces. Currently, these extended specifications exist for several SW-C of the following domains: Body and Comfort, Powertrain, Chassis, Occupant and Pedestrian Safety, and HMI, Multimedia and Telematics. The fact that suppliers do not necessarily sell an ECU any more but might sell SW-C which are installed on other ECUs is also considered by the decomposition of functionalities into SW-C. The extended specification further reduces development efforts through increased reusability, interworking, and exchangeability of components from different suppliers.

Methodology

Besides the software architecture and interfaces, AUTOSAR specifies the development methodology, called AUTOSAR methodology. It is particularly tailored to specifics and requirements of the automotive software development process, e.g., high division of labour, necessary cooperation between all parties during all stages of the development process (see Section 2.3.8), and variants management (Fürst & Bunzel, 2015). A central concept and tool during the functional specification of the system is the Virtual Functional Bus (VFB). It facilitates the hardware-independent modelling of the functional interaction between SW-C up to a model of the full system. During the following development process, this functional architecture is mapped step-by-step on the real technical architecture (e.g., ECUs, and communication technologies and topologies) (Fürst & Bunzel, 2015). This is supported by various AUTOSAR tools using the formal specifications for code generation and configuration, e.g., of the RTE and BSW (Fürst & Bunzel, 2015). Since the application-layer logic, located within Application SW-C, is decoupled from the hardware-dependent AUTOSAR Basic

Software through the utilisation of the VFB or RTE middleware, the AUTOSAR development methodology shares many principles with the PBD (Natale & Sangiovanni-Vincentelli, 2010).

AUTOSAR is an emerging approach for automotive software development, addressing many existing challenges (see Section 2.3) with respect to in-vehicle software on ECUs. It shows, that open architectures and the collaboration headline "contribute on standardization, compete on implementation" do have the potential to be advantageous not only from the software architecture point of view, but also from the organizational perspective. AUTOSAR currently focuses on the classical functional domains related to ECU software development. IVI, telematics, and in particular ITS are currently just at their beginnings or still out of scope. One reason might be that AUTOSAR typically addresses functionalities that remain static during the lifetime of the vehicle, despite software updates which might be necessary for bug fixing issues and not primarily for additional features. Hence, variants management and configuration capabilities of AUTOSAR relate to the development process of the vehicular software and ECUs, and not on aftermarket user-dependent individualisation. AUTOSAR has been started to integrate Ethernet and the "Scalable Service-Oriented Middleware over IP" (SOME/IP) middleware (Völker, 2013), but its focus still remains on vehicle-internal connectivity and vehicle-internal distribution of functionalities.

2.5.3 GENIVI

Like AUTOSAR for functionalities related to standard ECUs, the GENIVI (Geneva In-Vehicle Infotainment) alliance standardised the eponymous reference platform for IVI systems (GENIVI, 2013). It started out with 10 founding members: until today 11 OEMs, 19 Tier-1s and many more other companies have joined the GENIVI alliance (GENIVI, 2013).

The GENIVI reference platform specifies a standardised and reusable Linux-based middleware for the implementation of IVI applications (Holle, Groll, Ruland, Cankaya, & Wolf, 2011; Smethurst, 2010). The business logic and HMI remain the individual responsibility of the developing company to enable brand-specific user experience.

2.5.4 OEM Vehicle-to-Backend Platforms

Several OEMs have started to provide connected services to their vehicles by use of own vehicle-to-backend platforms (Johanning & Mildner, 2015; Picone et al., 2015). They are the technical foundation for products, such as Mercedes COMAND online (COMAND Online, 2016), BMW ConnectedDrive (BMW ConnectedDrive, 2016), and Audi connect (Audi AG, 2016a). While the functionalities are to a certain extend similar and even some applications that are provided by some of the named products, which emerged in the CE domain, the enabling platforms are completely different: The current OEM vehicle-to-backend platforms and architectures are proprietary developments. This is also the reason, why technical details are hardly publicly available.

So far, only few data exchange formats are standardised between some companies. Examples are the Navigation Data Standard (NDS), which is a standard for the storage of navigation data including incremental updates (Winner, Hakuli, Lotz, & Singer, 2015). Another example is the Advanced Driver Assistance Systems Interface Specification (ADASIS) standard, that is currently available in version 3 (ADASIS v3). It provides a standardised data format for the realisation of an electronic Horizon (eHorizon) (Siebenpfeiffer, 2014; Winner et al., 2015). They might be integrated in OEM vehicle-to-backend architectures and solutions, but the platforms so far remain proprietary.

2.6 Problem Statement

The "traditional approach" of adding functionalities to vehicles, is the physical integration (i.e., installation) of the source code into the ECUs or head-unit of the vehicle. Assuming that the relating challenges (see Section 2.3) to software and system architectures could be largely solved, which enables the integration of all envisaged functionalities in continuation of the traditional way, other limitations become visible: As described in Section 2.3.7, the unit-based cost model has so far led to ECUs and head-units that already operate to their full capacity at market launch. Accordingly, if the software architectures will no longer be the limiting factor, it might be the hardware. The reason is that new or enhanced functionalities, that will become integrated over time in general increase hardware requirements, such as disc space, memory, CPU speed, or necessary (in-vehicle) network bandwidths. Even if future ECUs might be launched overprovisioned with respect to their functionalities at delivery of the vehicle, it could be expected that they will reach their performance limits earlier than the vehicle reaches its (mechanical) end of life. It could be anticipated that the situation will become similar to the personal computing and CE products, where hardware upgrades or replacements become necessary from time to time although the products are not defect, just because their performance has become insufficient. Hence, continuing the "traditional approach" of integrating new automotive functionalities through truly installing them into the vehicular ECUs and head-unit might additionally require hardware upgrade or replacement strategies.

Finally, that "traditional approach" might be inefficient from an optimised overall system point of view: Many of the future functionalities related to connected vehicles include the data exchange with other entities, e.g., OEM or 3rd party servers, or other vehicles. Integrating them individually and traditionally means that each connected entity or functionality has its counterpart inside the vehicle, i.e., an installed client application, which provides or consumes the data to be exchanged. This makes the vehicle the hub for function and data integration. Besides the previously discussed considerations regarding computational requirements, this also has implications on the network requirements for vehicle-external connectivity (see Section 2.2.2). It could be assumed that in general each new or enhanced vehicular functionality that includes individual communication with external entities increases the overall bandwidth requirements of the car.

Considering just a single vehicle, this might increase the bandwidth requirements[8] to such an extent that they cannot be fulfilled by the network technology at all or at least that the required bandwidth could only be delivered in a smaller part of the coverage area, or only if there are few other vehicles in the same area, etc. This approach extends the increasing requirements to vehicle-internal computational capabilities to the external network capabilities, raising the same necessities: A strategy for the replacement of network hardware during vehicular lifetime might become necessary. Besides network technology constraints, a closer look to the data transferred might unveil that this approach of an individual client application for each functionality is inefficient. It could be expected that several connected functionalities are based on the same vehicular data, particularly its position, speed, remaining fuel range etc. Accordingly, redundant data transmissions are very likely.

However, not only related to a single vehicle optimisation, but especially related to an overall system consideration, the vehicle as individual and main hub for integration of functionalities might not be the optimal solution. Considering ITS scenarios, with distributed use cases across vehicles, vendors and domains, a parent entity as integration hub, such as an OEM server or 3rd party server, might be more appropriate and could decrease bandwidth requirements of the whole vehicle fleet.

As an alternative approach, this research proposes to integrate an increased number of functionalities on backend server facilities, making the OEM server the main integration hub. Such utilisation of vehicle-to-backend platforms as an equal part of the automotive software and system landscape is proposed to enable improved deployment scenarios concerning an overall optimisation in terms of computational, memory, and network capabilities of the future connected vehicle. This is expected to mitigate the vehicle-internal impacts of the introduction of new functionalities during the vehicle's lifetime.

However, the current situation of vehicle-to-backend platforms is characterised by proprietary, vendor-specific solutions (see Section 2.5.4). Indeed, besides standardisation of a limited number of protocols such as NDS (NDS, 2017) and ADASIS v3 (ADASIS, 2017), the situation with today's vehicle-to-backend platforms is similar to those of the in-vehicle ECU architecture before AUTOSAR. This is considered as a significant drawback, because of the following reasons: On one hand, the vehicle-external communication and interconnection of vehicles with backend server facilities introduces numerous new challenges which are not originally the core competencies of the OEMs, such as wireless access network specifics. On the other hand, in particular the functionalities associated with connected vehicles are requesting interworking across vehicle-series, vehicle-OEMs, and even beyond the automotive domain. In this regard,

[8] Network bandwidth is used here only as one example for network requirements and constraints. The considerations could also be extended to other parameters, such as latency, jitter, or error rate. It could be assumed that more communicating entities (i.e., vehicles) besides the bandwidth requirements also have negative implications on those other network performance indicators.

proprietary and vendor-specific vehicle-to-backend platforms are considered as contradictions. To address this named gap, this research focus on the development of a novel automotive vehicle-to-backend platform architecture, aiming at universality and interoperability.

2.7 Summary

The rise of electronics and particularly software together with communication technologies are the basis for the sophisticated and innovative functionalities of modern vehicles. Especially the functional domains In-Vehicle Infotainment and Advanced Driver Assistance Systems have driven this development in the past, and they are expected to also drive it in the future towards the visions of Highly Automated Driving and Intelligent Transportation Systems.

Historically grown, the automotive software and system landscape today consists of up to 100 Electronic Control Units which are very diverse, since they are used to be dedicated for specific functions. Even smartphones and other CE devices can be considered as part of the automotive landscape, since in many cases they are integrated beyond hands-free telephony to enable the usage of their apps. The ECUs are inter-connected by means of various in-vehicle communication technologies, and connected vehicles in addition have external communication capabilities, e.g., by means of wireless cellular network technologies. Thus, backend servers too are becoming a part of the distributed automotive software and system landscape.

The composition of these different functional domains, together with the individual characteristics of the automotive domain, make the automotive environment unique for software and system development. The task of integrating the ever-increasing range of functionalities with heterogeneous or even contradictory requirements into a homogenous overall system has generated various challenges. Over and above other obstacles, engineers have to deal with different life cycles and innovation cycles, mixed criticality, many variants and configurations.

Advanced software and system architectures are the key to dealing with these challenges. Therefore, fundamentals of software (and system) architecture design and the Platform-Based Design methodology are introduced, which are the methodical framework for the concept development and architecture design and analysis. With respect to the development efforts, reusability of functionalities, and interoperability, such architectures and platforms should be universal and OEM-independent. This knowledge motivated the development of the AUTOSAR standard for functionalities of in-vehicle ECUs and the standardisation activities of the GENIVI alliance for the IVI domain. However, in contrast to these developments within in-vehicle automotive software and system landscape, OEMs at the moment are developing and utilising proprietary vehicle-to-backend platforms. Considering the objectives of an ITS, this is an even greater contradiction in regard to inter-vehicle, inter-OEM, and inter-domain use cases.

Closing this gap through the development of a common vehicle-to-backend architecture and platform is hence the focus of this research and content of the subsequent chapters. First, the next chapter introduces the concept of a distributed Automotive Service Delivery Platform.

3 The Distributed Automotive Service Delivery Platform Concept

This chapter is intended to describe the concept of a distributed Automotive Service Delivery Platform as the proposed solution to the shortcomings of current vehicle-to-backend platforms (see Section 2.6). A concept can be defined as follows:

> *"[A concept is an] abstraction; a general idea inferred or derived from specific instances." (Oxford Dictionary Online, 2012)*

The task of concept definition can be defined as:

> *"[Concept definition is a] set of core technical activities of systems engineering in which the problem space and the needs of the stakeholders are closely examined. This consists of analysis of the problem space and definition of stakeholder needs for required services within it." (SEBoK authors, 2015)*

Based on these definitions, this chapter starts with the presentation of the concept and principles with the intention to scope and constrain the problem and solution space by means of fundamental design decisions such as service-orientation. Afterwards, the problem space is further examined through selected scenarios that are described in more detail, and the elicitation of criteria for the distribution of functionalities between the vehicle and the backend. Everything leads to the consideration of different viewpoints on the distributed Automotive Service Delivery Platform which are ultimately used to identify fundamental requirements (i.e. ASRs, see Section 2.4.2) to enable the realisation of the concept.

3.1 Concept

The visions related to connected vehicles not only increase the existing challenges for automotive software and system development – the connectivity also enables new ways of addressing them.

The fundamental of the concept, proposed by this research is to utilise this Internet connection together with backend server capabilities to widen the integration space for automotive functionalities beyond the vehicle itself. The basic architecture of such distributed automotive

software platform is illustrated in Figure 3.1. Here, the OEM On Board Unit (OBU)[9] is the central component of the vehicle in regard to its Internet connection. It integrates the vehicle-external connectivity while functionally connecting and abstracting the displays, sensors, actuators, and other internal components to the outside. The OEM On Board Unit is connected via wireless cellular networks with an OEM server[10], also referred to as backend, located within the Internet domain or "Cloud". The OEM server in turn might connect third party servers for service provision and consumption.

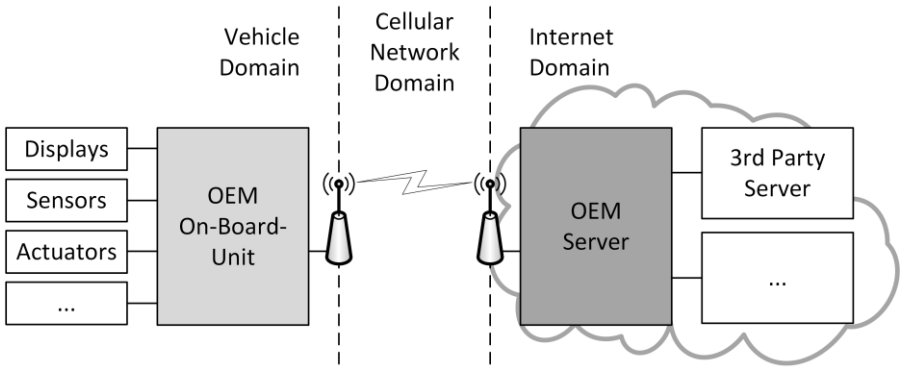

Figure 3.1: Basic architecture of the distributed automotive software platform

3.1.1 Principles

As counter draft to the "traditional approach" of automotive software development, where most functionalities are implemented inside the vehicle, this concept aims to achieve the following three basic principles:

Integration of Functionalities at the OEM Server

The first principle is to enable the integration of automotive applications outside the vehicle on the OEM server. It is envisaged that the OEM server will become an intrinsic and equal part of the automotive software and systems landscape. This should empower software architects to freely decide about an implementation within the vehicle or within the OEM server.

Offloading of Functionalities to the OEM Server

Secondly, functionalities that formerly used to be integrated into (an ECU of) the vehicle might be offloaded to the OEM server. This relates to the development-time of new vehicles, where the overall functionalities have to be distributed across the software and systems landscape

[9] Within the final technical architecture, the functionalities associated to the OBU might be distributed across several vehicle-internal components, including ECUs or the head-unit, while a dedicated Vehicle Communication Gateway may be used for external communication.

[10] The term OEM server means that the OEM in addition to its in-vehicle components (ECUs, head-unit, etc.) can also use a server, where functionalities can be implemented. Even if this OEM server is later operated by another provider or supplier, it shall be at the OEM's disposal regarding system and software development and considerations of functional distribution between the vehicle and the server.

including the OEM server. Furthermore, offloading of functionalities to the OEM server might be an interesting strategy for the integration of functionalities during the lifetime of the vehicle, e.g., if the upgraded functionality now exceeds the computational capabilities of the vehicle ECU.

OEM Server as Main Hub

Lastly, the OEM server shall be the main hub for integration and interworking with vehicle-external entities, particularly 3rd party servers. Although the direct connection of 3rd party servers with the OEM OBU in selected cases might be possible or meaningful, it is not generally envisaged here. The reason is that "the clash" of different domains (e.g., automotive and CE), should primarily occur at the OEM server. It is anticipated that an OEM server as mediator eases the handling of heterogeneity (e.g., regarding requirements, safety criticality, and lifecycles) compared to configurations, where the main hub is the vehicle.

3.1.2 Enabling Architecture: A Distributed Automotive Service Delivery Platform

As previously highlighted (see Sections 2.6 and 2.4.1), the software and system architecture is the key to enabling the concepts and functionalities related to the future connected vehicles. For this reason, the enabling architecture for our concept is the focus of this research (see Section 1.1).

In regard to the Platform-Based Design methodology (see Section 2.4.5), a platform should be defined that enables the previously described principles. This platform is referred to as a distributed Automotive Service Delivery Platform (ASDP), enabling the delivery of distributed automotive functionalities to the entities within a connected automotive environment. It emphasises the future role of a connected vehicle as part of a connected world, where the vehicle consumes services or data from other connected entities and at the same time provides services or data to other network participants. Thereby, a service can generally be defined as follows:

> *"A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description." (OASIS, 2006)*

Emphasising the usage of services in the context of Internet connectivity, the term "web service" is commonly used, which can be defined as:

> *According to the W3C, a web service is "a software system designed to support interoperable machine-to-machine interaction over a network" (W3C).*

However, in the following the term service is always referred to, whenever a sharp distinction between service and web service is not necessary, although it typically includes its usage over a network.

While service delivery, and thus service-orientation, can be considered as a constraint introduced by this concept, further requirements needs to be elicitated which shall ultimately be addressed by platform capabilities. In advance to this requirements elicitation, in the following, three scenarios are presented in more detail, followed by an assessment of distribution criteria for functionalities/services within the distributed ASDP.

3.2 Scenarios

An appropriate method to start system development is real-life examples of the systems' usage and behaviour, called scenarios (Sommerville, 2010). Compared to use cases, they are more coarse-grained, high-level descriptions of the system, i.e., a scenario usually comprises several use cases. Thus, they are particularly suitable for this research, since the three subsequently introduced exemplary usage scenarios for the ASDP, namely "Extended Floating Car Data", "Vehicle Maintenance / Fleet Management", and "Enhanced Navigation", facilitate the following purposes:

- The scenarios support inference about which subset of the overall vehicular functionalities (see Section 2.1) are particularly relevant for the ASDP. In this regard they contribute to elicitation of ASRs.
- Since the ASDP concept not only should provide a solution to a selected subset of the current automotive functionalities and related challenges, but in particular should be useful for upcoming functionalities, scenarios are useful to describe the greater context of the ASDP.
- They indicate future design decisions according to individual vehicular functionalities and their distribution within a holistic systemic consideration. In this regard, particularly the combined view of the scenarios indicates advantages of the ASDP and optimisation capabilities which facilitates the identification of beneficial ASDP functionalities and qualities.
- Finally, scenarios in general enable the analysis of the appropriateness of (software) architectures (Kazman et al., 1996). Therefore, the introduced scenarios, respectively selected aspects of them, can be used in the following to analyse the capabilities of the proposed architecture.

3.2.1 Extended Floating Car Data

Floating Car Data (FCD) is a comparatively easy and basic functionality, which has been already implemented propriety in some vehicles and smartphone applications (e.g., Google Maps Navigation, (Winner et al., 2012)). It usually works in the background and it is hidden

from the user perception. However, the fact that it is already in the field motivates its adduction as a scenario for this project since its extensive deployment is unquestionably and part of several considerations and projects within the ITS context (see Section 2.1.3).

With FCD vehicles are used as driving traffic information sensors. They periodically report at the very minimum their current location, together with the timestamp to the OEM server. The trigger for these reports may be time-related, distance-related, or a combination of both. This data is transmitted to a traffic management server that aggregates and analyses the data of the different vehicles (respectively smartphone applications). Based on that data, traffic models can be built which enable the detection of traffic jams, calculate the traffic density, and average trip times. So far, these servers are operated proprietarily from OEMs or e.g., vendors of the navigation application. The aggregated traffic information is provided to and used by navigation applications which take it into account for route calculation and optimised guidance.

Moreover, Extended Floating Car Data (XFCD) gathers further vehicle data that might be useful for traffic safety and management (Huber, Lädke, & Ogger, 1999; Quintero et al., 2011). Interesting values are, for instance, the outside temperature, rain sensor measurements, brightness sensor values, hazard light activation. Triggers may vary from low outside temperatures, heavy rain, to a driving control intervention (e.g., of ABS/ESP). The number of values continuously increases – e.g., the traffic sign recognition capabilities of modern vehicles is also interesting and might enable automated updates of map database (Messelodi et al., 2009). The provision of these measurements, together with the position and speed, enables advanced inference regarding the momentary traffic (safety) conditions on a certain road section. The intelligent selection of trigger values (Messelodi et al., 2009; Scheider & Böhm, 2010) is thereby not only necessary for the detection of critical situations but also necessary to trigger the event-based transmission of the relevant data, since a continuous transmission might not be suitable due to bandwidth and privacy considerations. Figure 3.2 illustrates the Extended Floating Car Data scenario as an example.

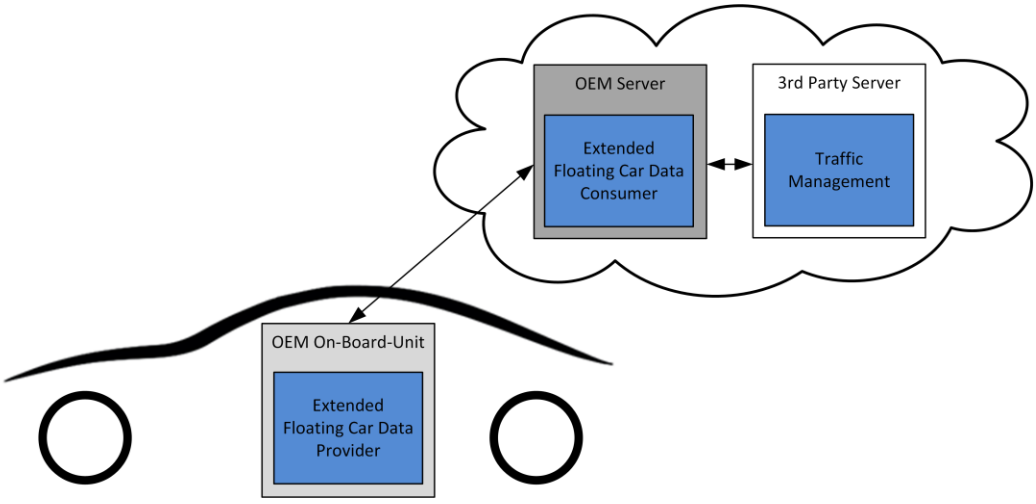

Figure 3.2: Extended Floating Car Data Scenario

3.2.2 Vehicle Maintenance / Fleet Management

Modern vehicles have variable service intervals, depending on their usage, which is monitored over time to estimate when thresholds are exceeded, and service has become necessary. Additionally, various vehicular sensors and check routines continuously monitor component status and individual component failures. In general, these are currently only locally stored using a fault recorder and manually readout at the garage. Connected vehicles enable use cases where relevant data can be submitted to the OEM server periodically or upon error occurrence. The gathered data may be subsequently used to initiate a separate business process of contacting the vehicle owner, discuss necessary service amounts, arrange workshop dates, or, in a wider scope, it might be used for quality management and product improvements. Further use cases are possible such as the monitoring of the remaining fuel range, where the underrun of a defined threshold might trigger other use cases that may propose a low-cost gas station on the route.

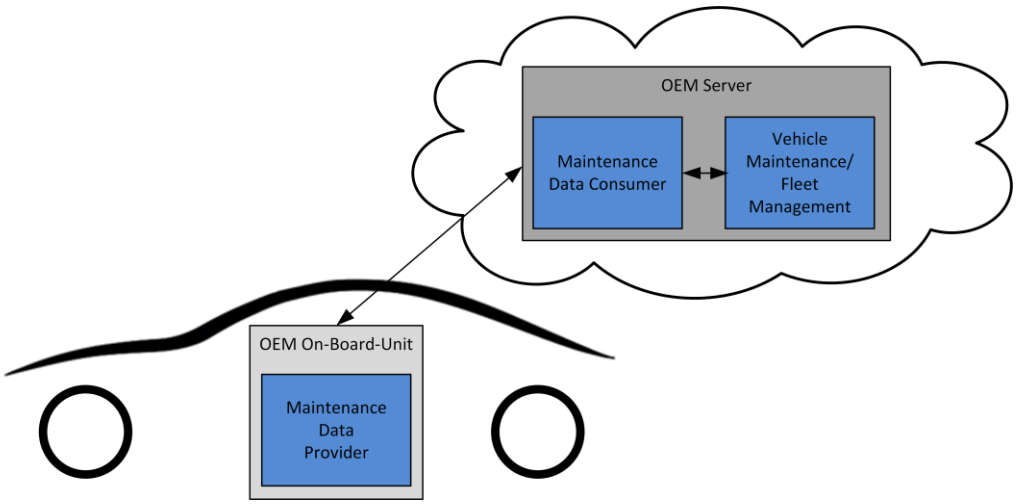

Figure 3.3: Vehicle Maintenance / Fleet Management Scenario

These kinds of functionalities are summarised within the Scenario "Vehicle Maintenance / Fleet Management", which is exemplarily illustrated in Figure 3.3. Similar to the prior scenario, it uses vehicular sensor data that might be transferred to an external entity on event occurrence. Nevertheless, in contrast to the prior scenario, the scope of the data is primarily related to the vehicle owner or driver. This scenario therefore reflects the mixed nature of data provision and functionalities.

3.2.3 Enhanced Navigation

A navigation system in general consists of the following components:

- Positioning (usually through a GPS receiver, possibly correlated with odometry data)
- Map-data, including Points-of-Interest (POI)
- Route-input (e.g., through use of touch-display or voice command)
- Route-calculation

- Route-output/guidance (usually through map display and voice output)

There are basically two alternative approaches for navigation systems or applications (Winner et al., 2012): The first is an on-board navigation system where all components are implemented "offline" within the vehicles' head-unit. The second alternative is an off-board navigation system where most of the components are implemented "online" on remote servers, in particular map-data and route-calculation while clearly positioning, route-input, route-output/guidance to a certain degree must remain within the vehicle. However, offline navigation systems nowadays also use online data, i.e., traffic information to dynamically calculate the best route in dependence of the current traffic situation. Additionally, it is meaningful to equip online navigation systems with limited offline functionalities, since vehicles might be temporally disconnected. In this regard, most modern navigation systems are to some degree hybrid navigation systems.

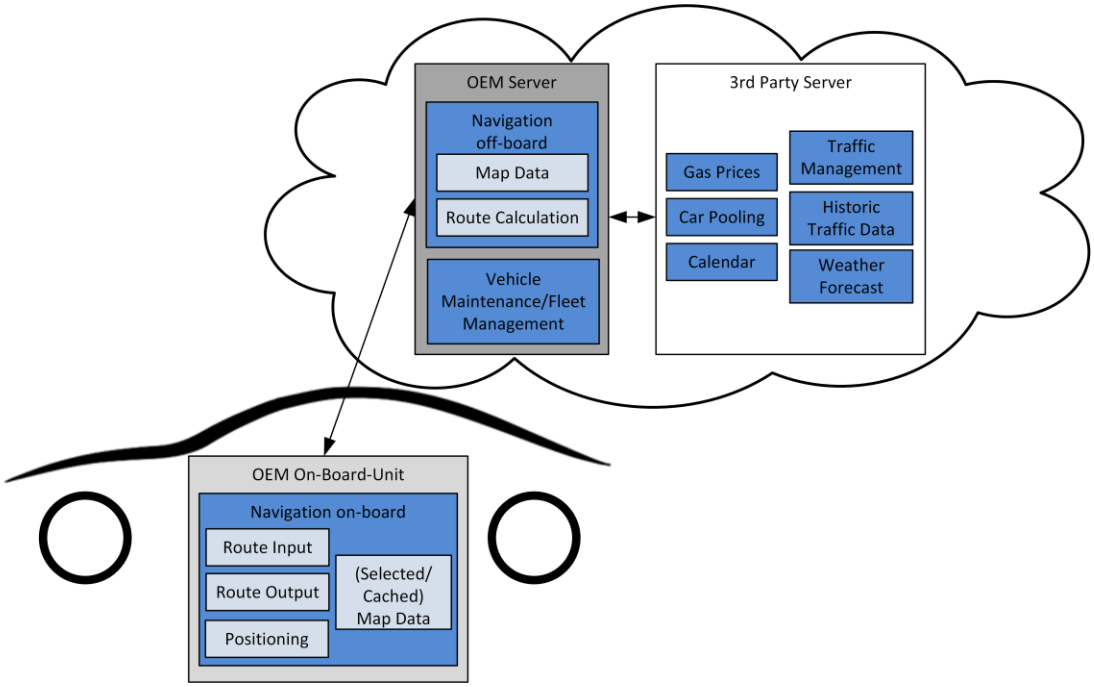

Figure 3.4: Enhanced Navigation Scenario

The scenario "Enhanced Navigation", which is exemplarily illustrated in Figure 3.4, envisages such modern hybrid navigation system, where as many functionalities as possible (respectively appropriate) are implemented off-board/online on server infrastructure due to the following considerations:

Since the vehicle requests the route by providing current position and destination, and – depending on the actual functional split between the server and the vehicle – may receive the route for navigation messages and related map tiles on demand, the server already knows the destination and the planned route. In combination with a connected online calendar, it may even know about upcoming trips. Additionally, with vehicle maintenance data (see Section 3.2.2),

the system also has information about the current fuel level, the distance remaining to destination, and average economy. Based on these data, combined with statistical analysis of historical gas price data regarding cities, day, and time, it can provide optimised suggestions for intermediate refuelling stops. The importance of such functionalities increases especially in regard to electro mobility where charging stops may require more comprehensive planning, with respect to minimized range, extended charging time, and available power plant capabilities. Further aspects might pro-actively influence the route calculation, such as historic traffic data of or current weather conditions on route sections (see Section 3.2.1), or weather forecasts in general. The advantage of an off-board route-calculation component is manifest for all these considerations: Extending the route calculation with additional constraints requires only service advancements on the OEM server and neither vehicular software updates nor considerably increased wireless access network capabilities.

Correspondingly, a car-pooling service can be added just by connecting it to the OEM server. The necessary data (e.g., driving destination, route, and current position) to automatically publish the trip is already available at the server within this “Enhanced Navigation” scenario. If the driver accepts a request for a lift of a car-pooling user, its pick-up and destination addresses can be automatically included as additional waypoints in the route calculation. The request-response workflow and related dialogues which must be presented to the driver within the vehicle might make a few enhancements within the on-board necessary if they could not make use of existing general functionalities e.g., for information about new traffic incidents. Despite this qualification nevertheless this example again shows that within this scenario such enhancements are predominantly only would require enhancements of the server functionalities. Accordingly, if the car-pooling service changes, if it becomes obsolete, or if another one should be integrated, necessary modifications can be limited to the vehicle-external parts of the navigation system.

3.3 Criteria for the Distribution of Automotive Functionalities

As discussed earlier, increasing connectivity together with server capabilities extend the automotive software and system landscape beyond the physical vehicle boundaries. While the increased usage of backend server capabilities in general is a promising approach to address existing challenges, not every automotive functionality in the same way is suitable for implementation at a backend server. Considering the distributed automotive software and system landscape (see Section 2.2), criteria are needed for profound decisions such as whether a certain function should (still) be realised within the vehicle or if it can be implemented externally.

This section identifies such assessment criteria and discusses them by means of selected automotive functionalities. According to the focus of this research on the enabling architectures (see Section 1.1), it is not the aim to make final statements where a certain functionality shall be implemented. The intention is rather to provide indications, which (kind of) applications are

more suitable candidates for realisation at the OEM server respectively offloading to the OEM server, and which are less suitable candidates. For that reason, the criteria are assessed qualitatively, focusing particularly on the relative assessment between the selected applications. This facilitates the identification of representative usage scenarios as well as the derivation of architectural requirements to the ASDP. Moreover, the identified criteria provide a general framework not limited to this research, which can be used to assess further applications in addition to those named here.

A software function can be defined as the mapping of one (or many) input(s) to one (or many) output(s). The data sources for the input can be, for example:

- Sensors, e.g., GPS, rain sensor, temperature, wheel speed, centrifugal force
- Keys/Knobs
- Touch-Inputs
- Audio, e.g., Microphone
- Outputs of other functions

The output of functions can be connected with data sinks such as:

- Actuators
- Audio, e.g., Amplifier/Speakers
- Displays
- Inputs of other functions

A distributed function is the composite of at least two single functions where these functions reside on different components/entities within the automotive software and system landscape (see Section 2.2). These entities might be different ECUs or an in-vehicle component and a backend component.

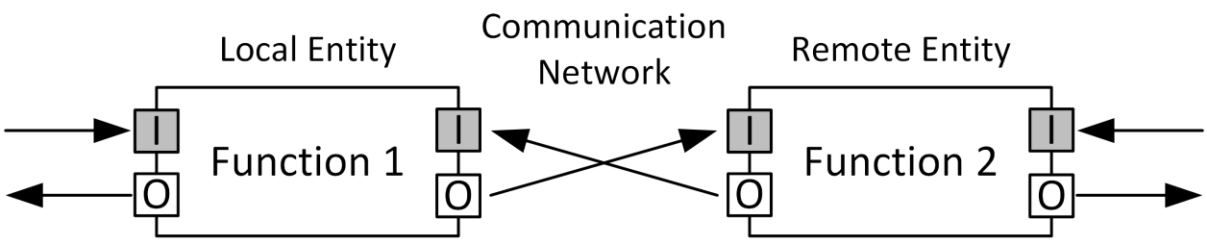

Figure 3.5: Abstract model for distributed functionalities

Figure 3.5 illustrates a general model for a distributed function consisting of a local entity and a remote entity which are connected through a communication medium. Both the local and the remote entity may be data source or sink for the analysed functionality, and both entities provide processing capabilities. This general model facilitates the assessment of functionalities at different abstraction levels. The functionality to be assessed could be high-level, such as the

navigation system in general, which consists of single functional building blocks (e.g., map database and routing component). The model also facilitates the assessment at lower level, such as the consideration of a routing component in detail, which in itself consists of several building blocks.

3.3.1 Application Type

The first general criterion is the application type. Distinction is made between:

- Machine-to-Machine (M2M)
- Human-to-Machine (H2M)
- Machine-to-Human (M2H)

In the first place, M2M applications do not have a Human-Machine-Interface (HMI) and are typically not actively configured by the driver or user. This basically eases their relocation between nodes, such as the offloading from the vehicle to backend server facilities. H2M and M2H applications need appropriate HMIs such as displays and speakers (M2H) or keys (H2M). The distinction between H2M and M2H is made to identify unidirectional involvement of human beings. In case of the realisation of M2H or H2M applications on backend server facilities, appropriate mechanisms and technologies must be used, to implement the necessary user interaction. Examples might be web browsers, audio/video-streaming-techniques or Remote-Desktop-Protocols (RDP) (Y. Lu, Li, & Shen, 2011; Simoens, De Turck, Dhoedt, & Demeester, 2011). Since these technologies are initially independent of concrete applications, they have not been dealt with in depth within the scope of this research.

3.3.2 Data flow

The second criterion is the data flow, evaluating the primary data source and data sink of the application. Here, again, the vehicle as well as the backend are considered in their entirety, since a finer differentiation (e.g., specific actuators, sensors, controllers of the car) is not needed with respect to requirements of the vehicle-to-backend platform or respective network requirements.

Table 3.1 shows the four possible data flows, and for the sake of reference, introduces numbers. The possible data flows differ in terms of the possibilities to deal with limitations from the perspective of a car manufacturer or vehicle-to-backend platform developer. This is indicated by the colour shading of the cells in Table 3.1, with “red” indicating most limited possibilities, yellow indicating moderately limited, and green indicating least limited possibilities. In this regard, data flow 4 (backend-to-backend) is the data flow with the least limitations (green) in terms of the handling of increasing requirements: For example, additional bandwidth between two entities residing in the backend can be provided most easily. This is followed by data flow 2 (vehicle-to-vehicle), which is limited by in-vehicle network capabilities: Since the OEM is in charge of dimensioning the in-vehicle network, the provision of, e.g., additional bandwidth

between two ECUs in general is possible, but this cannot be done during the lifetime of the vehicle without HW upgrades. Furthermore, in-vehicle networks might be more constraining than backend networks (see Section 2.2.2), which is the reason why the possibility of handling increasing requirements is considered as moderately limited (yellow). Most limited possibilities (red) exist with regard to the data flow between vehicle and backend, i.e., the data flows 2 (vehicle-to-backend), and 3 (backend-to-vehicle), since the capabilities of the wireless cellular network must be considered as constraint for OEMs and vehicle-to-backend platform developers: Wireless cellular network technologies are constrained, e.g., with regard to data rates and latencies (see Section 2.2.2), technology upgrades of the wireless cellular network are basically out of scope for OEMs, and possibly would require also HW upgrades of the wireless HW within the vehicles.

		Sink	
	Data flow number	Vehicle	Backend
Source	Vehicle	1	2
	Backend	3	4

Table 3.1: Possible data flows between vehicle and backend and their degree of limitation regarding increased requirements

The data flow of the functionalities is assessed through focussing the actual payload of the application. Possible control messages, e.g., of the transport protocols were left out, as they can be assessed as minor significant here. Some data flows can be transformed and replaced:

- **Replacement 1:** The data flow 1 (vehicle-to-vehicle) can be replaced by data flow 2 (vehicle-to-backend), optionally followed by any number of data flow 4 (backend-to-backend), and data flow 3 (backend-to-vehicle).
- **Replacement 2:** The data flow 4 (backend-to-backend) can be replaced by data flow 3 (backend-to-vehicle), optionally followed by any number of data flow 1 (vehicle-to-vehicle), and data flow 2 (vehicle-to-backend).

While the replacement 2 in practice might be of minor significance, replacement 1 could be beneficial for several functionalities. Particularly, when, e.g., the computational requirements of a functionality are high, the offloading of the computational-intensive part to the backend might be meaningful, given the necessary amount of data transferred to/from the backend (data flows 2 and 3) does not invert the advantage. This corresponds to the principle of functionality offloading of the ASDP concept and among others provides means to deal with increasing computational requirements of a functionality during vehicle lifetime (see Section 3.1.1).

3.3.3 Performance

With respect to the functional split and distribution of functionalities, further criteria to be considered are the performance requirements of the functionality. According to the above-stated method, computational and memory requirements have been qualitatively assessed, aiming to provide a general understanding for the respective function. Here, no distinction is made between different types of processors (e.g., CPU or GPU). The memory assessment focus is on non-volatile mass storage requirements. The main memory requirements during runtime could be another valuable criterion for a more detailed assessment. .Figure 3.6 shows the criteria range as used for the qualitative assessment, starting from “very low” to “very high”, whereas also intermediate shades (e.g., “medium-high”) could be used. Following this, “very low” computational requirements are, e.g., representing those of a climate control, whereas the routing algorithm of a navigation system has “very high” computational requirements. Whether these computational requirements are continuous or sporadic (e.g., only at the start of the navigation) should be considered during the assessment but might also need deeper investigation in a subsequent step. The memory requirements of the climate control are used as an example for “very low”, since it typically has merely a localized mapping of sensor measurements to target values for air distribution, air volume. By contrast, the map database of the navigation system typically requires several gigabytes (“high”), and a personal audio library may require up to one more order of magnitude gigabytes (“very high”).

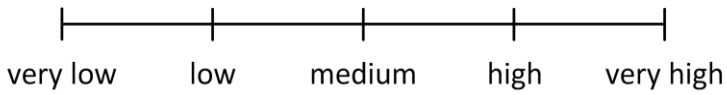

Figure 3.6: Qualitative computational and memory requirements assessment criteria

After the distribution of functionalities is made, the performance of the hosting component such as the head-unit can be dimensioned, considering the assessed utilisation of memory and computational resources. This inter-dependency also exists the other way around: The dimensioning of the hosting component could also impact the final decision about the distribution of functionalities and consequently the performance requirements are an important indicator. In addition, a suitable anticipation of the evolution of the automotive functionalities is needed to facilitate appropriate dimensioning of component hardware, in particular those integrated into the vehicle, including proper architecture and distribution decisions, because: If a future or upgraded functionality raises only a single performance requirement to an extent that cannot be met by the hosting in-vehicle component, this functionality cannot be realized as intended. Although the ASDP concept might provide ways to address such limitations (see Section 3.1.1), these together with other factors motivate the next category of criteria, namely application lifecycle and data update frequency.

3.3.4 Application Lifecycle and Data Update Frequency

The future vehicle is connected with the Internet and should be able to become customised and adapted to the users wishes, similar to Smartphones and related application store concepts nowadays (see Sections 2.1.1 and 2.3.4). As already elaborated on in Section 2.3.4, this has impacts on the life cycle or innovation cycle of automotive functionalities.

Although in general the necessary technologies for OTA updates exist such as the device management protocol from the Open Mobile Alliance (OMA) (OMA TS DM Protocol, 2016), and future vehicles will be equipped with such functionalities (Cacilo et al., 2016), the frequency of application updates are an additional aspect for the design decisions regarding the distribution of automotive applications: The fact that a frequent update of a function could be expected might be a good indicator for the consideration of the offloading of such functionality. In addition to the general efforts to ensure the proper OTA update, it must be considered that updated functionalities often come along with higher performance requirements to the hosting component.

Another related aspect that should be considered separately is the data update frequency of the functionality. In contrast to application updates, data updates do not require technologies such as OTA, but this criterion indicates, e.g., how long a disconnection between the data source and data sink (see Section 3.3.2) is meaningful. Thus, the data update frequency in combination with the consideration of the data source support the assessment about appropriate functional split and distributions of the respective application.

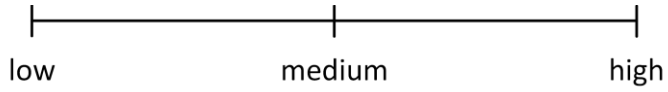

Figure 3.7: Qualitative application lifecycle and data update frequency assessment criteria

Figure 3.7 illustrates the qualitative criteria as used for the assessment of the application lifecycle and data update frequency. A “low” application update frequency as, e.g., with the speedometer indicates that a modification of the application is very rare or unlikely during the whole vehicle lifecycle. In contrast, CE applications such as social networks typically get updated within days or few weeks, which is assessed with “high”. Similarly, the data update frequency is assessed. For the speedometer, this data update frequency is “high”, since it receives the current speed at a very high frequency, i.e., several times per second. In contrast, data update frequency of a personal Online Address Book typically is less than once per day, which is assessed as “low”. Again, also intermediate shades (e.g., “low-medium”) could be used.

3.3.5 Quality of Service

The last group of criteria assesses the Quality of Service (QoS) requirements of selected functions at an abstract level. On the one hand, they can be used to derive concrete requirements towards the wireless access networks (Alasti, Neekzad, Hui, & Vannithamby, 2010; Ekström, 2009; Fuhrmann & Brass, 1994; Oyman, Foerster, Tcha, & Lee, 2010; Parkvall et al., 2008), given that the assessed functionality has a data flow from or to the backend (see Section 3.3.2). On the other hand, the QoS requirements can be used as criteria representing the "costs" of a candidate functional distribution between the vehicle and the backend because: Although advancements of 4G and particularly 5G wireless cellular networks (see Section 2.2.2) might have the capacity of providing the required QoS[11], an overall assessment may come to the conclusion that, e.g., a function relocation from the vehicle towards the backend (or vice versa) is too costly. For example, advantages such as reduction of computational effort within the vehicle (see Section 3.3.3), or considerations regarding lifecycle and simplified updates (see Section 3.3.4) might be disproportional compared to the "costs" in terms of resulting QoS requirements to the wireless access network.

The first criterion used for the qualitative assessment of the QoS requirements is the tolerable delay, see Figure 3.8. An assessment "low" means that the data must be available/transferred in real-time as it is given, e.g., for the speedometer. For the online calendar, data updates should be updated within seconds to few minutes ("medium"); general attributes of the map database (e.g., roadways) may tolerate data update delays of up to hours or even days ("high"). With the same scale (see Figure 3.8), the bandwidth requirements (during usage of the application) are assessed. Transmission of smaller amounts of data, such as for Floating Car Data is assessed as "low". In contrast, voice or video-telephony (Skype, VoIP, Video-Telephony) require "medium-high" bandwidth.

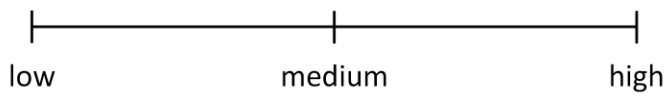

Figure 3.8: Qualitative tolerable delay and bandwidth requirements assessment criteria

Regardless of the QoS capabilities of the network, disconnections cannot be excluded, since it is possible for vehicles to reach areas without network coverage, e.g., underground parking or other (even short) shading situations. Thus, the criteria "Caching possible/reasonable" and "Resubmission possible/reasonable" have been evaluated, by use of the qualitative criteria of Figure 3.9 For real-time applications such as the Speedometer, both criteria are assessed with "no", i.e., neither caching nor resubmission is possible/reasonable. The Personal Audio Library could be cached at the data sink ("yes"), but the resubmission is only "limited" possible/reasonable. This means that the audio data out of the cache is used with real-time

[11] Here, the QoS considerations are limited to the parameters bandwidth and delay.

requirements, and resubmission, e.g., in case a file is not (fully) cached must be completed before the cache runs empty in order to provide an adequate QoS. Floating Car Data is most meaningful when it is up to date. Hence, caching is "limited" possible/reasonable, but resubmissions are generally possible/meaningful ("yes").

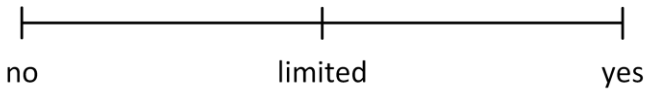

Figure 3.9: Qualitative caching or resubmission possible/reasonable assessment criteria

3.3.6 Assessment of Selected Functionalities

Some automotive functionalities are qualitatively evaluated with respect to the previously identified 12 criteria within five categories (see Table 3.2). These have been selected exemplarily out of the three functional domains (see Section 2.1) with the aim of providing an adequate range of different characteristics: Starting from well-known functions, e.g., speedometer and climate control through to those functionalities, which are currently to some extent still part of research and development, such as hazard warning, road condition warning or traffic-sign-assistant evaluations are detailed. The third part of the investigated applications are applications which are currently available on Smartphones and demanded for future IVI-Systems by many users, for example: Web radio, online calendar and online address book, and social networks, e.g., Facebook and LinkedIn.

As discussed in Section 2.3.2, future automotive functionalities will be highly distributed and interdependent. This makes the individual evaluation difficult, since this poses a "chicken-and-egg problem" with regard to the assessment prior to functional split and distribution. However, in such cases, the evaluation started with an anticipation of a meaningful functional split, aiming towards the most advanced solution, with as many functionalities offloaded to the backend as is reasonable. Hence, an off-board navigation is assumed here as starting point which is not evaluated as a whole but split into several sub-functions.

To start from the extremes, the function "speedometer"[12] is assessed (see Table 3.2, #1): It has a data flow vehicle-to-vehicle (data source and data sink are vehicle), and its performance requirements (computational and memory requirements) are considered as low. Furthermore, the application life cycle probably equals the car life cycle, hence, application update frequency is "low". In contrast, data updates (i.e., adoptions of the current speed) occur at a very high rate of frequency (e.g., several times per second). At the same time, the QoS requirements are extremely high, since delay and caching are thus resubmissions are tolerable. In summary, an

[12] Here, the software part of a classic Speedometer is assumed, by means of a needle that is positioned with a stepper motor, in contrast to modern speedometers, which are widgets on a GUI.

off-loading of the speedometer functionality to backend infrastructure can be estimated as not being beneficial.

In contrast, off-board navigation systems already present their feasibility in current CE products (e.g., Smartphones). Nevertheless, the evaluation is quite ambivalent (see Table 3.2, #17-#24): Such systems consist of several sub-functions, with different data flows, performance requirements, anticipated update frequencies and QoS requirements in case of relocating. Considering the computational and memory requirements, the off-loading of the map database, traffic information, Point of Interest, and routing algorithm is profitable. However, the close interactions with the sub-functions vehicle position and navigation display, which rely on the vehicle as data source and sink, still require a certain performance of the hosting in-vehicle hardware. In practice, hybrid navigation systems, as elaborated in the scenario above (see Section 3.2.3), might be more reasonable and the logical consequence of such assessment.

For Internet-based IVI applications, such as web radio, online video portal, and social networks (see Table 3.2, #5, #7, #15), where the data source is in any case the backend, off-loading offers great potential. They typically have medium up to high update frequencies of data as well as the application themselves. The QoS requirements are medium and can hardly be reduced when they have been realised as fully vehicle-integrated. The streaming of web radio is a typical application, where low-medium delays are acceptable. Accordingly, caching and resubmission is limited possible/reasonable, without reduction of user experience.

Even though M2M applications such as Floating Car Data do not have high performance requirements, they are qualified for most backend-based realisation, especially because they do not have H2M or M2H interfaces (see Table 3.2, #8).

			Data Flow		Performance		Lifecycle / Update Frequency		Quality of Service			
#	Application	Type	Data Source	Data Sink	Computational Requirements	Memory Requirements	Application Update Frequency	Data Update Frequency	Tolerable Delay	Bandwidth Requirements	Caching possible/ reasonable	Resubmission possible/ reasonable
1	**Speedometer**	M2H	Vehicle	Vehicle	low	low	low	high	low	low-medium	no	no
2	**Climate Control**	H2M	Vehicle	Vehicle	very low	low	low	low	medium	low	yes	yes
3	**Driving Mode Selector**	H2M	Vehicle	Vehicle	very low	low	low	low	medium	low	yes	yes
5	**Web radio**	M2H	Backend	Vehicle	low	- [1]	medium	high	low-medium	medium	limited	limited
6	**Personal Audio Library**	M2H	Backend	Vehicle	low	very high	medium	high	low-medium	medium	yes	limited
7	**Online Video Portal**	M2H	Backend	Vehicle	medium-high	- [1]	medium-high	high	low-medium	high	yes	limited
8	**Floating Car Data**	M2M	Vehicle	Backend	low	low	medium	medium-high	medium	low-medium	limited	yes
9	**Maintenance Data**	M2M	Vehicle	Backend	medium	medium	medium	medium	medium-high	low	yes	yes
10	**Hazard Warning**	M2H	Backend	Vehicle	medium	medium	medium	medium	low	low	limited	yes
11	**Road Condition Warning**	M2H	Backend	Vehicle	medium	medium	medium	medium	medium	low	limited	yes
12	**Traffic-Sign-Assistant (map-based)**	M2H	Backend	Vehicle	medium	medium-high	medium	low-medium	low	low	yes	yes
13	**Online Calendar**	M2H, H2M	Backend	Vehicle	low	medium	medium	medium-high	medium	low-medium	yes	yes
14	**Online Address Book**	M2H, H2M	Backend	Vehicle	low	medium	medium	medium-high	medium	low-medium	yes	yes
15	**Social Networks**	M2H, H2M	Backend	Vehicle	high	medium-high	high	high	medium	medium-high	limited	yes
16	**Skype, VoIP, Video-Telephony**	M2H, H2M	Vehicle, Backend	Backend, Vehicle	medium-high	medium	medium	high	low	medium-high	no	no
	Off-Board Navigation											
17	**- Destination, Routing Options Input**	H2M	Vehicle	Backend	low	low	low-medium	- [1]	medium	low	limited	limited
18	**- Vehicle Position**	M2M	Vehicle	Backend	low	low	- [1]	high	low	low	no	no
19	**- Map Database**	M2M	Backend	Backend	- [1]	very high	medium	low-medium	high	medium-high	yes	yes
20	**- Traffic Information**	M2M	Backend	Backend	low-medium	medium	medium	high	medium	low-medium	yes	yes
21	**- Point of Interest**	M2M	Backend	Backend	- [1]	high	medium	medium	medium	medium	yes	yes
22	**- Routing Algorithm**	M2M	Backend	Backend	very high	- [1]	medium	- [1]	medium	- [1]	- [1]	- [1]
23	**- Navigation Display**	M2H	Backend	Vehicle	high	medium	medium	high	low	medium-high	no	no
24	**- Navigation-(Voice-) Command**	M2H	Backend	Vehicle	medium	medium	medium	medium-high	low	medium	no	no

[1] no assessment possible/suitable

Table 3.2: Assessment of criteria for the distribution of selected automotive applications

3.3.7 Discussion

The identified criteria for the distribution of automotive functionalities contribute to adequate decisions about decomposition and distribution of functionalities between the vehicle and backend. In particular, the criteria are helpful to assess the "benefits" (e.g., less computational and memory requirements for in-vehicle components) in contrast to the "costs" (e.g., the QoS requirements towards the wireless cellular networks) of function relocations. In this regard, the identified criteria provide also a framework to assess further applications in addition to the exemplary selected applications here.

Nevertheless, it must be considered that most functionalities have interdependencies between each other (see Section 2.3.2), and with respect to the programmable vehicle, this is expected to further increase (see Section 2.3.1). Accordingly, the positioning of a map database or a routing algorithm of the navigation system cannot be individually assessed but should be considered in combination. In this regard, the actual decision about the distribution of vehicular functionalities must be made at the time of vehicular development and individually for the intended range of functions. Here, besides the criteria named, one must keep in mind that cross-vehicle and OEM software and platform strategies might have an impact, too. However, the assessment of the exemplary selected applications (see Table 3.2) in their entirety enable to identify which kind of applications are particularly suitable for offloading to the backend, which contributes to the development of the distributed ASDP.

The possibility of integrating functionalities within the vehicle or at the backend enable new strategies to deal with contradiction of vehicular life cycles and life time updates of functionalities, or to provide after-sales functional upgrades: Instead of the (hardware) upgrade of vehicle-internal components, some functionalities could be provided as services residing on backend server facilities (cf. Section 3.1.1). This might make the necessity to replacements of vehicle-internal hardware components obsolete, or possibly only communication-related components must be replaced (e.g., to utilise improved wireless cellular network capabilities), or both upgrades become necessary at a later time or less frequent during the vehicular lifetime. The assessment contributes to the identification and clustering of these applications.

Another outcome of such assessment could be different deployment/distribution strategies, even for one vehicle series. These could enable novel directions for different vehicle configurations with respect to one-time or ongoing costs. For example: It might be suitable to build a low-budget configuration with less powerful in-vehicular components. In such configuration, the same number of functionalities could be provided, whereas more functionalities are hosted at the backend server. The possible disadvantage that backend functionalities might not be usable if there is no network coverage, or that their operation might be slower, depending on the available wireless access network, might be an acceptable trade-off with regard to one-time costs at vehicle purchase and potentially higher ongoing costs during vehicle lifetime

To conclude, it must be stated that the identified criteria in fact support the decision of functional distribution between the vehicle and the backend, but that the related decisions have so many inter-dependencies that they, in most cases, cannot be performed in a manner that is universally valid. In this regard, ASDP requirements cannot be directly derived from a single (distributed) application, although the requirements should reflect exactly this fact – that the distribution of most functionalities is variable and depends on the actual configuration.

3.4 Viewpoints and Derived Requirements

After presentation of the basic concepts and principles, as well as scenarios and distribution criteria, requirements to the distributed ASDP have been derived. Hereof, principles, scenario considerations, and criteria contribute to and are reflected within three different viewpoints, namely:

- **Future vehicle-to-backend platforms**, which primarily reflect domain-internal viewpoints of the current automotive engineers.
- **The vehicle as part of an Internet of Things**, which mainly reflect external viewpoints on the connected vehicle.
- **Towards an Automotive Embedded Internet** which is a viewpoint from the telecommunication domain on connected vehicles.

Beginning with these viewpoints, ASRs to the distributed ASDP are derived. The herein defined requirements are not disjoined and not intended as a full specification of the software and system architecture of the platform. Instead, their aim is to describe cornerstones that should guide the selection and development of the software and system architecture, including platform capabilities.

3.4.1 Future Vehicle-to-Backend Platforms

The approach of a distributed ASDP is proposed as a solution for challenges related to the development of future connected vehicles. In particular, the increased utilisation of server capabilities is proposed as an alternative approach compared to the “traditional approach” of vehicle-internal integration of functionalities (see Section 2.6).

To make the server capabilities an equal part of the automotive software and system landscape, the server entities should be usable similarly to the vehicle-internal entities. This means during the development process, the ASDP shall support the process in that the actual distribution or location of functionalities can be made at the transition from the logical architecture to technical architecture. From these considerations the following ASDP requirement is derived:

REQ 1: The ASDP shall provide a standardised End-to-End (E2E) solution supporting the integration of automotive functionalities on all entities of the vehicle-to-backend platform

Furthermore, various feature configurations are possible for different vehicle series. Even for a single vehicle series, several feature variants should be provided. Moreover, in the future, an increasing customisability shall be possible (see Section 2.1.1). These aspects have two dimensions: On the one hand, OEMs and developers are requesting enhanced reusability of functionalities (see Section 2.3.3), leading to the following requirement:

REQ 2: The ASDP shall support the reusability of functionalities.

On the other hand, functionalities shall be portable between entities, such as the vehicle (i.e., OEM entity) and the backend server (i.e., OEM server). This is also motivated by the assessment results of selected functionalities with regard to criteria for their distribution between the vehicle and the backend (see Section 3.3). It showed that final decisions about the distribution of functionalities must be performed considering the overall system configuration and strategies. Concerning this, the following requirement is derived:

REQ 3: The ASDP shall support the portability of functionalities between network nodes.

To support the gradual introduction of such a distributed ASDP, the migration paths shall exist, enabling, e.g., the interworking with legacy vehicular and server functionalities, leading to the following ASDP requirement:

REQ 4: The ASDP shall enable migration paths for legacy functionalities.

3.4.2 The Vehicle as Part of an Internet of Things

Within prior viewpoint, a vehicle-centric or user-centric perspective is dominant, focusing the functionalities or services consumed by the vehicle or user. Another perspective on future connected vehicles opens up starting with regard to some ADAS or particularly with ITS scenarios: Here, in many cases the vehicle also functions as a data or service provider. One example is the XFCD scenario, introduced in Section 3.2.1; another one is the vehicle maintenance or fleet management as presented in Section 3.2.2. Particularly the former XFCD functionality is an example for the new kind of vehicular functionalities where the vehicles provide their data to backend server facilities to enable superior use cases. In the context of XFCD, this is the derivation of traffic or environmental situations on certain roads based on XFCD data from many vehicles.

In such cases, the future vehicle gets “embedded” into information, business, or social processes, making it a prime example for the visions related to an Internet of Things (IoT). According to the IoT European Research Cluster (IERC), the IoT can be defined as follows:

"[The Internet of Things (IoT) is a] dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'things' have identities, physical attributes and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network." (European Research Cluster on Internet of Things (IERC), 2017)

Moreover, assuming connected vehicles are finally holistically embedded into an ITS, respectively IoT, their functionalities will be highly inter-connected and inter-related: For instance, the XFCD (see Section 3.2.1) is not only used to gain information about the traffic, road or environmental situation on certain roads. This information is also transferred back as input for enhanced navigation systems, where it is utilised to enhance the route planning (see Section 3.2.3). In this regard, an ITS becomes a system-of-systems, with the following characteristics (Maia et al., 2014; Maier, 1996; Tolk & Jain, 2011):

- Operational independence of the individual systems
- Managerial independence of the systems
- Geographic distribution
- Emergent behaviour
- Evolutionary development

Fielding found in (2000): "Most software systems are created with the implicit assumption that the entire system is under the control of one entity, or at least that all entities participating within a system are acting towards a common goal and not at cross-purposes. Such an assumption cannot be safely made when the system runs openly on the Internet.". This is especially true for the automotive domain, where OEMs are used to explicitly and comprehensively control the whole software and system landscape. However, for the future vehicle as part of an ITS, which is connected with other entities over the Internet, thus building a system-of-systems, this is no longer a given. Fisher found in (2006): "Consequently, many of the techniques and approaches of traditional software and systems engineering are ineffective and sometimes counterproductive in systems of systems".

Reflecting these new kinds of unbounded systems (compared to former bounded systems) (Carney, Fisher, Morris, & Place, 2005), related distributed functionalities are not just integrated, but emerge by the interoperation of individual systems. This generates a number of challenges. The following requirement is considered as necessary foundation to enable system-of-systems, although it may not be sufficient.

REQ 5: The ASDP shall support an open and standardised approach, enabling cross-series, cross-vendor, and cross-domain interoperability.

Such examples show that the future connected vehicles will provide data without interaction with human beings, which goes beyond the individual usage scenarios of a driver. In this regard, the future vehicle is not just connected with the Internet and server facilities.

3.4.3 Towards an (Automotive) Embedded Internet

The emergence of connected vehicles is a significant part of the change in mobile Internet usage. Distributed ITS and ADAS functionalities as well as other vehicle-to-backend functionalities, e.g., sub-functionalities from the navigation system, will increase the amount of communication independent from human interaction, referred to as M2M communication (see Section 3.3.1). Hu et al. found in (2011): "Different from the traditional human to human (H2H) communications for which the current wireless networks are designed and optimized, M2M communications is seen as a form of data communications between entities that do not necessarily need any form of human intervention."

The M2M communication differs in many aspects from human-based communication (Boswarthick, Elloumi, & Hersent, 2012; Laya, Alonso, & Alonso-Zarate, 2014), whereas the automotive domain, with its powerful in-vehicle "machines" (see Section 2.2.1) and wide range of applications constitutes a specific class of M2M communication in itself with a high variety of functionalities, of which some probably have even higher requirements compared to those of other M2M domains (Booysen, Gilmore, Zeadally, & van Rooyen, 2012; Lequerica et al., 2010; Pereira & Aguiar, 2014). In addition to others, these are some of the communication specifics of the automotive domain (adopted from Laya et al., 2014):

- **Traffic direction:** The amount of uplink traffic is increased.
- **Message size:** For some applications, such as FCD status information, the message size can be very small.
- **QoS:** The QoS requirements range from lower than human-based communication (e.g., very time-tolerable status information) to real-time requirements (e.g., warnings about critical situations).
- **Mobility:** Vehicles have high mobility at potentially high moving speeds.
- **Number of devices:** The connection of every vehicle with the Internet can lead to a significant increase in network participants. In particular during high traffic volume situations (e.g., traffic jams), the number of devices can temporarily increase significantly in a certain area.

In this regard, M2M communication puts new requirements on mobile networks, driving a transformation towards an "Embedded Internet" (Wu, Talwar, Johnsson, Himayat, & Johnson, 2011), or with focus on this research and specifics previously listed, towards an "Automotive Embedded Internet". These requirements have to be addressed at different layers of the network stack, including the physical air interface, as well as the protocols above (Pereira & Aguiar, 2014).

This means that physical air interface advancements might not be sufficient, if the communication mechanisms of an ASDP layered above are not adequate. In this regard, this also puts requirements to the ASDP to enable the envisaged concept and to support the emergence of an "Embedded Internet" for connected vehicles: On the one hand, the ASDP should support the usage of state-of the art protocols and network mechanisms for M2M traffic. This refers to the foundation of the ASDP. On the other hand, the ASDP shall provide appropriate capabilities to the distributed automotive functionalities that facilitate effective data exchange between functionalities on the vehicle or backend or other network entities. This also reflects the assessment results regarding the criteria for function distribution as presented in Sections 3.3.6 and 3.3.7. All these aspects decide, whether an ASDP enables that distributed functionalities can be realised with the intended quality and aligned to the network capabilities (see Section 2.2.2), which means that the resulting network requirements can be fulfilled by the network. Following this, the subsequent ASDP requirement is derived:

REQ 6: The ASDP shall provide appropriate mechanisms to prevent network misalignment

Finally, it should be emphasised that the automotive engineers are required to use these mechanisms in a thorough manner: The considerations of IoT and system-of-systems (see Section 3.4.2), whereupon here the whole system is not under control of a single party (e.g., the OEM), similarly apply to wireless cellular networks of the vehicles. In this regard, the external vehicle connections significantly differ from the in-vehicle connections that OEMs are familiar with. This requires strong expertise in the technologies related to connected vehicles and vehicle-to-backend platforms because advanced wireless cellular networks, protocols, and ASDP capabilities cannot overcome inadequately or non-cooperatively designed distributed functionalities.

3.5 Summary

This chapter introduced the concept of a distributed Automotive Service Delivery Platform, which is proposed as a vehicle-to-backend platform solution for future connected vehicles. Taking into account the characteristics, challenges and gaps within current automotive software and system development, as well as the anticipated developments driven by the functional domains IVI, ADAS, and ITS, firstly principles are developed. These lead to the distributed ASDP as enabling platform for the concept of the same name.

In advance of the elicitation of fundamental requirements to such platform, three related scenarios and criteria for the distribution of automotive functionalities between the vehicle and backend server facilities are discussed. The major identified aspects are integrated in three viewpoints which reflect the domain-internal view of the current automotive engineers, the connected vehicle as part of an Internet of Things, and the transformation of telecommunication toward an Automotive Embedded Internet. Based on these viewpoints, six fundamental

requirements to the distributed ASDP are derived. In this regard, besides the introduction of the concept as general solution idea, this chapter primarily detailed the problem space which facilitates both profound identification of solution hypothesis and the assessment of its appropriateness.

4 An Automotive Service Delivery Platform Based on the oneM2M Service Platform

The previous chapter describes the concept of a distributed Automotive Service Delivery Platform (ASDP) as an approach to address the challenges related to software and system architectures of connected vehicles.

This chapter introduces the oneM2M Service Platform as specified within its release 1[13] (oneM2M TS-0001, 2015; oneM2M TS-0004, 2015). As Section 4.2 discusses in more detail, the oneM2M service platform is considered a good candidate to enable the concept of an ASDP because the oneM2M service platform has been developed as enabler for an IoT, of which ITS are an intrinsic part (see Section 3.4.2). Besides, the problem space of both, the distributed ASDP concept, and the oneM2M service platform share several ideas and requirements. However, this does not necessarily mean that the oneM2M service platform is also an adequate enabler for the ASDP concept.

The intention of this chapter is neither to duplicate the standards documents (i.e., technical reports and technical specifications) nor to write a comprehensive book about the oneM2M service platform covering absolutely all its details. The aim here is to give an introduction to the essential architectural elements and to make key architectural design decisions clear (see Section 2.4.4), for following reason: The observation of Jansen and Bosch in (2005), after which "[currently] almost all the knowledge and information regarding the design decisions on which the architecture is based on (e.g., results of domain analysis, architectural styles used, trade-offs made etc.) are implicitly embedded in the architecture, but lack a first class representation.", also applies to the oneM2M service platform. The oneM2M standards documents specify only "the/their solution", which is similar to many other standards or architecture documentations, but it also misses links of key architectural design decisions to objectives and identified requirements for a oneM2M platform. Furthermore, decisions about architectural trade-offs are not transparent. Even though hundreds of documents of the working groups are publicly available, including as well diverse discussions about architectural design

[13] Whenever this research links to the current version of oneM2M standards, the standards from the release 1 are referred to, particularly TS-0001-V.1.6.1 published on 30 January 2015 (oneM2M TS-0001, 2015), and TS-0004-V.1.0.1 published on 30 January 2015 (oneM2M TS-0004, 2015), unless other releases or versions are explicitly referred to.

decisions and necessary trade-offs, the oneM2M standard lacks linkage and disclosure of those. Hence, the oneM2M "software architecture [also] lacks a clear view on why the architecture looks as it does" (Jansen & Bosch, 2005), leading to the following consequences:

- Evolution in harmony to the current architecture and the design philosophy is difficult. This often leads to erosion of the initial architecture over time which may have unintended and negative effects (Jansen & Bosch, 2005; Strasser et al., 2014).
- It is difficult to anticipate the appropriateness of the software architecture, platform or standard, particularly if it is still in maturation like the oneM2M service platform. This is caused by the fact that during this time empirical measurements and tests are not possible due to the absence of full implementations. And even if implementations would exist, the significance of tests might be limited with respect to the evaluation whether a single architectural design decision poses a beneficial or adequate trade-off.

For these reasons, this chapter is also intended to make fundamental architectural design decisions of the oneM2M service platform transparent, through the identification of utilised paradigms, design principles, and technologies. This enables architecture analysis already early in the design process (see Section 2.4.6), which suits well to the current phase of oneM2M standards development. Furthermore, it utilises the fact that for many of those architectural design decisions analysis exists concerning to which requirements or qualities they contribute. Accordingly, this analysis is part of the introduction of the elements and functionalities of the oneM2M standard performed in the Sections 4.3, 4.4, 4.4.4, and 4.5. Afterwards a reference configuration of oneM2M for the distributed Automotive Service Delivery Platform is presented and basic communication scenarios are considered.

4.1 Introduction to M2M, oneM2M

This section provides an introduction to M2M communication in general and the oneM2M standard in particular. Additionally, related work to the oneM2M service platform is referenced.

4.1.1 M2M Communication

Smart Home, Smart Grid, Smart Cities, Intelligent Transportation Systems, and eHealth are currently subjects of research: For example, a Smart Home can increase the comfort of its residents and its energy efficiency through intelligent applications, integrating temperature sensors, thermostats, weather forecasts, energy meters, lights, washing machines, etc. Within the wider Smart Grid context, instantly-available consumption measurements from households and industry, together with derived predictions about upcoming energy demands, enable increased coordination of power plants' energy productions. This gains importance with respect to the growing volume of renewable energy the production of which is typically more volatile (e.g., contingent upon varying weather conditions). It is further important, because renewable

energy is much more decentralized, thus requiring a higher level of control efforts than conventional power plants.

Basis of these scenarios is the inter-connection of a myriad of devices (and applications) across vendors and domains. Machine-to-Machine Communication (M2M), also known as Machine-Type-Communication (MTC), is the general term for the related enabling platforms and technologies.

In this regard, M2M has overlaps with the Internet of Things (IoT), but: "IoT is dealing with things and objects [like Radio-Frequency Identification (RFID) 'tagged' objects] that may not be in a M2M relationship with an ICT [(Information and Communications Technology)] system. […] These objects are passive and have no direct means with which to communicate 'upstream' with the M2M application" (Boswarthick et al., 2012, p. 5). Since M2M devices, in contrast, always come with their own communication capabilities, M2M can be considered as a subset of the IoT. M2M devices might offer HMIs, but they essentially include functions with no user interaction.

4.1.2 oneM2M Standard

The term M2M initially only refers to the named inter-connection and communication of machines. Initially, this is independent from standards and technical solutions that aim to enable such Machine-to-Machine communication. Moreover, there is no single M2M standard or technology – M2M standards can be considered as an ecosystem and it is a collection of several single parts, developed by different research institutes, initiatives, companies and Standard Developments Organisations (SDOs).

This research uses the oneM2M Service Platform, which is basically a middleware that enables M2M communication. It has been developed and standardised by the standards initiative oneM2M with the objective of bundling the main standardisation activities for global harmonisation since July 2012. The oneM2M standards are built on previous work of ETSI, in particular these specifications:

- ETSI TS 102 689: "M2M service requirements" (ETSI TS 102 689, 2013)
- ETSI TS 102 690: "Functional architecture" (ETSI TS 102 690, 2013)
- ETSI TS 102 921: "mIa, dIa, mId interfaces" (ETSI TS 102 921, 2013)

The oneM2M initiatives consists of SDOs from all over the world, namely:

- Association of Radio Industries and Businesses (ARIB), Japan (ARIB, 2017)
- Alliance for Telecommunications Industry Solutions (ATIS), US (ATIS, 2017)
- China Communications Standards Association (CCSA), China (CCSA, 2017)
- European Telecommunications Standards Institute (ETSI), Europe (ETSI, 2017)
- Telecommunications Industry Association (TIA), US (TIA, 2017)

- Telecommunications Standards Development Society (TSDSI), India (TSDSI, 2017)
- Telecommunications Technology Association (TTA), Korea (TTA, 2017)
- Telecommunication Technology Committee (TTC), Japan (TTC, 2017)

These partners collaborate with industry fora and consortium:

- Broadband Forum (Broadband Forum, 2017)
- European Committee for Standardization, Comité Européen de Normalisation (CEN) (CEN, 2017)
- European Committee for Electrotechnical Standardization, Comité Européen de Normalisation Électrotechnique (CENELEC) (CENELEC, 2017)
- GlobalPlatform (GlobalPlatform, 2017)
- Open Mobile Alliance (OMA) (OMA, 2017)

These and over 200 additional member organisations are working on the oneM2M standards in order to develop and maintain the globally standardised oneM2M Service Platform enabling the objectives associated with M2M communication.

4.1.3 Related Work

As stated above, the M2M standards form an ecosystem. Accordingly, there is some work related to the oneM2M service platform. Those most relevant for this research are named below.

Basically, related work exists on various level. Several initiatives and activities have recently attempted to classify and sort them. Examples are ITU-T Y.2060 in 2013 (ITU-T, 2013), IoT-A in 2014 (Bassi et al., 2013), and AIOTI in 2017 (AIOTI, 2017), which came up with an architecture reference model that aims to provide the superior abstract framework to sort concrete implementation of reference architectures such as the oneM2M service platform as well as other standards or building blocks of the M2M landscape (AIOTI, 2017). AIOTI accompanies this, e.g., with a clustering regarding knowledge areas, such as: communication and connectivity, integration/interoperability, applications, infrastructure, IoT architecture, devices and sensor technology, and security and privacy.

Another related activity is AllJoyn (Alljoyn, 2017), which was initially provided by the AllSeen Alliance. AllJoyn is a collaborative open source software framework aiming to provide seamless Device-to-Device communication in local networks. Its focus is the smart home domain. Alljoyn is merging with IoTivity (IoTivity, 2017), which is a similar activity. IoTivity is the reference implementation of the specifications from Open Connectivity Foundation (OCF), formerly Open Interconnect Consortium (OIC). The focus is also local deployments, e.g., smart home and office scenarios. Although they have started to look into other domains, the oneM2M service platform still could be considered as more universal, since this was already a construction principle. Nevertheless, interworking between both is possible (oneM2M, 2016).

FIWARE (FIWARE, 2017), a result of the EU Future Internet Public-Private Partnership activities, aims to provide a scalable open source platform based on Generic Enablers (GE) that are connected through the OMA Next Generation Service Interface (NGSI) (OMA TS NGSI Context Management, 2012). The basic idea is comparable to that of the oneM2M service platform whereupon functionalities that are shared across many applications are commonly implemented (see oneM2M Common Services Entity layer, Section 4.3.1). However, FIWARE GEs could be individually selected and assembled to meet the individual requirements of a use case which is different from the oneM2M approach. In addition, FIWARE GEs focus less on general communication mechanisms and network specifics but more on data models and data interoperability. Thus, FIWARE is no alternative to deliver the same capabilities as the oneM2M service platform. But interworking between both is possible and they could even complement each other, e.g., as it is described within Wise-IoT (M. Bauer, 2017; Wise-IoT, 2017).

There also do exist commercial products and APIs for IoT and M2M purposes. Examples are the Amazon AWS IoT Core (Amazon Web Services, 2017), SAP Leonardo IoT (SAP, 2017), or Microsoft Azure IoT Suite (Microsoft, 2017). However, according to the REQ 1 and REQ 5 of the ASDP concept (see Section 3.4), the enabling platform should be open and standardised; it should provide an E2E support, and should enable cross-series, cross-vendor, and cross-domain (i.e., industries) interoperability. In this regard, commercial products or APIs are out of scope of this research.

To conclude, the oneM2M service platform can be considered as the most advanced standardised M2M platform (Medjiah, 2017), which is the reason why it is used for this research. The following section continues the discussion why M2M in general and oneM2M in particular might be an adequate enabler for the distributed ASDP concept.

4.2 M2M as Initial Hypothesis

When considering the problem space by means of challenges, objectives, and requirements related to connected vehicles, vehicle-to-backend platforms, and the distributed ASDP concept, several overlaps exists with M2M communication. This makes M2M communication respectively the oneM2M service platform a promising initial hypothesis with regard to its applicability as enabling platform for the ASDP concept, which means that the oneM2M service platform may provide a suitable solution space. Three main aspects supporting this hypothesis, namely objectives regarding interoperability and network efficiency, and current M2M considerations of automotive use cases. These are presented in the following.

4.2.1 M2M aims Interoperability

Darmoir and Elloumi found in (Boswarthick et al., 2012, p. 5): “In the vast majority of cases, the solutions developed and implemented to date have been addressing specific vertical

applications requirements in isolation from all others. This has created 'silo' solutions based on very heterogeneous forms of technology, platforms, and data models. Interoperability is in general very limited or non-existent". Here, similar to the context of 5G, the term "vertical" refers to a certain industry or sector such as Automotive, Energy, or Health. But, "silo" solutions with limited interoperability may even exist within a single sector, e.g., different OEM backend solutions (see Section 2.5.4). Figure 4.1 illustrates such situation with components from different vendors or domains using heterogeneous hardware, communication technologies and protocols, and application abstractions. Although standardised (and even well-documented) platforms may exist within each domain, they might lack generic interoperability at the necessary level.

Wu et al. found in (2011):"Future M2M ecosystems will be complex and span many industries, including telecom and electronics. Unlike current M2M markets, which are highly segmented and often rely on proprietary solutions, future M2M markets will need to be based on industry standards to achieve explosive growth" This raises the architectural task "to turn a patchwork of standalone elements and solutions into a coherent 'system of systems', gradually turning the focus from the 'what' to the 'how' and developing the appropriate technologies and standards" (Boswarthick et al., 2012, p. 1), to turn "[…] today's INTRAnet of things to a future INTERnet of things […]"(Zorzi, Gluhak, Lange, & Bassi, 2010).

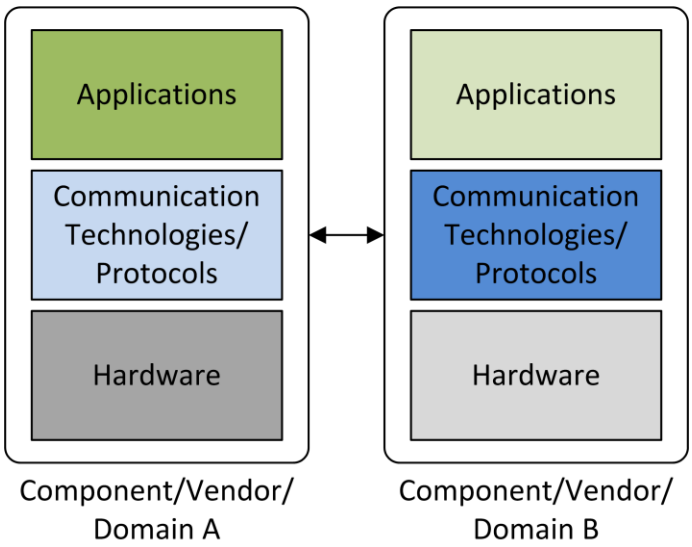

Figure 4.1: Limited interoperability between components, vendors, and domains through heterogeneous abstraction levels, interfaces, and technologies

This problem statement of the M2M market matches with the identified situation in the automotive domain, where proprietary vehicle-to-backend platforms (see Sections 2.5.4 and 2.6) are vertically integrated "silo solutions", which is why the ASDP concept derived the REQ 5 also with regard to the connected vehicle as part of an ITS (respectively IoT) and system-of-systems considerations (see Section 3.4.2).

Interoperability

To enable a better understanding of the interoperability challenges and in advance of the assessment of the oneM2M service platform, the term interoperability is discussed afterwards in more detail. A general definition is provided by Bass et al. in (2012, p. 103):

> *"**Interoperability** is about the degree to which two or more systems [or services, or applications] can usefully exchange meaningful information via interfaces in a particular context" (Bass et al., 2012, p. 103)*

This definition emphasises that interoperability, like every quality attribute, is no binary characteristic. A system or application or service (or whatever two entities are being considered) is not just interoperable or not, the question is to which degree interoperability between systems is present and accordingly how interoperability can be classified.

The term interoperability is used varying in different quality or interoperability models (Brownsword, 2004; The GridWise Architecture Council, 2008; Tolk, Diallo, & Turnitsa, 2007). A comprehensive interoperability model is the Levels of Conceptual Interoperability Model (LCIM), first described by Tolk and Muguira in 2003 (Tolk & Muguira, 2003). Although initially developed in the context of modelling and simulation, it provides general means to classify the interoperability level between systems including M2M communication. Besides descriptive mode, the LCIM supports also prescriptive assessment, i.e., the derivation of requirements to enhance interoperability (W. Wang, 2009). Since its first publication, the LCIM has constantly emerged to the current version describing seven levels of interoperability, which are according to Tolk et al. in (Tolk et al., 2007):

- Level 0: No Interoperability
  The system is stand-alone and has no communication capabilities.
- Level 1: Technical Interoperability
  The systems can exchange data through a common communication protocol.
- Level 2: Syntactic Interoperability
  The systems can exchange data through a common structure, i.e., a data format.
- Level 3: Semantic Interoperability
  The systems share the same meaning of the exchanged data.
- Level 4: Pragmatic Interoperability
  The systems are aware of the mutual application context, i.e., employed methods and procedures.
- Level 5: Dynamic Interoperability
  The systems are aware of the mutual effects of the information exchange, e.g., state changes over time.
- Level 6: Conceptual Interoperability
  The systems share one conceptual model, i.e., a "fully specified but implementation independent model" (Davis Paul & Anderson Robert, 2003).

If systems shall be empowered to be interoperable with each other on the respective level, the interoperability gap must be identified and closed. The GridWise Architecture Council names this gap “distance to integrate” (The GridWise Architecture Council, 2008), but with respect to the intrinsic characteristics of system-of-systems, the naming “distance to interoperability” is considered as more precisely here.

This also facilitates a more accurate definition and understanding of a “silo solution”, namely: A “silo solution” lacks interoperability capabilities at the required level with other systems, meaning the “distance to interoperability” is not zero. Considering this, each system that has been developed independently at first might be a silo solution. The use of service-oriented or component-based design approaches does not change this fact. However, it might reduce the necessary efforts to achieve interoperability at the envisaged level.

This puts further important aspects into place: Interoperability can be achieved manually through engineering efforts (e.g., the development of appropriate adaptors), or automatically by the systems themselves. While the first refers to the design-time of the system, the latter is the preliminary to enable the achievement of interoperability during run-time of the system (Carney et al., 2005). Finally, automatic run-time interoperability without a priori knowledge is the ultimate goal for ITS, M2M, and visions to materialise, because this is a fundamental capability to enable emergent behaviour of system-of-systems (Blair, Paolucci, Grace, & Georgantas, 2011; Carney et al., 2005).

4.2.2 M2M Aims Network Efficiency

M2M communication differs in various ways from the traditional communication, as it occurs when human beings use the communication technologies (referred to as H2H communication) (R. Q. Hu et al., 2011). Following Laya et al., these differences exist with regard to (Laya et al., 2014):

- Traffic direction
- Message size
- Connection and access delay
- Transmission periodicity
- Mobility
- Information priority
- Security and Monitoring
- Lifetime and energy efficiency

Addressing these changed usage patterns and requirements of M2M communication is the intrinsic objective of the same-titled research and development efforts and the related enabling platforms, such as the oneM2M service platform. Considering this, M2M communication shall

be facilitate the necessary change from the mobile Internet towards an “Embedded Internet” (Wu et al., 2011) which is required to effectively support the communication of machines.

Section 3.4.3 pointed out that the amount of such new M2M communication traffic of connected vehicles increases with scenarios, such as ITS. Although vehicles constitute an own class within M2M communication, with a specific instantiation or subset of requirements (see Section 3.4.3), it could be expected that the requirements of an “Automotive Embedded Internet” are considered and fulfilled by an “Embedded Internet”. Not least, the automotive domain is explicitly considered within M2M scenarios, which is detailed in the next section.

4.2.3 M2M Considers the Automotive Domain

The automotive domain is emphatically constituted as part of M2M communications. With status of April 2013 ETSI names the following use cases in their Technical Report “Machine to Machine communications (M2M); Use cases of Automotive Applications in M2M capable networks” (ETSI TR 102 898, 2013):

- Electric Vehicle Charging
- Fleet Management / Theft Tracking
- Vehicle-to-Infrastructure communication

This is considered as another indicator that the M2M platform might provide appropriate capabilities to facilitate distributed automotive functionalities and thus the concept of a distributed ASDP.

4.3 Functional Architecture

The previous sections motivated the consideration of M2M communication and hence the oneM2M service platform as promising candidate to enable the ASDP concept by emphasising overlaps of the respective problem spaces. In the following, the oneM2M service platform is introduced in more detail to enable the assessment how appropriate its provided solution space addresses the identified requirements of the ASDP concept.

Due to this regard, firstly the functional architecture of the oneM2M service platform is introduced. This includes its layers, reference points, Common Service Functions, protocol stack and bindings, domains, nodes, and configurations.

4.3.1 Layers and Entities

To overcome widespread vertically-integrated applications, oneM2M specifies a universal horizontal integration platform. Following this approach, oneM2M splits the overall functionality into common services and application-specific parts. Furthermore, an underlying network layer may provide additional network services. This results in three layers respectively entities, as described in the following:

- **Application Entity (AE):** An AE contains the application service logic of a oneM2M application. Hence, it implements the actual use cases and functionalities, such as Extended Floating Car Data, Vehicle Maintenance, or a Vehicle Data Provider. According to the envisaged split between the AE and the CSE, the AEs shall contain the application-specific parts of the applications that are not common to other applications.
- **Common Services Entity (CSE):** "A Common Services Entity represents an instantiation of a set of 'common service functions' of the M2M environments" (oneM2M TS-0001, 2015, p. 18). In this regard, the term CSE must be used slightly differently to the AE and NSE since the plural usage of CSEs describes CSEs residing on different nodes and does not describes single common services within this CSE, because they are "logically and informatively conceptualized as Common Services Functions (CSFs)" (oneM2M TS-0001, 2015, p. 18). According to the envisaged split between the AE and CSE, the CSE shall contain the common parts of the distributed application functionalities that are to be shared by typical/most AEs.
- Underlying **Network Services Entity (NSE):** The underlying networks basically provide data transport and communication services to the CSE. These capabilities are separate and not explicitly referred within the service entities which is the reason why the NSE is often not explicitly mentioned. The NSE is introduced to encapsulate additional services of the underlying networks beyond data transport and communication services. Examples are location services, device management, or device triggering (oneM2M TS-0001, 2015; Song, 2014).

The AEs are located in the application layer, the NSEs are located in the underlying network services entity layer, and the CSE can also be considered as a layer, containing an instantiation of Common Services Functions. However, oneM2M does not introduce dedicated terms or abbreviations for those layers. Due to this, the AE, CSE, or NSE sometimes may more suitably be understood as the collection of the entities or functions, and hence as layers. Therefore, in the following the distinction between the respective entity layer and a single entity or function is explicitly made in descriptions, when it serves to accommodate better understanding, although not requested by the oneM2M standard.

Figure 4.2 illustrates how the unified layers of the oneM2M service platform improve the interoperability between components, vendors, and domains.

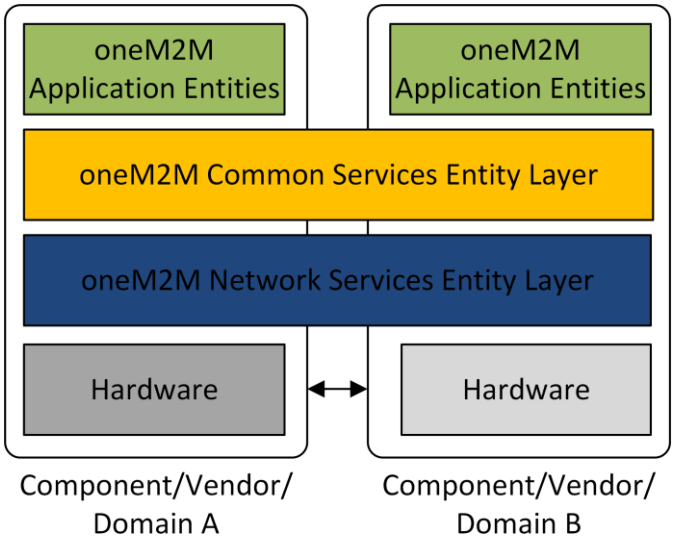

Figure 4.2: Improved interoperability between components, vendors, and domains through unified abstraction levels, interfaces, and technologies

The introduction of the CSE layer constitutes a middleware approach. The CSE abstracts AEs not only from the specifics of certain hardware, but also from the specifics of network technologies and protocols. This includes specific network services that may only be provided from certain network technologies. Furthermore, the approach to implement functionalities that are to be shared between different AEs commonly in the CSE reduces non-application-specific parts within the AEs. These architectural design decisions contribute to interoperability and portability of AEs and hence REQ 5 and REQ 3 of the ASDP concept (cf. Bass et al., 2012). Furthermore, this also contributes to the REQ 2 and REQ 3 (cf. Bass et al., 2012).

4.3.2 Reference Points

Additionally, oneM2M specifies four reference points for the communication between the three layers (see Section 4.3.1), whereby "a reference point consists of one or more interfaces of any kind" (oneM2M TS-0001, 2015, p. 18). These reference points are independent from basic transport and connectivity services of the Underlying Network, which is required in any case, for the communication between oneM2M Entities(oneM2M TS-0001, 2015, p. 18).

The oneM2M nomenclature for the reference points is basically "Mc" followed by the first letter of the Entity to which the reference point is provided from the perspective of the CSE layer. Accordingly, these are the four reference points(oneM2M TS-0001, 2015, p. 18):

- **Mca**: This reference point is located between the CSE and the AE layer. Accordingly, AEs use this reference point to use the services of the CSE and to communicate with other AEs on the same or on other nodes. Since the AE and the CSE are not necessarily co-located within the same node (see Section 4.3.5 and 4.3.6), the Mca reference point may also be used between nodes.

- **Mcc**: This reference point is located between two CSE layers and enable the mutual provision of the CSE services and thus inter-CSE communication (oneM2M TS-0001, 2015; Swetina, Lu, Jacobs, Ennesser, & Song, 2014).
- **Mcc'**: This reference point is located between two CSE layers, residing on Infrastructure Nodes in different service provider domains. The Mcc' similar to the Mcc also enable the mutual provision of the CSE services and thus inter-CSE communication. To facilitate the restriction of service provision between service providers and to thus enable differentiation between domain-internal services and inter-domain services, this particular Mcc' reference point is specified (oneM2M TS-0001, 2015, p. 19).
- **Mcn**: This reference point is located between the CSE and the NSE. It enables the CSE to use additional services (besides basic transport and connectivity services) of the Underlying Network that may be available (oneM2M TS-0001, 2015, p. 18).

The location of these reference points is illustrated in Figure 4.4.

4.3.3 Common Services Functions

The CSE is the core of the oneM2M Service Platform, since it constitutes the middleware that provide various services across the reference points to the other Entities (see Section 4.3.2). The overall CSE services and functionalities are composed of Common Services Functions (CSF), see Figure 4.3.

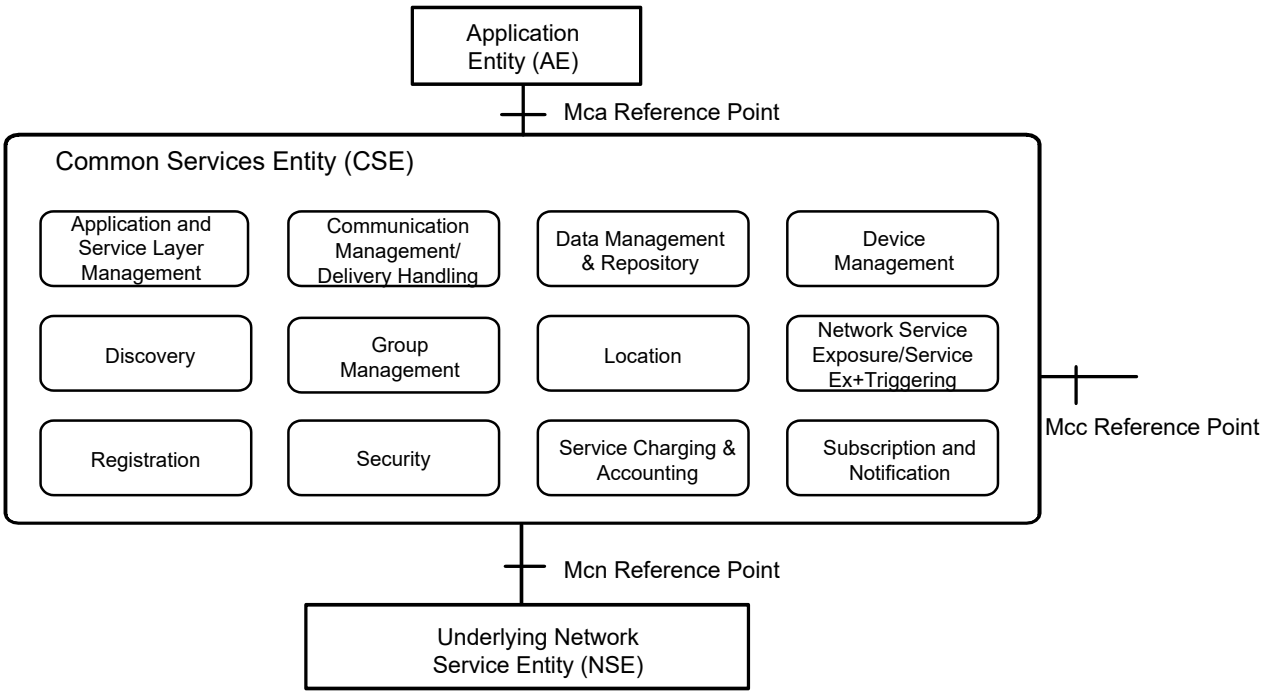

Figure 4.3: Common Service Functions in the Common Services Entity (Source: oneM2M TS-0001, 2015)

The oneM2M standard specifies the services by means of their external capabilities and behaviour via the reference points. Due to this, the following CSF descriptions are merely informative and serve the purpose of describing existing functionalities by grouping them logically together as CSFs. However, oneM2M does not prescribe this CSF splits, and also does not specify the CSE-internal management and interworking of the CSFs (oneM2M TS-0001, 2015, p. 21).

The CSF descriptions, provided in the following, are intended to provide an overview of the related functionalities and not to fully specify the functionalities. Since it is not the intention here to duplication the oneM2M standard, the descriptions provided are tailored to the necessities of this research. Nevertheless, every CSF is introduced at least briefly, to describe the philosophy behind it, which is of major importance to understanding the oneM2M Service Platform, its overall volume of functionality, its general design philosophy, as well as its design space, which is provided to AEs and their users (cf. Boswarthick et al., 2012; oneM2M TS-0001, 2015).

Application and Service Layer Management (ASM)
The ASM CSF deals with the management of AEs and CSEs on Nodes. This includes functionalities related to the lifecycle of software packages for the CSE and AE consisting of states such as: Installing, Installed, Updating, Uninstalling and Uninstalled, and transitions between those states actioned when, e.g., applying Download, Install, Update, or Remove to the software package (oneM2M TS-0001, 2015, p. 22). Furthermore, the lifecycle management of a software module is supported, e.g., by means of the action Start and Stop which trigger transitions between states, such as Idle, Starting, Active, and Stopping (oneM2M TS-0001, 2015, p. 22). The ASM CSF also provides capabilities to configure software packages (oneM2M TS-0001, 2015, p. 21).

Discovery (DIS)
The CSE is not a centralised entity but could be part of various nodes within the oneM2M landscape. Hence, the oneM2M Service Platform builds a distributed Service Delivery Platform. Service-orientation, or rather late binding, requires the discovery of functionalities prior to interworking during run-time. This means, for example, prior to data exchange between oneM2M resources (e.g., AEs or CSEs), these in general must be discovered. Thereby, discovery can be defined as follows:

> *Discovery: The act of locating a machine-processable description of a Web service-related resource that may have been previously unknown and that meets certain functional criteria. It involves matching a set of functional and other criteria with a set of resource descriptions. The goal is to find an appropriate Web service-related resource. (W3C, 2013)*

For this reason, the CSE provides a Discovery (DIS) CSF. It facilitates the discovery of resources according to filter criteria such as the resource type, resource timestamps (e.g., creation time), and descriptive labels. In this regard "[t]he scope of the search could be within one CSE, or in more than one CSE" (oneM2M TS-0001, 2015, p. 30), which builds a distributed discovery functionality.

Registration (REG)

The REG CSF enables the registration of an AE at a CSE of a MN or of the IN, or the registration of a CSE at another CSE. It is the prerequisite that the registering entity is capable of using the services of the respective CSE. The registration between CSEs is not mutually achieved, but only from the child position of the hierarchy to its parent (see Section 4.3.6), accordingly:

- The CSE of an ASN may register with the CSE of a parent CSE of a MN or of an IN.
- The CSE of an MN may register with the CSE of a parent CSE of another MN or of an IN.

After completion of the unidirectional registration procedure, the communication is possible in both ways (given that the registering entity is server-capable, which is given for every CSE, but could be absent for AEs). Finally, besides identification and contact information, the registering CSE may also provide a reachability schedule (on per Node basis) if it is not always reachable (oneM2M TS-0001, 2015, p. 33).

Communication Management, Delivery Handling (CMDH)

The CMDH CSF basically provides capabilities for the communication of the CSE with AEs, NSEs, and other CSEs. It "decides at what time to use which communication connection for delivering communications (e.g., CSE-to-CSE communication) and, when needed and allowed, to buffer communication requests so that they can be forwarded at a later time" (oneM2M TS-0001, 2015, p. 22). This facilitates the specification of delivery policies for each communication request through the originator. Besides, Service Provider can specify CMDH policies that regulate the usage of Underlying Network(s), e.g., according to service contracts. Finally, the CMDH also considers reachability policies if specified for the receiving node, e.g., during registrations (cf. REG CSF).

The CMDH is an important CSF of the oneM2M service platform with regard to the handling of M2M communication specifics (see Section 4.2.2). It has significant share in improving "network-friendliness" of M2M traffic, e.g., by means of aggregation of small data packets to bigger data packets. Considering this, the CMDH CSF contributes to the REQ 6 of the ASDP concept.

Group Management (GMG)
The GMG CSF provides capabilities "to perform bulk operations on multiple devices, applications or resources that are part of the group" (oneM2M TS-0001, 2015, p. 30). It can leverage broadcast or multicast capabilities of the Underlying Network where applicable. This is transparent to the using entity, such as an AE.

To enable group management capabilities, the GMG CSF facilitates the creation, retrieving, updating, or deletion of a group, including joining and leaving the group according to policies such as maximum number of group members, by use of the respective resource. It is the responsibility of the GMG CSF to utilise the functionalities of other CSFs in the necessary way to transfer the operations performed on the group internally to all group members. In addition to others, these are functionalities related to SEC, SN, DMG.

Security (SEC)
The SEC CSF provides services related to security of the distributed oneM2M Service Platform. Among others, this comprises(oneM2M TS-0001, 2015, p. 34):

- Sensitive data handling.
- Security administration.
- Security association establishment.
- Access control.
- Identity protection.

Thus, the SEC CSF provides services for basic security goals, such as confidentiality, integrity, and authenticity.

Data Management & Repository (DMR)
Within the CSE a great deal of data must be stored and managed. This not only relates to application data and its exchange but also to manifold configuration and management data, which is also part of the resource tree (see Section 4.4.3). Due to that reason, the DMR CSF is introduced, providing Data Management and Repository services at an abstracted level to other CSFs of the CSE. The functionalities also comprise the aggregation of data from different entities and the mediation of data between different entities. In the future, semantic interoperability may become another important functionality provided by the CMR CSF (oneM2M TR-0007, 2015). Currently, such capabilities are limited to the storage of semantic descriptions for resources (oneM2M TS-0001, 2015). Furthermore, the DMR may become enhanced towards a Big Data Repository, facilitating related scenarios (oneM2M TS-0001, 2015, p. 23).

Location (LOC)
The LOC CSF provides services to AEs, which facilitate the acquisition of geographical location information of Nodes, e.g., for location-based services (oneM2M TS-0001, 2015, p. 31). For this purpose the LOC may use any of the following (oneM2M TS-0001, 2015, p. 31):

- a location server from the underlying network (e.g., by means of Cell-ID, assisted-GPS, or fingerprint);
- a GPS module of the M2M device; or
- information for inferring the location of other nodes.

Service Charging & Accounting (SCA)
The SCA CSF provides services with respect to charging and accounting. In accordance with the general approach of oneM2M and the CSFs of its CSE, the SCA is independent of network and technology, although it may utilise capabilities provided by the underlying networks. The charging models provided by the SCA may also reflect the oneM2M capabilities. For example, charging could be subscription-based or event-based (oneM2M TS-0001, 2015, p. 35).

Device Management (DMG)
Another important CSF for M2M environments is Device Management (DMG). In contrast to the ASM CSF, DMG addresses more low-level aspects, such as:

- Device Configuration.
- Device Diagnostics and Monitoring.
- Device Firmware Management.
- Device Topology Management.

The necessity for device management is not new. It is rather a well-known requirement, e.g., in the area of mobile devices and network equipment. Thus, oneM2M here again can utilise mature and widely-used standards. According to the philosophy of a technology-independent horizontal CSE, these are adapted and wrapped for common usage in the oneM2M CSE. Currently, oneM2M specifies bindings to the OMA Device Management Protocol (OMA TS DM Protocol, 2016) and the OMA Lightweight Machine to Machine Protocol (OMA TS LightweightM2M, 2016), (oneM2M TS-0005, 2016). Furthermore, it provides bindings to the BBF TR-069 protocol (Broadband Forum, 2013).

Network Service Exposure, Service Execution and Triggering (NSSE)
The NSSE CSF "manages communications with the Underlying Networks for accessing network service functions over the Mcn reference point" (oneM2M TS-0001, 2015, p. 32). It is the adaptor for functions of specific network technologies and mechanisms of the Underlying Network for the purpose of providing those functionalities to other CSFs and AEs technology-independent (see Section 4.3.2).

Subscription and Notification (SN)
Besides direct communication capabilities for communication between AEs and CSEs, oneM2M also provides indirect communication capabilities. These are provided by the SN CSF. Here, AEs and CSEs can subscribe to particular resource changes including additional notification policies. If such resource change occurs and the notification policies, if present, are fulfilled, the CSE triggers the sending of a notification message to the subscriber. This may also depend on CMDH policies.

Indirect communication capabilities are particularly suitable for oneM2M applications which in many cases are event-triggered. Accordingly, the capabilities of the SN CSF are of major importance for the efficiency of distributed oneM2M use cases and hence for the REQ 6. Hence, the SN CSF capabilities provided by the current oneM2M standard are investigated in more detail in the following (see Section 4.5.4 and Chapter 5).

4.3.4 Domains

The oneM2M Service Platform specifies two different domains, where the different nodes are located. These are:

- Infrastructure domain.
- Field domain.

4.3.5 Nodes

The oneM2M Service Platform specifies the following Nodes:

- **Infrastructure Node (IN)**, containing one CSE and optionally one or many AE(s)
- **Middle Node (MN)**, containing one CSE and optionally one or many AE(s)
- **Application Service Node (ASN)**, containing one CSE and at least one AE.
- **Application Dedicated Node (ADN)**, containing only at least one AE but no CSE. This specific node type is possible because the communication across the Mca interface runs through the complete communication protocol stack (see Sections 4.3.2 and 4.4.5).
- **Non-oneM2M Device Node (NoDN)**, containing neither an AE nor a CSE. Hence, the NoDN cannot achieve interoperability with the oneM2M-compliant nodes automatically. However, oneM2M describes several interworking scenarios possible for NoDN (oneM2M TS-0001, 2015). For example, hybrid applications referred to as Inter-working Proxy Application Entity (IPE) can implement the non-oneM2M interface from the NoDN and the oneM2M-compliant Mca interface to facilitate interworking. This fulfils the REQ 4 regarding migration paths for legacy functionalities.

The various node types support scalable solutions with regard to the node capabilities.

4.3.6 Configurations

NoDN, ADN, ASN, and MN are located inside the Field Domain. The Infrastructure Domain of a service provider from a functional view contains only one IN. The positioning of the nodes facilitates hierarchical structures where

- One or many NoDN can connect to an ASN, a MN, or IN.
- One or many ADN can connect to an ASN, a MN, or IN.
- One or many ASN can connect to a MN or IN.
- One or many MN can connect to another MN or IN.

Figure 4.4 shows possible configurations supported by the oneM2M service platform at a glance. According to the possible connections named above, even deeper hierarchies (e.g., with several cascaded MNs) are possible that are not shown in this figure. These hierarchical structures contribute to scalability and the REQ 6, because, e.g., communication can be limited to certain sub-structures or regions, and data exchange can be aggregated and pre-processed at the respective intermediary node of the sub-structure. This matches with and complements approaches such as MEC (see Section 2.2.2): For example, an MN could be operated at the edge of the infrastructure network, hosting the CSE or even selected AEs to enable their required QoS. This also contributes to the prevention of network misalignments (cf. REQ 6)

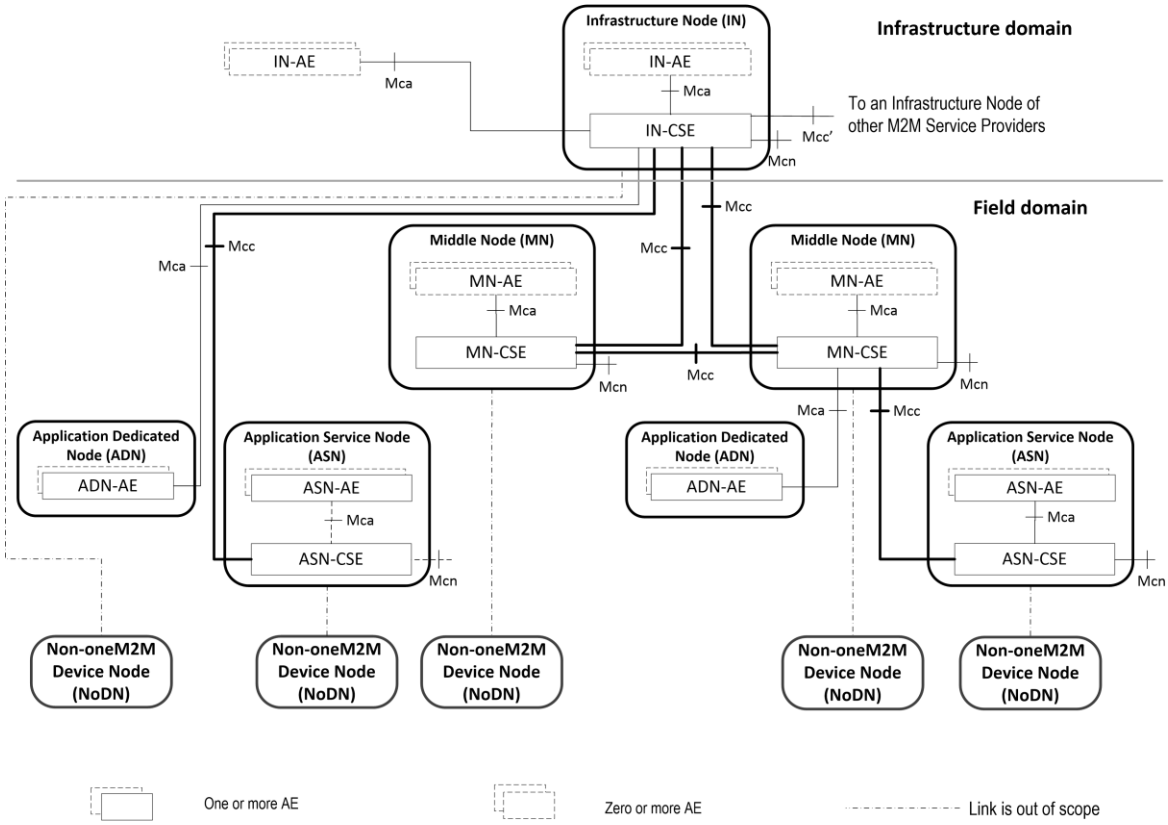

Figure 4.4: Configurations supported by the oneM2M service platform (Source: oneM2M TS-0001, 2015, p. 19)

4.4 Service/Resource Model and Technologies

After presenting the functional architecture of the oneM2M service platform, this section introduces details of the technologies applied. To enable the assessment of oneM2M service platform capabilities, a central aspect is to investigate its service (or resource model) and implementation technology, which show the underlying architectural principles, patterns and tactics. Identifying these architectural principles is important, because it facilitates inference about the architectural capabilities or qualities and thus inference about how suited oneM2M is as the enabler of an ASDP.

Considering this, the service and resource model of the oneM2M service platform is introduced below, including description of the principles and related terms, e.g., service-orientation, SOA, REST, and RESTful. Afterwards, oneM2M resources, methods, and the protocol stack and bindings are presented.

4.4.1 Service-Orientation, Service-Oriented Architecture

The oneM2M standard specifies a Service-Oriented Architecture (SOA) for M2M communication. Following Erl et al. in (2013), service-orientation can be defined as:

> *"**Service-orientation** is a design paradigm intended for the creation of solution logic units that are individually shaped so that they can be collectively and repeatedly utilized in support of the realization of a specific set of strategic goals and benefits associated with SOA and service-oriented computing." (Erl et al., 2013)*

This definition of service-orientation provides the foundation to define the term SOA:

> *"[A] **Service-oriented architecture** is a technology architectural model for service-oriented solutions with distinct characteristics in support of realizing service-orientation and the strategic goals associated with service-oriented computing." (Erl et al., 2013)*

Kral found in (2000) that the "Service-oriented (SO) paradigm and SOA are the proper solution for the majority (if not almost all) of large software (intensive) systems", which matches with the considerations of Tiako et al. in (2008). Consequently, service-orientation is also considered as a solution for automotive software and system development (C. Farcas, Farcas, Krüger, & Menarini, 2010; I. H. Krüger, 2005; I. H. Krüger et al., 2004). Kazman et al. noticed in (2013): "The SOA pattern is a natural and popular choice for many modern systems of systems. It is organized around the concept of loosely coupled, distributed services, which are offered, described, and implemented by service providers."

The reason, why service-oriented approaches are beneficial solutions for software-intensive systems as well as system-of-systems and hence also M2M are the design paradigms that are basically following eight principles (Erl et al., 2013):

- Standardised Service Contract
- Service Loose Coupling
- Service Abstraction
- Service Reusability
- Service Autonomy
- Service Statelessness
- Service Discoverability
- Service Composability

Following service-oriented principles, besides others, contributes to interoperability and maintainability (particularly modifiability) (Bass et al., 2012; Coulouris et al., 2012). Hence, this architectural design decision of oneM2M contributes to REQ 5. Moreover, maintainability respectively modifiability contribute to reusability and thus the REQ 2 (Erl et al., 2013). Additionally, service-orientation provides new capabilities for reusability by means of service composition, whereas according to Vinoski in (2008) "[the} goal of service composition is to reuse existing services by means of assembling them in composite applications that combine them in novel and unexpected ways" (Vinoski, 2008). These kinds of functionalities are also referred to as mashups (cf. Daniel & Matera, 2014), and the dynamic and automated creation of such mashups facilitates smart of intelligent system-of-systems.

4.4.2 REST, RESTful Architecture

There are different approaches to build a SOA and to design its service model and interfaces. One possibility is the adaption of a RESTful SOA approach.

Principles

REpresentational State Transfer (REST) was first described by Fielding in his doctoral thesis in (Fielding, 2000). REST, however, is often misunderstood and misused. As well as SOA, REST is a design paradigm or an architectural style comprised of a number of design decisions (Erl et al., 2013; Richardson & Ruby, 2007). Hence, REST is neither an architecture nor it is tied to the Web or even "depends on the mechanics of HTTP" (Richardson & Ruby, 2007). Architectures, following the REST design criteria, can be referred to as RESTful architectures.

The REST "design decisions are defined through constraints that are associated with architectural properties that represent design goals" (Erl et al., 2013). The formal REST constraints are:

- Client-Server
- Stateless
- Cache
- Uniform Interface
- Layered System

- Code-on-demand (optional)

An important principle of REST is the use of hypermedia as the engine of application state (HATEOAS) (Fielding, 2000). This means, the server can guide the client through the application or API by providing hypermedia (Fielding, 2000; Richardson, Amundsen, & Ruby, 2013). This aspect gains importance with respect to the resource structure, discussed in the Section 4.4.3.

REST and M2M

In contrast to publications and presentations from oneM2M members, such as (Ben Alaya, Medjiah, Monteil, & Drira, 2015; Elloumi, 2014), where the oneM2M service platform is referred to as adapting the RESTful architecture principles, the oneM2M standards documents (oneM2M TS-0001, 2015; oneM2M TS-0004, 2015) lacks in clearly committing or referring to RESTful architecture principles or service-oriented principles. This means neither these terms nor reference to their specifications or design principles are used. The absence of these references aggravates the understanding of the oneM2M design decisions, their trade-offs, and consideration of architectural alternatives. It further contributes to architecture erosion, as discussed at the beginning of this chapter. Thus, it is proposed to refer to the oneM2M service platform as a RESTful SOA, while at the same time clearly stating architectural design decisions that reduce the compliance to the related principles. One example is the cessation of the collection pattern, which is discussed in Section 4.4.3.

According to Erl, the "REST design goals directly or indirectly support and enhance the interoperability potential of services within a service inventory" (Erl et al., 2013, p. 97), whereas the service inventory in the context of oneM2M is the distributed CSE layer. Thus, the selection of REST by the oneM2M standard, despite small deviations, contributes to the REQ 5. Moreover, the REST constraints additionally to SOA are contributing reusability and portability, hence REQ 2 and REQ 3. Not least, the simplicity and lightweight nature of RESTful interfaces are particularly suitable for M2M communications (Pautasso, Zimmermann, & Leymann, 2008).

4.4.3 Resources and Resource Structure

To provide the previously described CSFs following a RESTful SOA approach, the functionalities are mapped to resources.

The ETSI M2M standard, which is the major precursor of the oneM2M standard that was harmonised and evolved to the oneM2M service platform (see Section 4.1), specified a generic resource tree structure (Boswarthick et al., 2012; ETSI TS 102 690, 2013). This resource tree implemented the collection pattern, where resources of the same type are collected (i.e., subordinated) to the respective collection resource, typically indicated by the plural name of the resource type (or a related abbreviation). Following this pattern, all AEs would be

subordinated below a respective collection resource, i.e., ApplicationEntities. This facilitates the retrieve of all resources of a certain type possible through one Retrieve operation on the superior collection resource. It further facilitates the modelling of the scope of those resources that can occur at several positions in the resource tree. Examples are a (data) container resource, subordinated to either an AE, or a (data) container resource, subordinated to a CSEBase resource. While the first corresponds to AE-specific data, the second indicates CSE-global data.

However, oneM2M has broken with this collection pattern substituting the explicit collection-resource-based methods with implicit discovery including filterCriteria and more excessive use of links between resources. The reasons might be the shorting of URIs, a more flexible tree structure, or the reduction of server processing overhead (Ben Alaya, Monteil, & Drira, 2014). However, this decision was subject of discussions, and it is considered as an architectural design decisions decreasing the compliance with RESTful principles (Ben Alaya et al., 2014; Richardson et al., 2013), see Section 4.4.2.

Due to the cessation of the collection pattern and because resources nesting is variable, oneM2M does not specify a resource tree structure, but merely standardises the different resource types which can be arranged in a tree structure and linked together in various ways. Table 4.1 presents a selection of oneM2M Resource Types, including description and selection of possible child or parent resources.

Although oneM2M ceases the collection pattern, parent-child relations are still facilitated in various ways, as the Table 4.1 shows. Hence, they are still used in the following for the considered scenarios of the oneM2M-based ASDP because it supports the understanding of resource relationships, even without collection resources. Figure 4.5 presents a resource tree structure example of a CSEBase. It shows the relationships and types of Attributes and Resources including their cardinality, whereas “(L)” states that the attribute itself contains a list. This example details an AE with a container and a contentInstance resource. The light grey attributes are for the purpose of CSE-internal resource management and will not be externally accessible.

Resource Type	Short Description	Child Resource Types (selection)	Parent Resource Types (selection)
accessControlPolicy	Stores a representation of privileges. It is associated with resources that shall be accessible to entities external to the Hosting CSE. It controls "who" is allowed to do "what" and the context in which it can be used for accessing resources	*subscription*	*AE, AEAnnc, remoteCSE, remoteCSEAnnc, CSEBase*
AE	Stores information about the AE. It is created as a result of successful registration of an AE with the Registrar CSE	*subscription, container, group, accessControlPolicy,*	*remoteCSE, remoteCSEAnnc, CSEBase*
container	Shares data instances among entities. Used as a mediator that buffers data exchanged between AEs and/or CSEs. The exchange of data between AEs (e.g. an AE on a Node in a field domain and the peer-AE on the infrastructure domain) is abstracted from the need to set up direct connections and allows for scenarios where both entities in the exchange do not have the same reachability schedule	*container, contentInstance, subscription, latest, oldest*	*AE, AEAnnc, container, containerAnnc, remoteCSE, remoteCESAnnc, CSEBase*
contentInstance	Represents a data instance in the *<container>* resource	*None specified*	*Container, containerAnnc*
CSEBase	The structural root for all the resources that are residing on a CSE. Stores information about the CSE itself	*remoteCSE, remoteCSEAnnc, node, AE, container, group, accessControlPolicy, subscription, schedule*	*None specified*
group	Stores information about resources of the same type that need to be addressed as a Group. Operations addressed to a Group resource shall be executed in a bulk mode for all members belonging to the Group	*subscription*	*AE, AEAnnc, remoteCSE, remoteCSEAnnc, CSEBase*
latest (V)	Virtual resource that points to most recently created *<contentInstance>* child resource within a *<container>* resource	*None specified*	*container*
node	Represents specific Node information	*subscription*	*CSEBase, remoteCSE*
oldest (V)	Virtual resource that points to first created *<contentInstance>* child resource within a *<container>* resource	*None specified*	*container*
remoteCSE	Represents a remote CSE for which there has been a registration procedure with the registrar CSE identified by the CSEBase resource	*AE, container, group, accessControlPolicy, subscription, schedule, node*	*CSEBase*
schedule	Contains scheduling information for delivery of messages	*subscription*	*subscription, CSEBase, remoteCSE*
subscription	Subscription resource represents the subscription information related to a resource. Such a resource shall be a child resource for the subscribe-to resource	*schedule*	*accessControlPolicy, accessControlPolicyAnnc, AE, AEAnnc, container, CSEBase, group, groupAnnc, node, nodeAnnc, remoteCSE, remoteCSEAnnc, schedule*

Table 4.1: Selection of oneM2M resource types (oneM2M TS-0001, 2015)

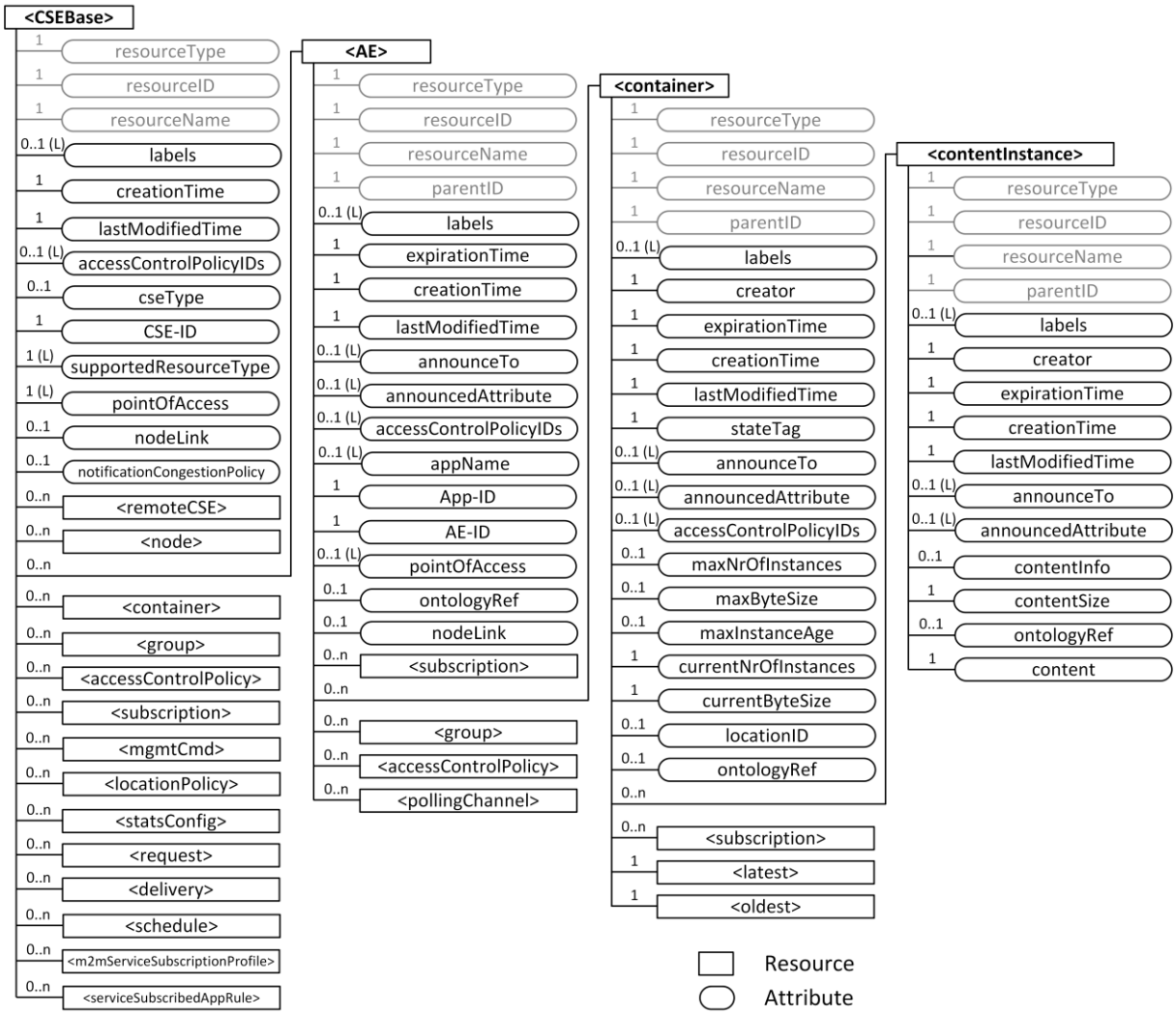

Figure 4.5: Resource tree structure example of a CSEBase

4.4.4 Methods

Communication between entities is possible across the respective reference points (see Section 4.3.2). This means, for example, AEs never communicate directly but by use of their CSE(s) which reflects the middleware approach of the CSE.

The oneM2M service platform applies a request/response communication pattern between the originator (AEs, or CSEs) and the receiver (CSEs, or AEs if they are server-capable). Each request can be one of the respectively supported method, namely: Create, Retrieve, Update, Delete, or Notify (in the following abbreviated as CRUD+N). Table 4.2 shows which of the CRUD+N operations are allowed with respect to the originator and receiver of the request.

An AE always sends the request to its registrar CSE. If this registrar CSE is not the hosting CSE of the requested resource, the receiving CSE gets a transit CSE and forwards the request to another CSE which might be another transit CSE or the hosting CSE. Thereby the AE knows

its registrar CSE through its own registration and the CSE knows their next CSE hops, e.g., through pre-provision during the bootstrap phase.

		Receiver	
	Supported Methods	AE	CSE
Originator	AE	**not supported**	Create Retrieve Update Delete
	CSE	**Notify**	Create Retrieve Update Delete **Notify**

Table 4.2: Supported operations between originator and receiver entity

To summarise: The resource tree is mapped to Uniform Resource Identifiers (URI) and exposed through the standardised interfaces (Mca, Mcc, Mcc'), following the RESTful architectural style. Accordingly, the resources are manipulated using CRUD+N operations, which are mapped to the applied application layer protocols, most likely HTTP, CoAP, and MQTT, see Section 4.4.5.

4.4.5 Protocol Stack and Bindings

The oneM2M standard specifies the CSE layer as a middleware on which the AE layer is placed (see Figure 4.2). The communication across the reference points Mca, Mcc, Mcc' is bound to certain application layer protocols (see Figure 4.6)[14]. This is the technical foundation for the fact that AEs and CSEs can be co-located in a single Node (i.e., ASN), or that AEs are also capable of residing on a different Node than their registrar CSE. Possible configurations are detailed in Section 4.3.6.

14 The Mcn reference point from the CSE to the NSE might also be bound to the named application layer protocols. But, there could be also other mappings in order to integrate the respective network APIs (oneM2M TS-0001, 2015, p. 35)

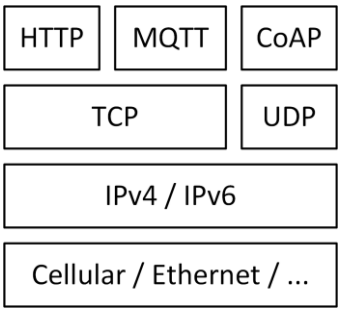

Figure 4.6: Current oneM2M protocol stack for Mca, Mcc, Mcc' reference points

The oneM2M standard currently provides bindings for the following application layer protocols:

- Hyper Text Transfer Protocol (HTTP) (oneM2M TS-0009, 2015)
- Constraint Application Protocol (CoAP) (oneM2M TS-0008, 2015)
- Message Queue Telemetry Transport (MQTT) (oneM2M TS-0010, 2015)

CoAP was especially designed for the RESTful communication of very limited electronics devices (Bormann, Castellani, & Shelby, 2012; Shelby, Hartke, & Bormann, 2013). It typically runs over User Datagram Protocol (UDP), which contributes to its simplicity (Bormann et al., 2012). In contrast, MQTT typically runs over Transmission Control Protocol (TCP), as well as HTTP. Hence, CoAP is more lightweight than MQTT, followed by HTTP that in comparison is heavyweight in comparison (Pereira & Aguiar, 2014).

Besides, MQTT and CoAP, in comparison to HTTP, intrinsically support asynchronous communication by means of the publish/subscribe communication pattern which could be considered as a benefit for M2M communication. These are only few aspects as to why CoAP and MQTT outperform the HTTP protocol in many M2M communication situations (Pereira & Aguiar, 2014).

However, regarding the appropriateness of the oneM2M service platform for the enabling of the distributed ASDP concept, the important aspect is that the protocol stack of the oneM2M standard provides bindings for classic HTTP as well as modern MQTT and CoAP that are more commonly dedicated to certain M2M scenarios. Due to the abstraction introduced by the CSE layer the application layer protocol applied between nodes can be changed transparently, which means without necessary modifications of the AEs. In this regard, the availability of those three protocols bindings contributes to the fulfilment of the REQ 6 of the ASDP concept.

4.5 Communication and Data Exchange Mechanisms

Adequate data exchange capabilities are crucial for the oneM2M service platform as middleware because it spans the design space for the communication of distributed AEs and CSEs. This design space directly relates to the resulting data flows and network requirements

of the distributed functionality, and hence is a central aspect for the REQ 6. Due to this fact, the communication and data exchange mechanisms decide which functional splits are practicable between the applications and nodes of the oneM2M service platform (see Section 3.3).

This section presents the currently available communication and data exchange mechanisms of the oneM2M service platform and assesses their spanned design space. The focuses are particularly those communication mechanisms relevant for the ASDP which expects AEs to be server-capable. Thus, the polling channel as a mechanism to deal with non-server-capable AEs is not considered here. Furthermore, the mechanisms are introduced without considering group communication capabilities (see related CSF, Section 4.3.2).

4.5.1 Principles

The oneM2M service platform is designed as cross-domain platform. Hence, the communication and data exchange mechanisms have to consider the fact that the nodes to be integrated are very heterogeneous with respect to their computational and functional capabilities. For example, in the Smart Home domain, quite limited nodes, such as temperature sensors ADNs, might be operated, possibly without integrated storage capabilities. They may provide measurements only at low frequency, e.g., one measurement per minute to an MN, which connects such sensors to the oneM2M service platform. Such a simple sensor, storing its data directly at a MN via existing in-house wireless technologies, may not require enhanced communication mechanisms such as indirect communication techniques or filter criteria, but may require mechanisms to deal with its missing server-capability of the AEs.

In comparison, within the context of an ITS, there might be nodes such as vehicles which nowadays offer great computational capabilities, several gigabytes of storage, and potentially could provide several hundred megabytes of sensor data per second (if optical sensors, such as cameras, are also taken into account). While the vehicle node is expected to be server-capable, it may require enhanced indirect communication capabilities to deal with the wireless cellular network characteristics and temporary disconnections. Furthermore, enhanced data filtering capabilities might be required because it is neither feasible nor reasonable to transfer such amounts of vehicular data using the available wireless cellular networks (see Section 2.2.2).

It shall be noted that data exchange is not limited to the exchange of application data. Since within the resource tree structure of a CSE much more than merely application data is stored, and because data exchange in general refers to the full or partial transmission of sub-trees or resources, it also includes all other parts, such as management and meta-data.

For the following explanations, a oneM2M configuration according to Figure 4.7 is assumed. It consists of a Vehicle ASN and an OEM Server ASN. Within this example, a $VehicleAE_1$ is located on the Vehicle ASN, and an $OEMAE_1$ is located on the OEM Server ASN.

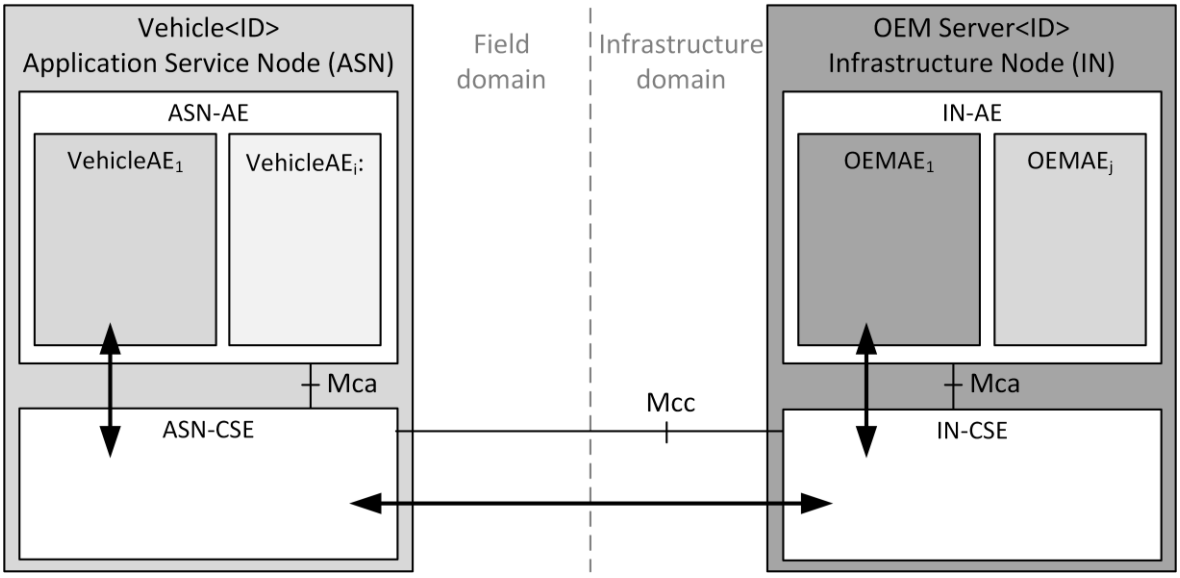

Figure 4.7: Generic oneM2M configuration for communication and data exchange considerations

Figure 4.7 also illustrates a fundamental characteristic of oneM2M communication and data exchange: AEs never communicate directly, but only by use of at least one CSE, even if the AEs are registered at the same CSE or located on the same node. In this regard, the data exchange between AEs refers to indirect communication that is defined as:

> *Indirect communication is defined as communication between entities in a distributed system through an intermediary with no direct coupling between the sender and the receiver(s). (Coulouris et al., 2012, p. 230)*

The actual degree of coupling or uncoupling between AEs depends on the communication mechanism used. The three possibilities, provided by the oneM2M service platform, namely request/response, announcement, and subscribe/notify, are subsequently discussed.

4.5.2 Request/Response

The basic communication mechanism is a request/response. It is available on each reference point. The originator of a request can choose between three different types for the response handling (oneM2M TS-0001, 2015, p. 54):

1. **blockingRequest**: This means that the response is only sent, if the receiver CSE has fully processed the request and is capable of responding with the result. The originator is blocked during the whole processing until the response is received.
2. **nonBlockingRequestSynch**: The receiver CSE is requested to accept the request and shall provide a reference to a resource where it stores the result of the request after completion of the request. Here, a response is already sent on request acceptance. The originator of the request is responsible to retrieve the result through a subsequent request to the referenced resource. If the result is not yet available, it is the

responsibility of the originator to try again later. If the result is available, it is provided within the response.

3. **nonBlockingRequestAsynch**: The receiver CSE is requested to accept the request and shall respond to this individually. After processing of the request and availability of the result, the receiver CSE provides the result through an asynchronous notification request to the originator. For this variant, the originator can optionally specify several notification targets with which the response can be directly distributed to other entities.

The different response types (particularly nonBlockingRequestSynch) can facilitate the minimisation of the communication time of the originator of the request which may have a positive effect on energy efficiency (oneM2M TS-0001, 2015).

4.5.3 Announcement

Another possible communication mechanism is the resource announcement. It can be utilised by an AE or CSE if the announcedTo attribute of a supported resource within the respective Create or Update method is configured accordingly, e.g., with the target(s). Since a CSE is the receiver of such a resource Create or Update, it is the responsibility of the CSE to announce the resource to the target as intended by the originator (oneM2M TS-0001, 2015). The announced resource "can have a limited set of attributes and a limited set of child resources from the original resource" (oneM2M TS-0001, 2015, p. 138). In more detail: The oneM2M standard specifies for each attribute of an announcable resource whether this attribute is:

- **Mandatory Announced (MA):** This attribute is announced from the original to the announcedTo resource(s) in any case.
- **Optionally Announced (OA):** This attribute is optionally announced, if it is explicitly specified within the announcedAttribute attribute at the original resource.
- **Not Announced (NA):** This attribute is never announced to the announcedTo resource(s).

The announcement of resources facilitates, in addition to others, the discovery of AEs on remote nodes, if they are announced with meaningful attributes, such as label or contentInfo to the superior CSE. Furthermore, the announcement also enables the data exchange between nodes if the announceTo and announcedAttribute attributes of the respective container and contentInstance resources are configured accordingly. However, the announcement forces an exact copy of the announced attributes of the original resource and does not enable additional filtering which is the reason why it is of limited relevance for application data exchange.

4.5.4 Subscribe/Notify

The subscribe/notify mechanism of the oneM2M service platform can be considered as an implementation of the publish/subscribe communication paradigm. Systems implementing this

paradigm can also be referred to as distributed event-based systems (Coulouris et al., 2012). In the oneM2M implementation, the CSE middleware layer acts as neutral dealer, bringing publisher and subscribers together. Intrinsic to the oneM2M implementation of the publish/subscribe communication paradigm is its distributed nature, since each CSE on every node has this neutral dealer capability. This puts efforts to the distributed discovery mechanism and routing capabilities to interconnect those distributed dealer capabilities.

Figure 4.8 shows an exemplary sequence diagram of a subscription and notification procedure, within the current oneM2M service platform. The example assumes the usage of a server-capable $OEMAE_1$.

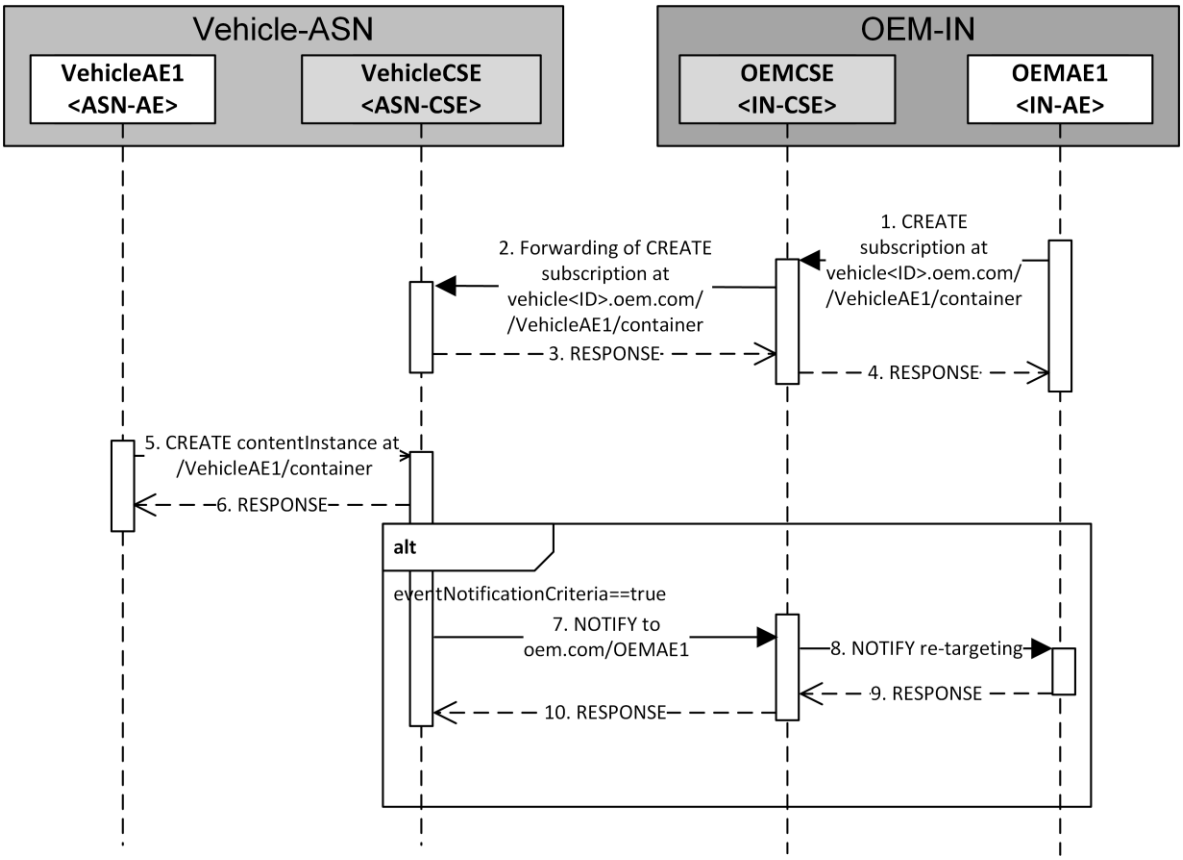

Figure 4.8: Sequence Diagram of Subscribe/Notify example

Step 1: The $OEMAE_1$ requests the creation of a subscription on the resource vehicle<ID>.oem.com/VehicleAE1/container. This CREATE request is sent to the OEMCSE, where the $OEMAE_1$ is registered.

Step 2: The OEMCSE detects that the subscription request targets a resource located within another CSE. It forwards the request to the VehicleCSE.

Step 3: The subscription request is accepted, and the subscription is created. Response OK.

Step 4: The Response OK is forwarded to the initiating $OEMAE_1$.

Step 5: The $VehicleAE_1$ publishes data through the creation of a contentInstance at /VehicleAE1/container.
Step 6: Response OK.
Step 7: Since subscriptions to the resource /VehicleAE1/container exist, the VehicleCSE checks whether the constraints, in particular eventNotficationCriteria, are fulfilled. If this is the case, a notification is sent to oem.com/$OEMAE_1$. Since the $OEMAE_1$ is registered at the OEMCSE, the VehicleCSE sends the NOTIFY to the VehicleCSE.
Step 8: The VehicleCSE re-targets the NOTIFY to the $VehicleAE_1$.
Step 9: The $VehicleAE_1$ acknowledges the correct receipt of the notification with Response OK. This is sent to the OEMCSE.
Step 10: The OEMCSE forwards the Response OK to the VehicleCSE.

The design of the publish/subscribe communication pattern, respectively its occurrence as subscribe/notify mechanism within oneM2M, offers advanced decoupling of sender and receiver(s), i.e., AEs (Aldred, van der Aalst, Dumas, & Hofstede, 2005; cf. Coulouris et al., 2012; Eugster, Felber, Guerraoui, & Kermarrec, 2003). In detail, subscribe/notify mechanism of oneM2M enables the following decoupling (Aldred et al., 2005; cf. Eugster et al., 2003):

- **Space Decoupling:** Subscribing AEs do not need to know the identity/URI of the actual producer of the data, but only the identity/URI of the providing resource that is hosted within the CSE. Similarly, producing AEs do not need to know the identify/URL of the actual receiver(s) AE(s). The producing AE only publishes its data to a (neutral) resource, acting as dealer. This is the hosting CSE of the producer which is then responsible for the forwarding (or notification) of the receiver(s).
  This space decoupling between subscribing and publishing AE is achieved through the "subscription and notification CSF" which checks for existing subscriptions to that resource. It further checks the fulfilment of individual notification constraints, and afterwards triggers notification to the subscriber AE(s). This also enables a one-to-many relation, where one resource modification could be notified to many subscribers. The space decoupling also exists for the subscribing AE: They do not know which (and how many) AE(s) published data to the subscribed CSE resource hosted within the CSE.
  Example: Figure 4.8 steps 1-4 illustrate that the subscribing AE ($OEMAE_1$) addresses its subscription towards a container resource, located inside the VehicleCSE. This is independent of the $VehicleAE_1$, publishing its data (through a CREATE of a contentInstance) at the same VehicleCSE resource: see Figure 4.8 steps 5-6.
- **Time Decoupling:** AEs publish their data to a related resource within the CSE layer. Since the notification to subscribing AE(s) is performed through the "subscription and notification CSF", the originating AE is time-decoupled from the subscribing AE(s). This means that both AEs do not need to be online at the same time. In the usual case

where the originating AE and the subscribing AE(s) are hosted on different nodes, the node lifecycle and availability also get decoupled. The time-decoupling capabilities are even greater, if the actual oneM2M configuration includes an MN – then, even the CSEs (and hence complete Nodes) are time-decoupled.
Example: In the example illustrated in Figure 4.8, the $VehicleAE_1$ must only be online for the steps 5 and 6. The subsequent notification (steps 7-10) is time-decoupled. Depending on the CMDH policies, it could happen that these steps are performed hours later if the OEM-IN is not available at the time when the notification shall be send.

- **Synchronisation Decoupling:** The AE only publishes its data to the related resource within the CSE layer which is individually acknowledged. The originating AE is not part of any subsequent, asynchronously triggered notification(s) to subscribing AE(s). Hence, the originating AE is not blocked during that notification(s). Furthermore, the subscribing AE(s) is/are not (necessarily) blocked until it/they get(s) a notification, as the contact URI could be used as an asynchronous call-back. Therefore, the originating AE is synchronisation decoupled from the receiving AE(s).
  Example: Figure 4.8, steps 5 and 6 show the individual acknowledgement of the data provision of the $VehicleAE_1$ to the CSE. With the exception of these steps, the $VehicleAE_1$ is not active or blocked. Furthermore, the OEMAE1 is not active/blocked between the steps 4 and 8, and it directly respond in the following step 9 and afterwards again must not be active and is not blocked.

The subscribe/notify (i.e. publish/subscribe) communication mechanism is particularly suitable to deal with the requirements of large-scale M2M communications (Eugster et al., 2003; Uckelmann, Harrison, & Michahelles, 2011). Furthermore, the subscribe/notify contributes to interoperability and hence REQ 5 (Bass et al., 2012; Kazman et al., 2013). Hence, it is particularly suitable for application data exchange.

4.6 The oneM2M-based Automotive Service Delivery Platform

Similarities within the problem space of M2M and the ASDP concept motivated the considerations as M2M as initial hypothesis. The previous analysis of the oneM2M service platform by means of its functional architecture, service/resource model and technologies, and general communication and data exchange mechanisms showed suitable fundamental architectural design decisions. These contribute to the previously identified requirements of the ASDP. In this regard the investigation of the solution space provided by the oneM2M service platform supported its selection as enabler for the realisation of the ASDP concept.

However, the oneM2M service platform is a generic reference architecture. Hence, in the following section a reference configuration for automotive environments by means of a oneM2M-based ASDP is introduced. The term reference configuration thereby emphasises that the oneM2M-based ASDP is a selection and instantiation of appropriate subsets in the sense of

the ASDP concept, but it remains fully compatible with the oneM2M service platform. Since this reference configuration further constrains the solution space provided by the oneM2M service platform towards the requirements of our ASDP concept, it continues the ideas of the PBD methodology (see Section 2.4.5).

Despite the general suitability of the fundamental architectural design decisions of oneM2M, particularly with respect to REQ 6 of the ASDP concept, the communication and data exchange capabilities of oneM2M needs to be further investigated beyond the already introduced fundamentals. As a starting point, the Section 4.6.2 introduces basic communication scenarios of the oneM2M-based ASDP.

4.6.1 Reference Configuration

Following the oneM2M service platform specifications, a vehicle could be an ASN or a MN that integrates all vehicle-internal ASNs or it could be a NoDN (see Section 4.3.5). Considering that in a first deployment phase, the vehicular oneM2M endpoint might be a communication gateway, the head-unit, or in general one single On-Board Unit, which hides vehicle-internal non-oneM2M-compliant components, it has been decided that the vehicle externally appears and works as a oneM2M-compliant ASN, called vehicle ASN. Further, the OEM server is being realised as IN. The ASN is located inside the field domain and it is connected to the infrastructure domain, using wireless cellular network technologies. Since oneM2M-compliant third-party servers might themselves combine and abstract a multitude of nodes (e.g., if the third party is another OEM, a traffic infrastructure provider, or an energy company), the reference configuration assumes another IN, called 3rd party IN. Third party servers and applications that are not oneM2M-compliant could be referred to as NoDN. However, because the integration of NoDN is currently out of scope of oneM2M standardisation, the integration of third party servers using other technologies are integrated at AE level through appropriate adaptor AEs (e.g. IPE, see Section 4.3.5) residing on the OEM server IN.

Figure 4.9 illustrates the compound functional architecture of the reference configuration of a oneM2M-based ASDP with the following reference points: Mca (vertical interface for AEs), Mcc (horizontal between two CSEs). OneM2M-compliant third party server INs are connected by use of the Mcc' interface with the OEM server IN. This indicates that the OEM might restrict the external functionalities provided to third parties. The integration of non-compliant 3rd Party Server is an individual task out of scope of standardised reference points.

For the sake of completeness, the possibility of a direct connection between the vehicle and other M2M-conformant IN or MN, e.g., third party servers, via the Mcc interface is hinted at. Against the background of the envisaged mediation capabilities between a vehicle ASN and 3rd party server IN, the ASDP concept describes the value of an interposed OEM server (see Section 3.1). Hence, the direct connection to third party servers is currently not favoured although technically – because of similar reference points – feasible. Within a real-world

deployment there might be additional MN introduced between ASN and IN, e.g., for optimisation purposes by means of deeper hierarchies. This could be done transparently and needs no further consideration here.

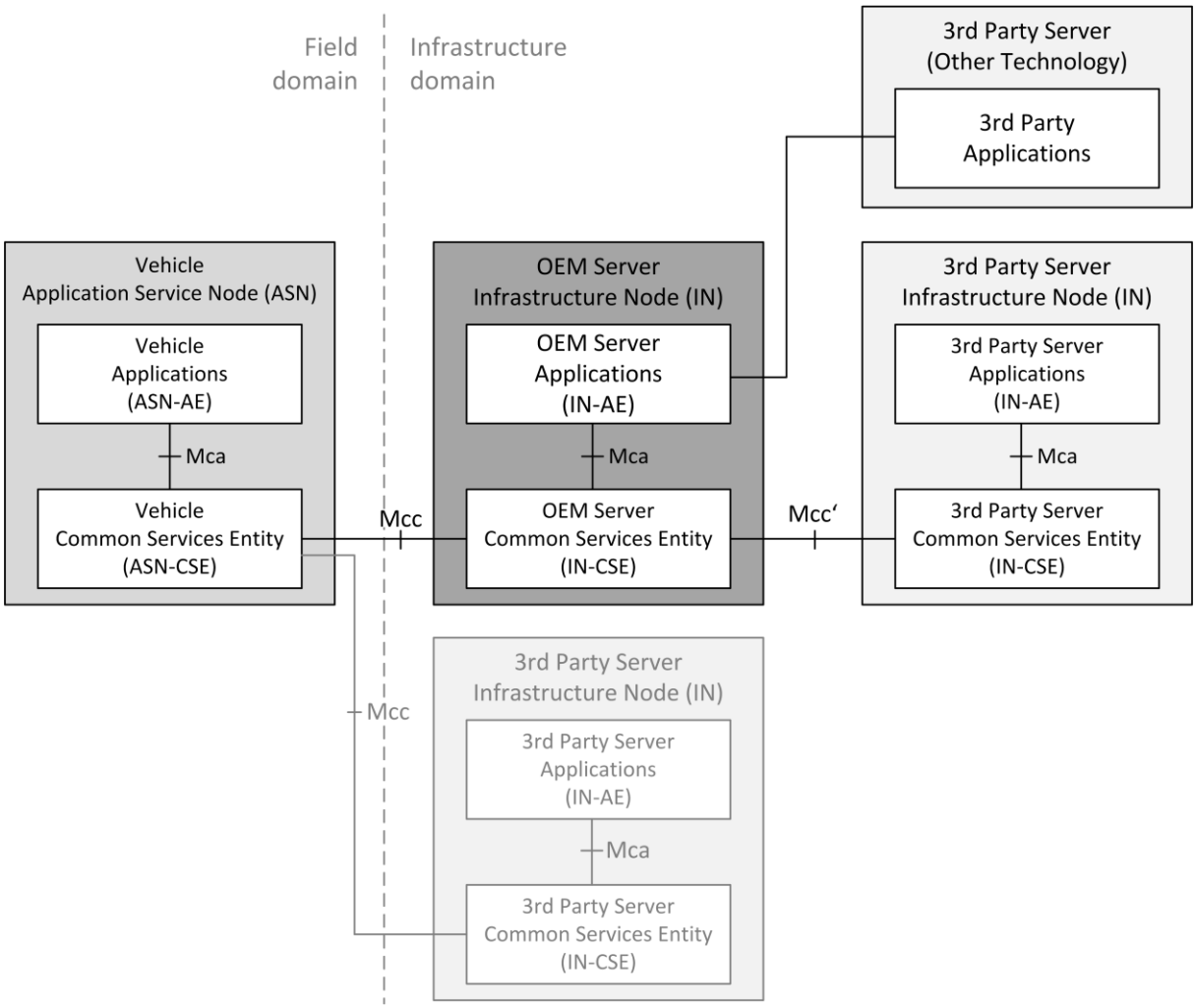

Figure 4.9: Functional architecture of the reference configuration of a oneM2M-based Automotive Service Delivery Platform

4.6.2 Basic Communication Scenarios

Considering the reference configuration of oneM2M for the ASDP two basic communication scenarios can be identified whenever communication across at least one intermediate node is considered. This is the case, e.g., for the communication of vehicle AEs with 3rd party AEs, or if in a real-world deployment additional MNs are integrated. Thereby, the intermediate Node (e.g., the OEM-IN) can either occur transparent or with back-to-back functionalities realised through one or more AEs. These communication scenarios are introduced in the following with examples utilising the subscribe/notify mechanism.

Transparent Intermediary Node

The oneM2M service platform meet the REQ 1 of the ASDP concept and offers a standardised End-to-End solution. Moreover, the horizontal Mcc interface occurs between ASNs, MN, and IN, and also the Mcc' interface between INs of two different Service Providers (see Section 4.3.2) must not necessarily differ from the Mcc interface, depending on the service contract. Accordingly, from the functional point of view, CSEs and respectively AEs residing on these nodes are capable of direct interworking. Nevertheless, the actual deployment configuration might introduce intermediary nodes. OneM2M uses a re-targeting mechanism to forward requests (and responses) that are not addressing a local resource to another CSE, according to pre-configuration. This re-targeting is transparent to the actual request (and responses).

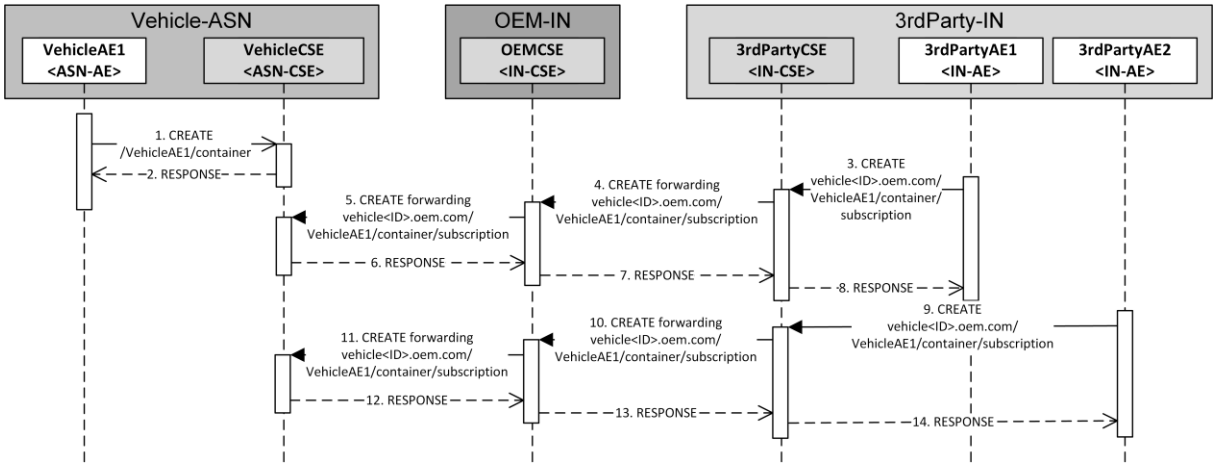

Figure 4.10: Sequence diagram of subscription setup within transparent intermediary node scenario

Considering a configuration with a Vehicle-ASN, hosting a $VehicleAE_1$, an OEM-IN, and a 3rd Party-IN, hosting two applications 3rdPartyAE1 and 3rdPartyAE2 that want to subscribe to the data container of $VehicleAE_1$. Hereby, Figure 4.10 illustrates the sequence diagram for the message flow of the subscription setup:

Step 1: $VehicleAE_1$ requests creation data container

Step 2: Response OK

Step 3: 3rdPartyAE1 requests creation of subscription at vehicle<ID>.oem.com/VehicleAE1/container/subscription

Step 4: The 3rdPartyCSE determines that the requested resource is not part of the local CSE and re-targets the request to the OEMCSE

Step 5: The OEMCSE determines that the requested resource is not part of the local CSE and re-targets the request to the VehicleCSE with the address vehicle<ID>.oem.com. The VehicleCSE determines that the requested resource is local, and performs the create request on the VehicleAE1/container/subscription resource

Step 6: Response OK to OEMCSE

Step 7: Re-targeting of response OK to 3rdPartyCSE
Step 8: Re-targeting of response OK to 3rdPartyAE1
Step 9: 3rdPartyAE2 requests creation of subscription at vehicle<ID>.oem.com/VehicleAE1/container/subscription
Step 10: The 3rdPartyCSE determines that the requested resource is not part of the local CSE and re-targets the request to the OEMCSE
Step 11: The OEMCSE determines that the requested resource is not part of the local CSE and re-targets the request to the VehicleCSE with the address vehicle<ID>.oem.com. The VehicleCSE determines that the requested resource is local, and performs the create request on the VehicleAE1/container/subscription resource
Step 12: Response OK to OEMCSE
Step 13: Re-targeting of response OK to 3rdPartyCSE
Step 14: Re-targeting of response OK to 3rdPartyAE1

Afterwards, two subscriptions exist for the VehicleAE$_1$ data container. Figure 4.11 shows the sequence diagram for the related notifications.

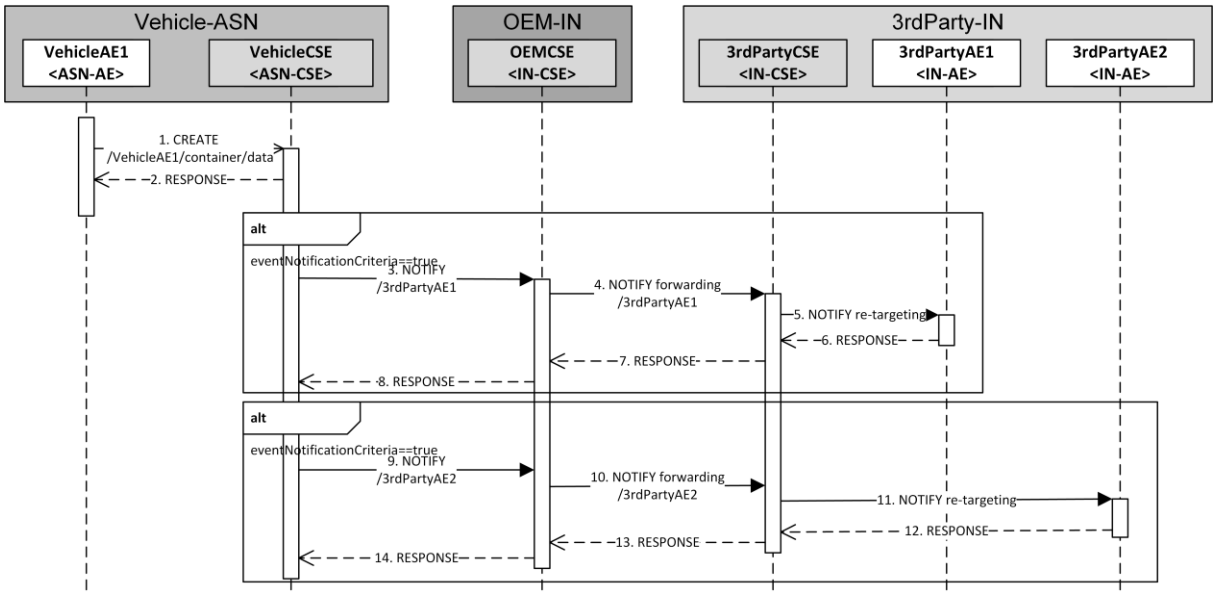

Figure 4.11: Sequence diagram of notification within transparent intermediary node scenario

Step 1: VehicleAE1 requests creation of data within its container
Step 2: Response OK
Step 3: In case that the filterCriteria of the Notification (according to the subscription of 3rdPartyAE1) are fulfilled, notify request to 3rdParty.com/3rdPartyAE1 is performed. Because this target is not at this originating CSE, it is forwarded to the OEMCSE.
Step 4: The OEMCSE determines that the notification target is not a local resource and forwards the request to the 3rdPartyCSE.

Step 5: The 3rdPartyCSE performs re-targeting of the notify request to the 3rdPartyAE1
Step 6: Response OK to 3rdPartyCSE.
Step 7: 3rdPartyCSE forwards the response to the OEMCSE
Step 8: OEMCSE forwards the response to the VehicleCSE
Step 9: In case that the filterCriteria of the Notification (according to the subscription of 3rdPartyAE2) are fulfilled, notify request to 3rdParty.com/3rdPartyAE2 is performed. Because this target is not at this originating CSE, it is forwarded to the OEMCSE.
Step 10: The OEMCSE determines that the notification target is not a local resource and forwards the request to the 3rdPartyCSE.
Step 11: The 3rdPartyCSE performs re-targeting of the notify request to the 3rdPartyAE2
Step 12: Response OK to 3rdPartyCSE.
Step 13: 3rdPartyCSE forwards the response to the OEMCSE
Step 14: OEMCSE forwards the response to the VehicleCSE

The transparent forwarding of requests (and responses) is a fundamental functionality of the oneM2M service platform that is necessary to enable hierarchical structures (see Section 4.3.6). However, this scenario of a transparent intermediary OEM-IN has several drawbacks: At first, the transparent forwarding of subscriptions and notifications potentially causes redundant data transmissions from the Vehicle-ASN over the OEM-IN to the 3rdParty-IN. Although redundant data transmissions should be avoided at any time, they are particularly negative between the Vehicle-ASN and the OEM-IN, since this will utilise wireless cellular network bandwidth. Furthermore, considering privacy and mediation capabilities, the OEM might not favour the fact that 3rdPartyAEs can directly connect to the vehicle. Summarised, the fully transparent connection of VehicleAEs with 3rdPartyAEs is not considered as a favoured communication scenario for the ASDP. An alternative is described in the next paragraph.

Intermediary Node with Back-to-Back AE

In contrast to the transparent intermediate node scenario, the communication can be limited to a certain intermediate node, e.g., an OEM server IN. In this regard, an AE that realises a back-to-back functionality can be meaningful.

The foundations are different views on its resources (including AEs) that can be provided by an OEM Server IN. These can, e.g., depend on the role of the communication counterpart to distinguish between an internal view/access for the vehicle ASNs of the OEM and an external view/access towards 3rd Party Nodes. This can be realised by means of oneM2M accessControlPolicies, which support constraints according to the originator, the operations (i.e., Create, Retrieve, Update, Delete, Discovery, Notify), and contexts, such as time, address or location (oneM2M TS-0001, 2015; oneM2M TS-0003, 2015).

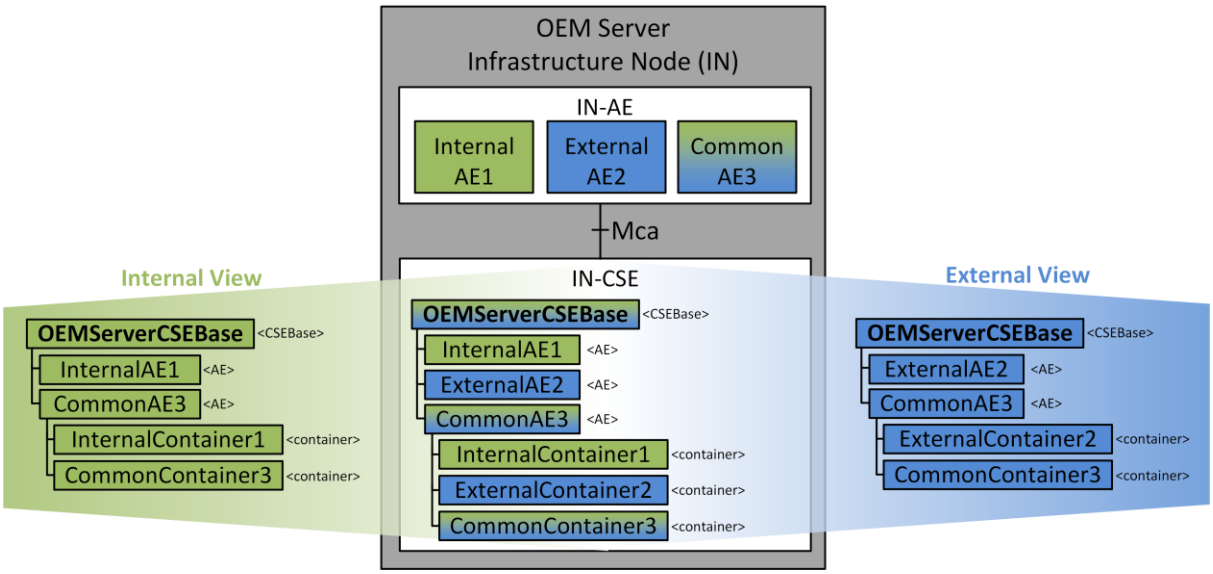

Figure 4.12: oneM2M Node with back-to-back functionality: Realisation of different views on the same node

Figure 4.12 illustrates the use of those accessControlPolicies to realise an internal and external view of the OEM Server IN: Here, the InternalAE1 should only be available for internal usage, the ExternalAE2 should only be available for external usage and the CommonAE3 should be available to both. CommonAE3 further shows that access constraints are possible at different levels of the hierarchical structure and for different resources. Accordingly, the InternalContainer1 of the CommonAE3 is restricted to internal usage, the ExternalContainer2 is restricted to external usage, and the CommonContainer3 is visible and modifiable for both. AEs, such as the CommonAE3, which are accessible for two or more counterparts but also contain resources that are dedicated to one (or a subset of the overall counterparts), can be used to realise the envisaged back-to-back functionalities. For example, data that is received at the InternalContainer1 can be processed at the CommonAE3 and provided differently at the ExternalContainer2.

Figure 4.13 shows the sequence diagram of the subscription setup according to a back-to-back intermediary node scenario.

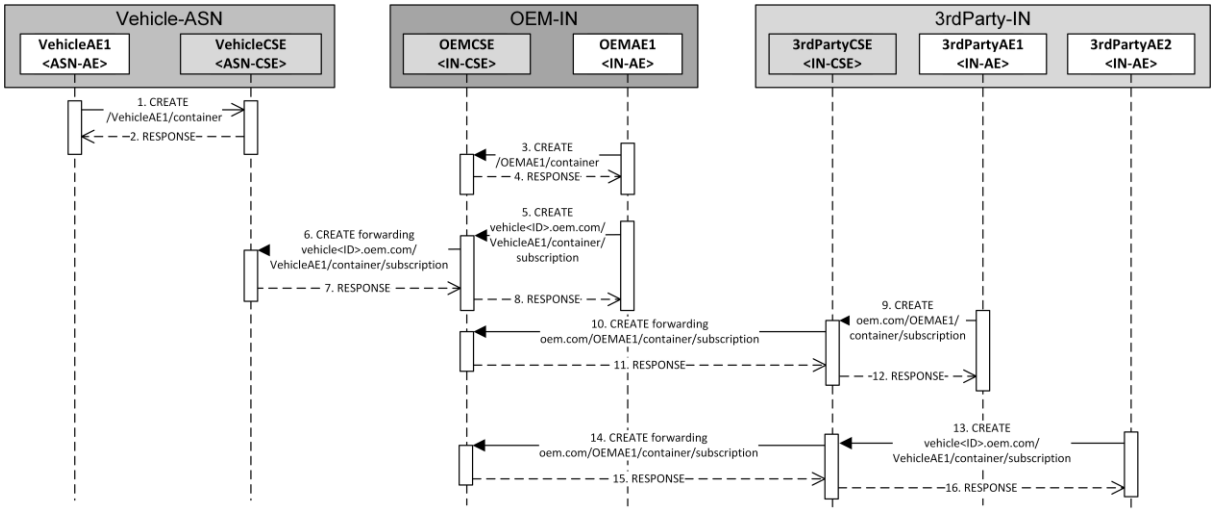

Figure 4.13: Sequence diagram of subscription setup within back-to-back intermediary node scenario

Step 1: VehicleAE1 requests the creation data container.
Step 2: Response created.
Step 3: $OEMAE_1$, which is the AE with back-to-back functionality, requests the creation of a data container that is accessible for 3rdPartyAEs.
Step 4: Response created.
Step 5: $OEMAE_1$ requests creation of subscription at vehicle<ID>.oem.com/VehicleAE1/container/subscription.
Step 6: The OEMCSE determines that the requested resource is not part of the originating CSE and forwards the request to the VehicleCSE.
Step 7: Response created to OEMCSE.
Step 8: Response created re-targeting to $OEMAE_1$.
Step 9: 3rdPartyAE1 requests creation of subscription at oem.com/OEMAE1/container/subscription.
Step 10: The 3rdPartyCSE determines that the requested resource is not part of the originating CSE and forwards the request to the OEMCSE.
Step 11: Response created to 3rdPartyCSE.
Step 12: Response created re-targeting to 3rdPartyAE1.
Step 13: 3rdPartyAE2 requests creation of subscription at oem.com/OEMAE1/container/subscription.
Step 14: The 3rdPartyCSE determines that the requested resource is not part of the originating CSE and forwards the request to the OEMCSE.
Step 15: Response created to 3rdPartyCSE.
Step 16: Response created re-targeting to 3rdPartyAE2.

This setup facilitates the following message flow, as illustrated in Figure 4.14:

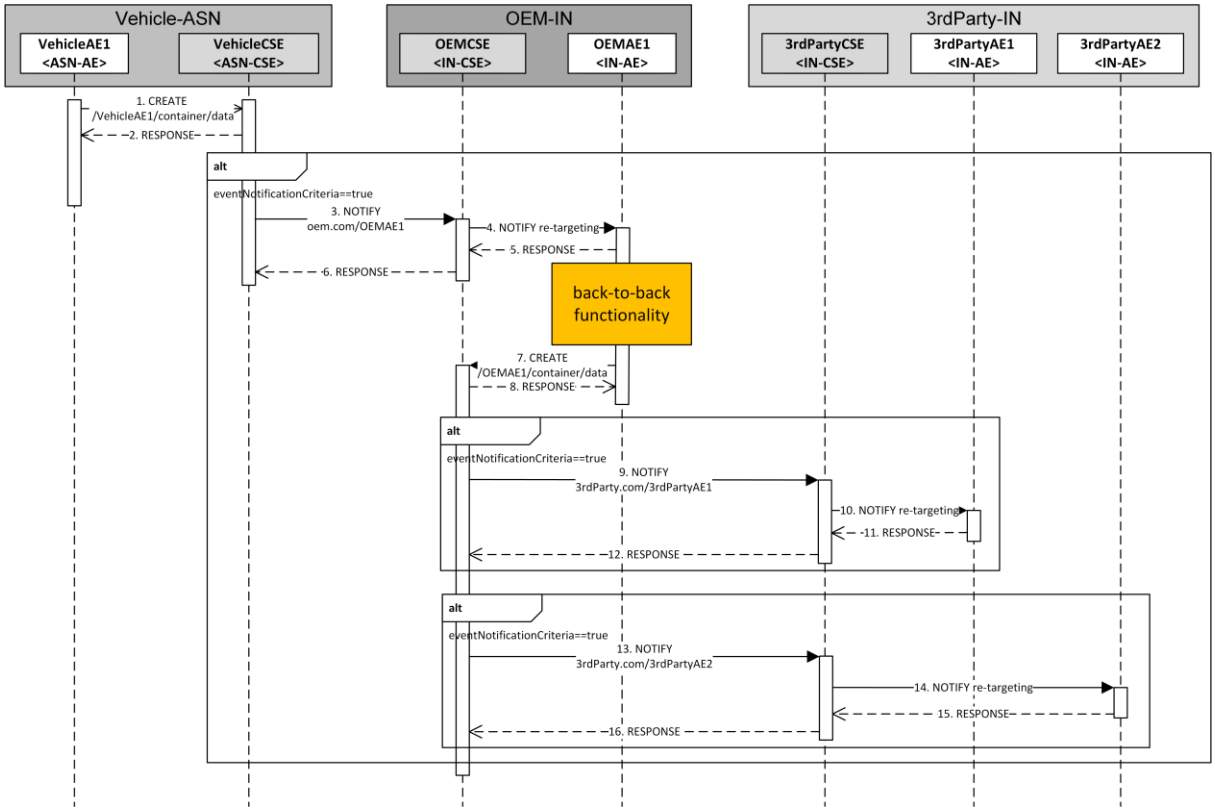

Figure 4.14: Sequence diagram of notification within back-to-back intermediary node scenario

Step 1: VehicleAE1 requests creation of data within its container

Step 2: Response OK

Step 3: In case that the filterCriteria of the notification (according to the subscription of $OEMAE_1$) are fulfilled, notify request to oem.com/OEMAE1 is performed. Because this target is not at this originating CSE, it is forwarded to the OEMCSE.

Step 4: The OEMCSE performs re-targeting of the notify request to the OEMAE1

Step 5: Response OK to OEMCSE.

Step 6: OEMCSE forwards the response to the VehicleCSE

Step 7: The $OEMAE_1$ stores the notification data (i.e., content of data container) within its container through a create request of /OEMAE1/container/data.

Step 8: Response created to $OEMAE_1$.

Step 9: In case that the filterCriteria of the Notification (according to the subscription of 3rdPartyAE1) are fulfilled, notify request to 3rdParty.com/3rdPartyAE1 is performed. Because this target is not at this originating CSE, it is forwarded to the 3rdPartyCSE.

Step 10: The 3rdPartyCSE performs re-targeting of the notify request to the 3rdPartyAE1

Step 11: Response OK to 3rdPartyCSE.

Step 12: 3rdPartyCSE forwards the response to the OEMCSE

Step 13: In case that the filterCriteria of the Notification (according to the subscription of 3rdPartyAE2) are fulfilled, notify request to 3rdParty.com/3rdPartyAE2 is performed. Because this target is not at this originating CSE, it is forwarded to the 3rdPartyCSE.
Step 14: The 3rdPartyCSE performs re-targeting of the notify request to the 3rdPartyAE2
Step 15: Response OK to 3rdPartyCSE.
Step 16: 3rdPartyCSE forwards the response to the OEMCSE

The usage of AEs with back-to-back functionalities can be a meaningful communication scenario for certain situations. It can be used for the purpose of information hiding, or to increase privacy and mediation capabilities. However, it requires the deployment of the respective AE at the intermediate node, which might not be possible or reasonable in any case (e.g., with respect to ownership or operation of the intermediate node). Furthermore, the indirection that occurs through the multiple transfers of data between the AE that realises the back-to-back functionality and its CSE is inefficient.

Considering this, the communication mechanisms of oneM2M particularly with regard to data exchange between AEs should be considered in more detail.

4.7 Summary

The oneM2M standard describes a domain-independent and universal service platform for M2M communication. It introduces a common middleware layer (i.e., CSE) that implements functionalities, referred to as Common Service Functions (CSF), that are shared across applications (i.e., AE). Its End-2-End approach includes various node types and supports hierarchical network configurations. Although architectural design decisions can be identified that pose minor deviation on related principles, oneM2M can be considered as following a RESTful Service-Oriented Architecture approach. The available protocol binding that in addition to HTTP also support CoAP and MQTT, in combination with the available communication mechanisms including the subscribe/notify mechanism, provide meaningful capabilities for the requirements of M2M communications as well as the ASDP concept.

In this regard, the analysis of the key architectural design decisions of oneM2M service platform confirmed its general suitability as enabler for the concept of a distributed Automotive Service Delivery Platform. Hence, the initial hypothesis that was based on similarities of the related problem spaces could be confirmed through the investigation of the provided solution space. Consequently, a reference configuration of oneM2M to realise the distributed Automotive Service Delivery Platform concept has been described. On this basis, basic communication mechanisms have been discussed which are a starting point for more detailed investigation of communication mechanisms and data exchange capabilities facilitated by oneM2M.

However, particularly with respect to the considerations about the distribution and portability of functionalities between the vehicle and backend, and towards an Automotive Embedded Internet, a more detailed consideration of the existing data exchange capabilities is necessary to enable final assessments about the appropriateness of oneM2M as enabler for the ASDP concept. This analysis is performed in the following chapter.

5 Analysis of Current Data Exchange Capabilities of the oneM2M Service Platform

The previous chapter introduced the oneM2M service platform and showed its general suitability as enabler for the distributed ASDP concept through the analysis of architectural design decisions and trade-offs being made against the background of the prior identified requirements of the concept (see Section 3.4). Afterwards, it introduced a reference configuration of a oneM2M-based ASDP for automotive environments. Thereby, the detailed analysis of the current data exchange capabilities of the oneM2M service platform has been motived.

One beneficial architectural design decision is the uniform Mca interface between the CSE and AEs. From the functional point of view this facilitates the movement of AEs between Nodes, e.g., between the vehicle and the OEM server, hence contributes to the REQ 3. Nevertheless, the placements and functional splits which are actually feasible between AEs and Nodes are limited by the resulting requirements against the intermediate networks. Hence, in the context of an ASDP, the capabilities of the wireless cellular network (see Section 2.2.2) constrain the design space for functional splits and distributions of AEs between the vehicle, the OEM server, and other nodes. In more detail, the available maximum bandwidth between Nodes, minimum latency, potential jitter, and error rate, are initially constrained by the cellular network technology. Moreover, upper layer protocols, i.e., IP, TCP, HTTP (see Section 4.4.5) in general introduce additional overhead, which could be expected to further diminish the actually achievable communication performance at application layer, hence between AEs. On the highest oneM2M layer, the AEs provide consuming and processing application data according to their use case. In this regard, the distributed AEs introduce requirements against the communication layers that are derived from the (distributed) use case to be realised. If these requirements already exceed the capabilities of the underlying network technologies and protocols, the intended distribution or functional split is not feasible, independent of the oneM2M service platform capabilities.

However, AEs never exchange data directly, but by use of the CSE middleware. Hence, if the requirements derived from the distributed use case in theory could be realised by the underlying communication networks and protocols, it is a matter of the oneM2M data exchange capabilities if the intended distribution or functional split is actually feasible. The data exchange capabilities of the oneM2M service platform further affect the network efficiency of the distributed

application, which has implications on scalability and costs. Since the oneM2M data exchange capabilities directly influence its applicability for and the potential of a oneM2M-based ASDP, they are analysed in more detail in the following.

Towards the detailed analysis of current oneM2M data exchange capabilities, this chapter first introduces a condensed ASDP scenario. Secondly, principle considerations with respect to data exchange, data subsets and filtering positions are presented. Afterwards, the existing data exchange capabilities of the subscribe/notify mechanism, which is the most advanced communication mechanism provided, are comprehensively analysed. Finally, existing shortcomings of oneM2M data exchange and their consequences are revealed, using the prior introduced condensed scenario.

5.1 Condensed ASDP Scenario

To analyse the data exchange capabilities of the oneM2M service platform for application data, the following condensed ASDP scenario is used. It is derived from the scenarios that were introduced and discussed in Section 3.2.

5.1.1 Node Configuration

This ASDP scenario considers one vehicle (realised as ASN) and one OEMServer (realised as IN) which reflect the proposed reference configuration of the oneM2M service platform for the realisation of the ASDP concept (see Section 4.6).

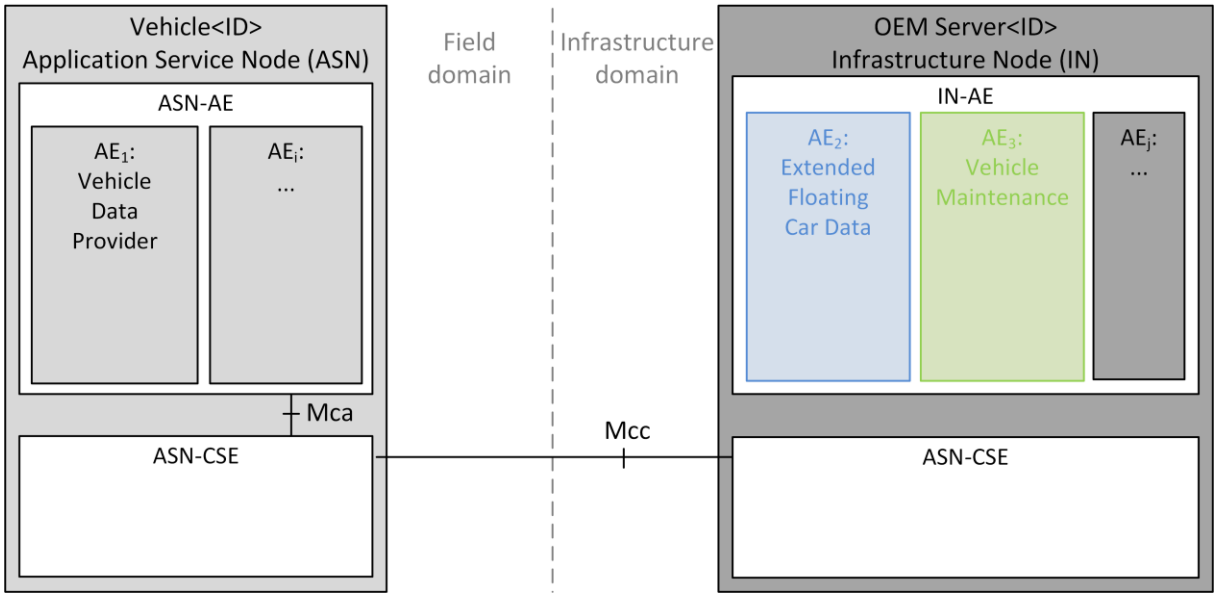

Figure 5.1: Condensed ASDP scenario

5.1.2 Application Entities

In this scenario, the ASN hosts one AE, namely AE_1 'Vehicle Data Provider'. The IN hosts two AEs, namely AE_2 'Extended Floating Car Data' and AE_3 'Vehicle Maintenance'. Their functionalities and use cases are introduced below.

AE1: Vehicle Data Provider

Since vehicular sensor data is the foundation for many automotive-related applications, a generic AE_1 'Vehicle Data Provider' is integrated into the vehicle that facilitates the provision of vehicular data to the oneM2M service platform.

To obtain the vehicular data, the AE_1 is connected with the in-vehicle communication networks, such as the CAN-bus (see Section 2.2.2). These proprietary in-vehicle connections are currently out of scope of the oneM2M service platform and standardisation activities. The AE_1 is capable of reading the proprietary data formats of the in-vehicle networks and can transform them to a universal and vendor-independent data format according to the oneM2M necessities. In this regard, the AE_1 can also be considered as a hybrid Inter-working Proxy Application Entity (IPE) (oneM2M TS-0001, 2015, p. 310) containing both – a oneM2M-compliant Mca reference point as well as a proprietary (non-oneM2M reference point) to the in-vehicle networks. However, since the focus here is on the vehicle-to-backend part of the ASDP, the distinction between an AE and an IPE is from minor importance. Hence, the 'Vehicle Data Provider' is treated as a usual AE, hiding the proprietary vehicle internals.

Since the AE_1 'Vehicle Data Provider' is a generic data provider, it could provide manifold data within several different data containers in the resource tree. However, this scenario focuses one exemplary data format, providing the following vehicular data within the attribute content of contentInstance resources of the container 'VehicleData':

- The vehicle position, determined through the values 'latitude' and 'longitude', including the vehicular driving direction, provided by the 'heading' value.
- The vehicle driving speed ('speed' value) in km/h.
- The remaining fuel range in km, as calculated by the vehicle, e.g., through usage of a (weighted) moving average of the fuel consumption and remaining fuel level. This data is provided through the value 'fuelRange'.
- The information as to whether a driving dynamic control intervention is currently applied or not, represented through a Boolean 'ESP' value (i.e., true or false).

This vehicular data is provided with the content attribute by use of an Extensible Markup Language (XML) document, including a reference to a XML Schema Definition (XSD) that provides detailed specification of XML structure and data types. The following listing shows an example of an XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<VehicleData xmlns="http://oem.com/xml/VehicleData">
  <Position>
    <latitude>49.866208</latitude>
    <longitude>8.640403</longitude>
    <heading>90</heading>
  </Position>
  <speed>35</speed>
  <fuelRange>96</fuelRange>
  <ESP>true</ESP>
</VehicleData>
```

Within the automotive context as well as most other typical M2M-scenarios, fundamental data available at a node or AE typically relates to vehicle-internal sensors measurements and computations. Hence, the data provided by such an application with respect to its accuracy and resolution initially reflects the physical capabilities of the sensors behind them. Position data is usually determined using an internal Global Positioning System (GPS) receiver connected to the CAN-bus with a typical update rate of 1 to 4 Hz, while other sensor values might be available at a much higher rate. Other sensor values might be available at a much higher rate, particularly safety-relevant values such as wheel rotational speeds, driving dynamic interventions, etc. Others, such as fuel level might be provided less frequently. Since the AE1 is unaware of data resolution requirements of (future) AEs within the ASDP, it provides the values with undiminished resolution. In any case, since the Vehicle Data Provider targets universality, the resolution cannot be tailored to one specific consuming AE. For this condensed scenario, it is assumed that the AE1 provides a new contentInstance to the VehicleData container every one to two seconds.

AE2: Extended Floating Car Data

One of the two AEs within the scenario hosted on the IN, is the AE_2 'Extended Floating Car Data'. It implements selected, related application logic (see Section 3.2.1). Considering the data provided by the AE_1, the AE_2 can, among other things, beneficially utilise the following vehicular data:

- To enable inference concerning the current traffic situation, it acquires the position (latitude, longitude, heading) and may also determine the speed of the vehicle (together with the respective timestamp) at reasonable intervals, which, e.g., might be time-related or distance-related.
- To detect the occurrence of dangerous traffic and road situations, the AE_2, among other things, can utilise the 'ESP' value. To locate the dangerous situation, the position (latitude, longitude, heading) and perhaps as well the speed of the vehicle and the timestamp of occurrence could be acquired.

AE3: Vehicle Maintenance

The second AE, which is hosted at the IN, is the AE_3 'Vehicle Maintenance'. It implements selected aspects of the respective application logic (see Section 3.2.2). In this condensed scenario, the value 'fuelRange' is of interest to initiate subsequent use cases, such as the offering of nearby gas stations when the remaining distance falls below a certain threshold. However, such subsequent use cases are not part of this condensed scenario. The focus here is on the acquisition of the 'fuelRange' according to suitable policies.

5.2 Principles

AEs within an M2M scenario typically provide data, reflecting the capabilities of their source, such as its sensor or computational capabilities. This does not necessarily reflect the data requirements of consuming AEs, which may come from a different vendor or domain.

In the previously introduced ASDP scenario, the AE_1 is a general vehicle data provider that basically reflects the capabilities of the vehicular sensors. Thus, consuming AEs might only need a subset of the overall data provided according to their to be realised functionality.

In preparation for detailed analysis of the oneM2M data exchange capabilities, this Section details such principles, considering filter requirements in contrast to filter capabilities. It further investigates possible filter positions within the data flow and discusses their appropriateness.

5.2.1 Filter Requirements vs. Capabilities

With respect to the data exchange between AEs, it is a matter of the filtering capabilities, as to how precisely the data being transferred can be tailored to the necessary data subset required according to the AE functionality. Since in general, within an M2M/IoT context, it could be assumed that the nodes or AEs are from different vendors or domains, and oneM2M as application and domain independent horizontal integration platform targets universality, so that filtering capabilities might not exactly match the requirements of the AEs.

Figure 5.2 illustrates such typical circumstances with respect to the data set provided by AE_1 and the subsets required by AE_2 and AE_3. It further indicates that the data subsets required by different AEs are not necessarily disjoined but might have overlaps, such as the left two blue and green rectangles which are intended to illustrate the (nearly) exact overlaps of a required data subset.

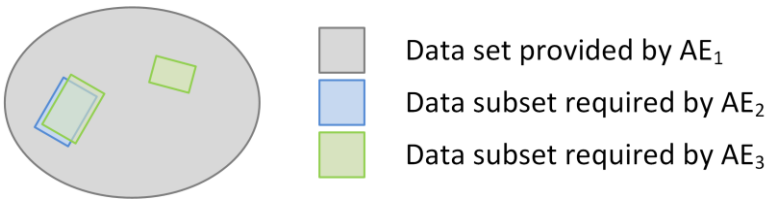

Figure 5.2: Data set provided by an AE and required subset by other AEs

This illustration indicates requirements placed on the data exchange capabilities of the oneM2M platform. However, the oneM2M service platform is universal, and possible effects of filter design or selection capabilities are illustrated by Figure 5.3: It is assumed that the AE_2 requires the data subset, symbolised by the blue rectangle, from the overall available data subset provided by AE_1 (that is symbolised by the grey circle). The left black bordered rectangle represents the data selected by an exemplary filter. Ideally this filter exactly matches the data subset required by the respective AE_2. If the filter differs, it could cause two situations: Either the filter might select more data than required which would result in an overhead (see red hatching), or the filter would cut off required data that causes a filter loss (see yellow hatching), or both.

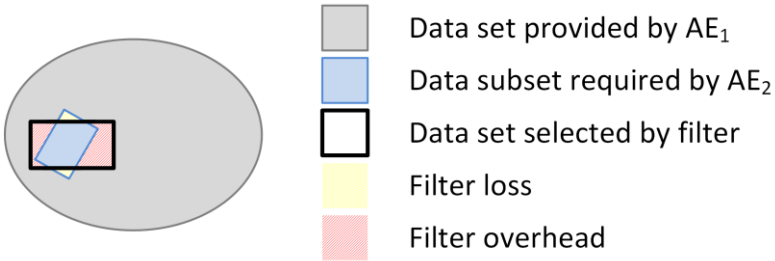

Figure 5.3: Data set provided/required by AEs, filter selection and resulting loss or overhead

Thereby the filter loss may prevent the AE from correctly fulfilling its intended functions, so that this is why in most cases it should be prevented (i.e., through selection of a sufficiently large filter, see Figure 5.4).

Figure 5.4: Data set provided/required by AEs, increased filter selection and resulting overhead

However, the filter overhead should be as small as possible, since the remaining filter overhead in any case decreases the network efficiency of the distributed application. The filter overhead is particularly critical if it remains at a transmission path that is realised through wireless networks. In the given scenario, this transmission path is between the vehicle and the OEM server. Thus, in a worst-case scenario, the absence of appropriate filtering capabilities of the oneM2M service platform might result in too much overhead, which would prevent the practical realisation of distributed use cases or functional splits between AEs, although in theory, the network capabilities are sufficient.

5.2.2 Filter Positions

The prior section detailed general considerations with respect to application data filtering capabilities and requirements. This section continues the discussion, adding the basic data flow between two different AEs residing on different nodes. In the simplified ASDP scenario, such data flow is present, in addition to others, between the AE_1 and AE_2.

The basic data flow consists of four single transmission paths (see Figure 5.5):

1. Data acquisition of the AE_1 through a proprietary interface.
2. Data provision of AE_1 to its local ASN-CSE through the Mca interface.
3. Data transmission from the ASN-CSE to the IN-CSE through the Mcc interface.
4. Data transmission from the IN-CSE to the AE_1 through the Mca interface.

Independent of the actual communication mechanisms used on the single transmission paths (see Section 4.5), this data flow leads to four different positions where filtering can be applied:

- FilterPosition1 is located inside the data providing AE that in this example is the Vehicle Data Provider AE_1.
- FilterPosition2 is located inside the originating CSE that in this example is the ASN-CSE.
- FilterPosition3 is located inside the receiving CSE that in this example is the IN-CSE
- FilterPosition4 is located inside the receiving AE that in this example is the Extended Floating Car Data functionality AE_2.

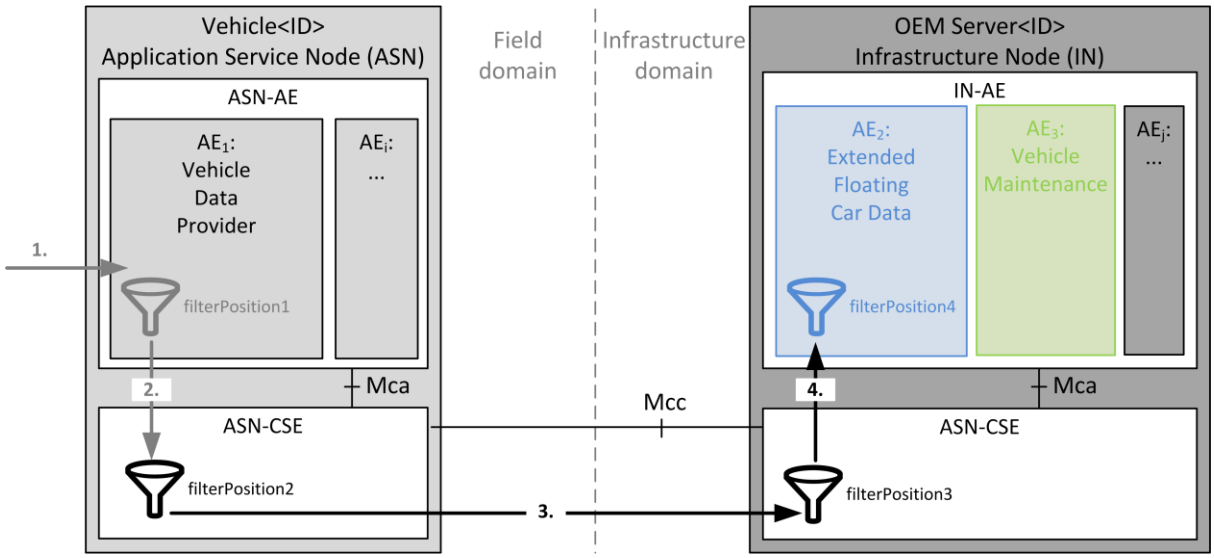

Figure 5.5: Dataflow with possible filter positions between a distributed AE scenario

These filter positions are subsequently discussed with respect to their appropriateness for data filtering against the background of network efficiency, particularly at the wireless network. The

assessment focuses the functionalities to be realised and hence starts from the data consuming AE_2 backwards to the data providing AE_1.

FilterPosition4

The most "expensive" filter position within this data exchange example is filterPosition4, since it means that the remaining overhead at this position will be discarded because it is not required by the AE_2 to fulfil the functionality (see Figure 5.4). Thereby "expensive" relates to the fact that every data which has been transferred until AE_2 has passed the transmission path 3 (see Figure 5.5) and hence has utilised wireless network capacity.

FilterPosition3

If other AEs on the receiving node, such as AE_3 up to AE_j are left aside, filterPosition3 is equally "expensive" for network efficiency compared to filterPosition4, since again the data has already passed the transmission path 3 (see Figure 5.5). FilterPosition3 is becoming more relevant for data aggregation of different AEs, requiring not disjoint data subsets of remote AEs. Thus, it is discussed in more detail at an extended data exchange example, including more than one AE on each side. For this basic example, the filterPosition3 could be assessed as equal to filterPosition4.

Finally, filterPosition1 and filterPosition2 at the originating node are remaining, which are most important for network efficiency of the distributed functionality, since they have the capability to decrease potentially overhead before being transferred across the critical transmission path 3.

FilterPosition1

FilterPosition1 is located inside the AE_1 Vehicle Data Provider. As discussed, the AE_1 is merely a general provider of vehicular data (see Section 5.1.2). With respect to the decoupling of AEs with one another against the background of facilitating vendor and domain independent functionalities, the AE_1 cannot be aware of required data resolutions of current or future consuming AEs. In this regard, the general and a priori clipping of sensor data at filterPosition1, is basically considered as inappropriate.

In particular, the clipping of sensor data at filterPosition1 in anticipation of possible remote consuming AEs and possible resulting wireless access network requirements is not a suitable approach. This prevents other AEs that might be located on the same node from consuming and benefiting from the data at the its original accuracy.

However, this does not mean that each available data in any case has to be provided completely raw without any pre-processing or clipping. In automotive context, visionary sensors, such as cameras, radar, LiDAR (Light Detection And Ranging) or LaDAR (Laser Detection And Ranging) could be counterexamples: Their raw data might be so high that the handling thereof within the vehicle could already be considered as neither effective nor meaningful. In such cases, it is often appropriate to provide pre-processed data that might be smaller by several

orders of magnitude, such as data about objects recognised instead of the raw image. But this counterexample shall not affect the basic statement that filtering at filterPosition1 in general is inappropriate because of the absence of knowledge regarding potential (future) AEs that might even come from other vendors or domains. Particularly, it should be emphasised that design decisions regarding data clipping at the first place shall be guided through their anticipated usefulness for functionalities and not through possible bandwidth or computational requirements.

FilterPosition2

Finally, these explanations lead to the filterPosition2 as the most important position to achieve both functional flexibility and network efficiency. Since the filterPosition2 is located within the (data providing) CSE, related capabilities according to oneM2M service platform will be analysed in the following[15].

5.3 Detailed Analysis of oneM2M Data Exchange Capabilities

In continuation of the previous Section, where the principle considerations regarding filter requirements and positions are discussed, now the actual oneM2M data exchange capabilities are analysed in more detail. These are of major importance with respect to the capabilities and the feasibility of the proposed oneM2M-based ASDP.

Thus, this section starts by presenting the detailed application data handling within the oneM2M service platform. Afterwards, the existing filtering capabilities will be investigated.

5.3.1 Application Data Handling

Prior to data exchange between different AEs, they have to provide their data to the oneM2M service platform. This Section details the related mechanisms.

The general frame for the handling of application data is the container resource where the actual application data is stored within the content attribute of a contentInstance resource (see Figure 5.6). The container can be located below the CSE root, an AE, or another container. The positioning of the data container within the resource tree facilitates coarse-grained modelling of its scope. In addition, the RESTful approach including HATEOS principle, enables links between resources, which can be used to build virtual, “mash-up” resources (see Section 4.4.2). The container itself is a data collection resource whose storage characteristics (retention policies) can be specified with respect to maximum number of stored contentInstance resources

[15] It should be noted here that the technical capabilities of filterPosition2 are equal to those of filterPosition3, because both relate to the CSE, which functionalities are standardised. However, the given example shows that within a scenario with two AEs residing on two different nodes, the filter located inside the data providing node, i.e., the data providing CSE, is that one of major importance.

(maxNrOfInstanes), the maximum storage size (maxByteSize), and the maximum age of contentInstance resources (maxInstanceAge).

Inside the container resource zero to n contentInstance resources can be stored according to the previously named constraints. It must be considered that contentInstance resources can only be created, read, and deleted - the update of an existing contentInstance is not supported. Finally, the contentInstance resource can be divided into the application data itself, which is stored within the attribute content, and accompanying meta data. A reduced illustration of this twofold contentInstance resources within their container, as used in some figures in the following, is exemplified on the right side of Figure 5.6 by means of a VehicleDataProvider AE with an VehicleData container including three contentInstance resources.

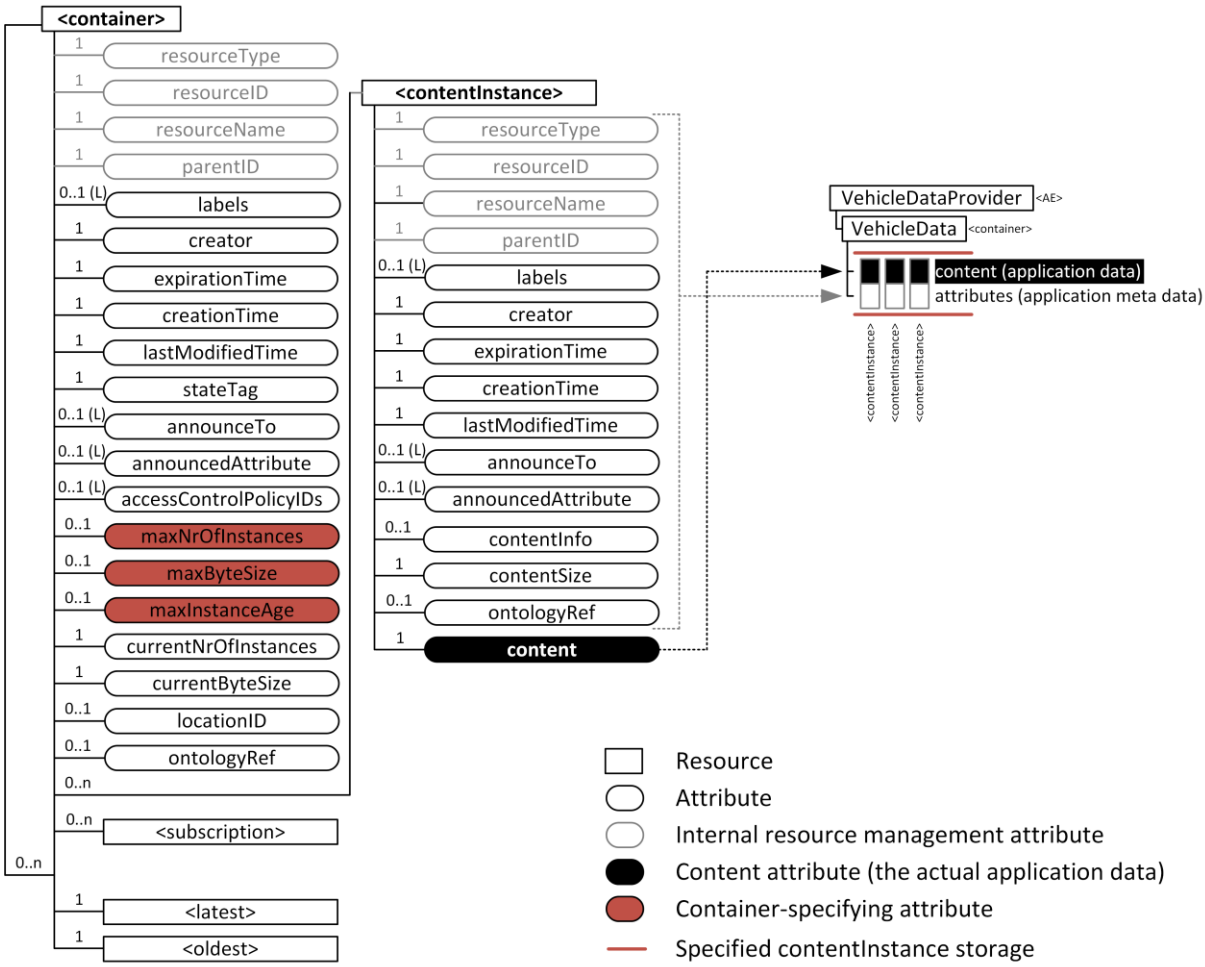

Figure 5.6: Resource structure of container and contentInstance

Application meta data are, for example, textual descriptions within the attribute label, the size of the content, attributes related to creation and expiration time, as well as the universal lastModifiedTime attribute, which is equal to the creationTime since an update is not supported. Further, two attributes are related to resource announcement capabilities (see Section 4.5.3), which are also present for contentInstance resources.

Finally, two attributes are available for specification of the actual application data which is stored within the content attribute: These are the attribute contentInfo, which provides the syntactical information, and the attribute ontologyRef that provides a reference (URI) to the semantic information, i.e., an ontology. Both attributes are intended to facilitate the interoperability between AEs without a priori knowledge (see Section 4.2.1).

ContentInfo is composed from two parts: The first part is the Internet Media Type according to IETF RFC 6838 (Freed, Klensin, & Hansen, 2013), followed by a ":" as separator and the encoding information (oneM2M TS-0001, 2015; oneM2M TS-0004, 2015). Up to now, possible encodings are "0" for plain, "1" for base64-encoded string, and "2" for base64 encoded binary (oneM2M TS-0004, 2015). The default value of the attribute contentInfo is "text/plain:0" (oneM2M TS-0004, 2015), specifying that the attribute content provides a plain text. Another common setting might be "application/xml:1", if the attribute content provides base64-encoded application-related XML data (cf. Freed et al., 2013). The fact that the contentInfo attribute is (only) available within the contentInstance resource and thus only for each contentInstance individually but not commonly for the whole container resource, reflects the RESTful approach of different representations of a single resource (see Section 4.4.2). For example, the speed of the car might be provided as "application/xml:1", "application/json:1", or even visually rendered as "image/jpeg:2". All of these different representations can be stored within the same container resource.

The ontologyRef attribute is a first outcome from considerations about semantic interoperability between different AEs (cf. oneM2M TR-0007, 2015). However, its detailed usage is out of scope within the current release oneM2M (oneM2M TS-0001, 2015; oneM2M TS-0004, 2015). Hence the attribute ontologyRef, specifying a URI is currently more or less a placeholder. However, semantic interoperability is an important aspect within M2M, IoT and automotive scenarios, which is why it will be subject of a more detailed excursion below (see Section 6.3.3). It should be noted that the ontologyRef attribute is also present for the container and for AE resources, to enable the reference to their superior semantics.

After having detailed the attributes that technically can be used to specify the syntax and (with restrictions) semantic of the content, their concrete usage for the purpose of storage, provision and exchange of application data remains. The oneM2M service platform only provides the framework for it and does not specify particular rules or guidelines. In this regard, it is up to the AE (developer), how application data is structured and split across different containers, and what media types and encodings are provided. For example: The content might contain only a single value, such as the speed of the vehicle. Then again, it might also be possible to provide a content that composes several values, as it might be meaningful, e.g., for the vehicle position that is specified by longitude, latitude, and altitude. The decision as to whether application data such as sensor values are provided within a single or different container is a trade-off that must be decided individually. Common practices for RESTful service and resource modelling might

also be an appropriate methodology here (cf. Erl et al., 2013, p. 155 ff.). However, the smallest unit of application data exchange is currently a single contentInstance resource. Consuming AEs cannot adapt the application data splitting across different containers, and hence must be capable of handling the application data, i.e., the content attribute, as provided. This increases the necessity to build adequate data containers, albeit envisaged enhancements towards full semantic interoperability might enable individually tailored data containers through dynamically build virtual resources (see Section 6.3.3).

Although AEs are capable of providing in particular the syntactical information by use of the contentInfo attribute to facilitate the decoding of their application data stored within the content attribute, this information is currently not being used at CSE level. During the process of developing a universal horizontal integration platform, oneM2M has been chosen to handle application data in a first step opaque at CSE level. This means any application data gets a black box at CSE level, and hence is not transparent to the CSE functionalities (CSFs) of the oneM2M service platform. This has strong implications on the data exchange and filtering capabilities which are the subject of the next paragraph.

5.3.2 Existing Filtering Capabilities for Exchange of Application Data by Use of the Subscribe/Notify Mechanism

The prior Section detailed the application data handling within oneM2M service platform. This is the foundation to analyse the existing capabilities for exchange of application data between AEs in more detail.

Application data exchange within the oneM2M service platform refers to the transmission of contentInstance resources between two AEs. As discussed in Section 4.5, the oneM2M service platform offers different mechanisms for this resource exchange, which are combinations of Create/Retrieve, Announcement, and Subscribe/Notify. In any case, filtering in the context of application data exchange refers to the capability to select (and transmit) only a subset of the overall available application data, i.e., the selection of certain contentInstance resources from a container resource (cf. Section 5.2.1).

The analysis of application data exchange capabilities is subsequently performed by means of the subscribe/notify mechanism for to the following reasons: At first, the subscribe/notify mechanism enables the most decoupling of the involved communication entities, see Section 4.5.4. Furthermore, currently the subscribe/notify mechanism provides the most extensive capabilities for application data filtering. Both make the subscribe/notify mechanism particularly suitable for a oneM2M-based ASDP.

The subscribe/notify mechanism is not limited to the monitoring of contentInstance resources within a container resource for the purpose of application data exchange. It is a general mechanism that enables the subscription to and detection of resource changes, referred to as

resource events. In this regard, filtering at first means the decision as to whether a particular resource event triggers a notification or not. Thus, the filterCriteria within the subscription resource has been renamed to the more adequate term eventNotificationCriteria over time of M2M/oneM2M standardisation and harmonisation process (cf. ETSI TS 102 690, 2013; oneM2M TS-0001, 2015). Furthermore, it emphasises the difference to a resource retrieve, where the term filterCriteria is still present because here it actually specifies a filter that may provide a subset of the resource being retrieved. However, if the subscribe/notify mechanism is used to monitor the creation of application data (i.e., the creation of contentInstance resources), the notification criteria and all other criteria that decide on the triggering and transmission of notification can be considered as filter criteria for application data exchange. In this regard, the term filter criteria can be used synonymously for that purpose.

Initial Filter Considerations

Section 5.2.1 discussed general considerations about the significance of appropriate filtering capabilities with respect to network efficiency of the distributed use case. In this regard, the filtering capabilities for the subscribe/notify mechanism of oneM2M are investigated. Continuing the illustrations of Section 5.2.1 there must be analysis performed to determine which capabilities the oneM2M service platform provides "cut the data filtering shape" to the actually required data subset, such as the respective AEs of the exemplary ASDP scenario (see Section 5.1). While the analysis by use of the ASDP scenario is treated in the next Section, here the entire filtering capabilities provided by the current version of the oneM2M standard are discussed in detail.

Ultimately, AE developers want to know their actual filter capabilities for notifications provided by the oneM2M service platform (i.e., the CSE) to evaluate how this relates to their requirements. As this Section will show, these actual filter capabilities (referred to as filterCombined) are the combination of three single filterAspects (see Figure 5.7).

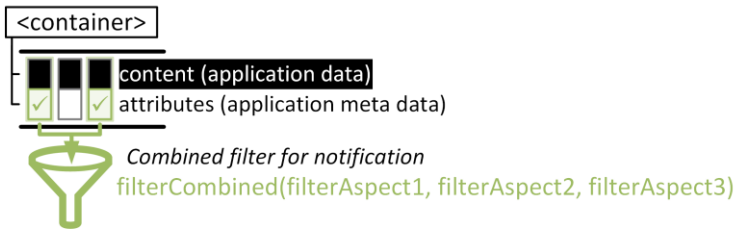

Figure 5.7: Combined filter for notifications

Figure 5.8 shows the compound container resource including child resources related to application data storage (contentInstance resource) and exchange through the use of the subscription/notify mechanism (subscription resource), including filter criteria (eventNotificationCriteria attribute and schedule resource).

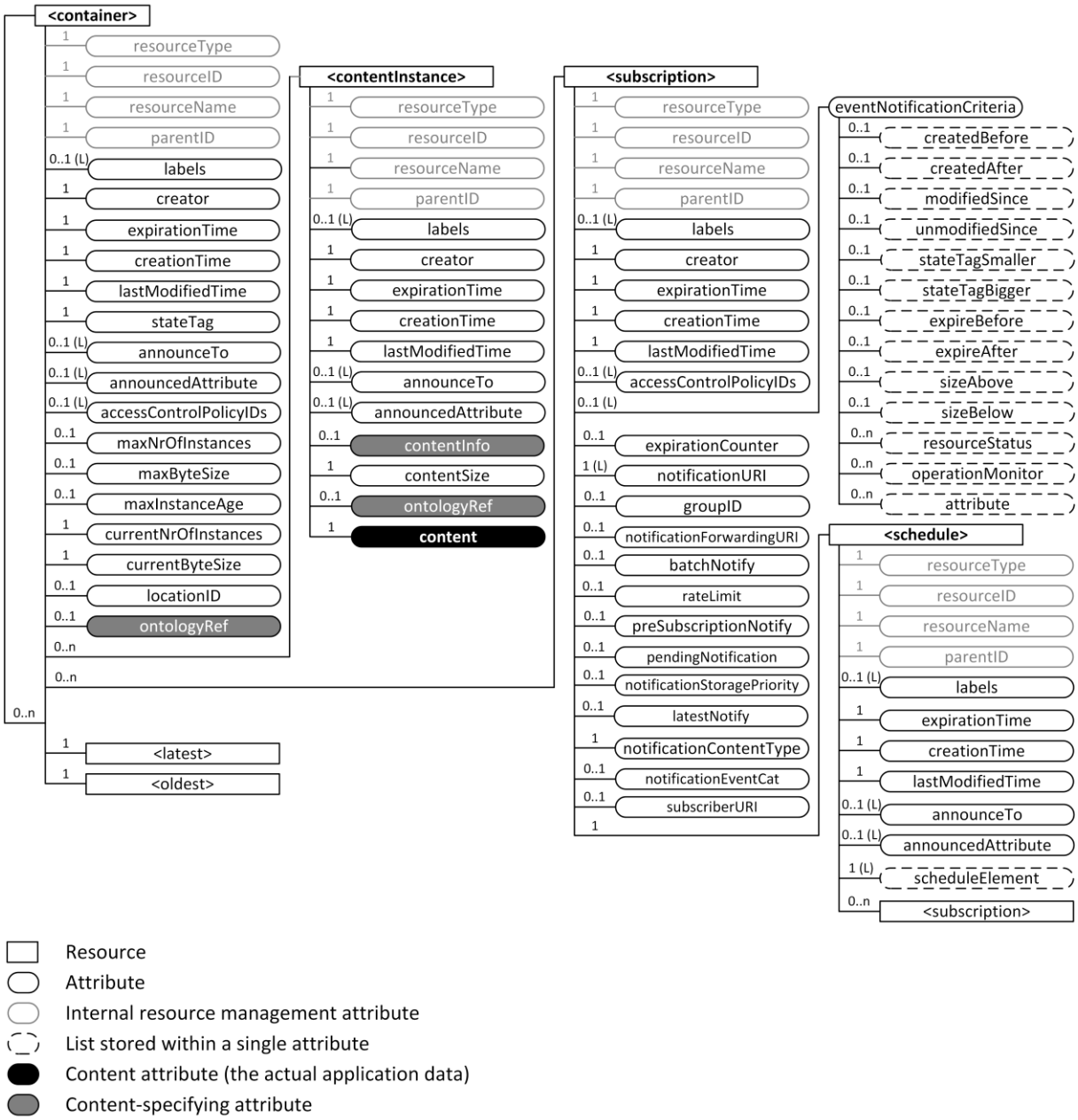

Figure 5.8: Compound resource structure of container, contentInstance, subscription, eventNotificationCriteria, and schedule

A closer investigation of the subscribe/notify mechanism shows that the overall filter capabilities can be considered as the concatenation of three single filter aspects or filter functions on their respective (remaining) data subsets. These three filter aspects in their entirety decide whether a notification is being sent or not (see Figure 5.9). Before detailing every single filter aspect, they are briefly introduced in the following enumeration:

- filterAspect1 decides if a particular resource event triggers a notification. Instantiated for the purpose of application data exchange, the filterAspect1 hence filters which contentInstance basically should get notified.

- filterAspect2 facilitates the specification determining which subset of the resource that is to be announced shall be part of the notification. In the context of application data exchange, the filterAspect2 specifies which subset of the contentInstance resource should be included in the notification.
- filterAspect3 provides several capabilities for specifying notification transmission constraints. Since these constraints could also discard previously triggered notifications, these CMDH-related policies constitute another filter criterion.

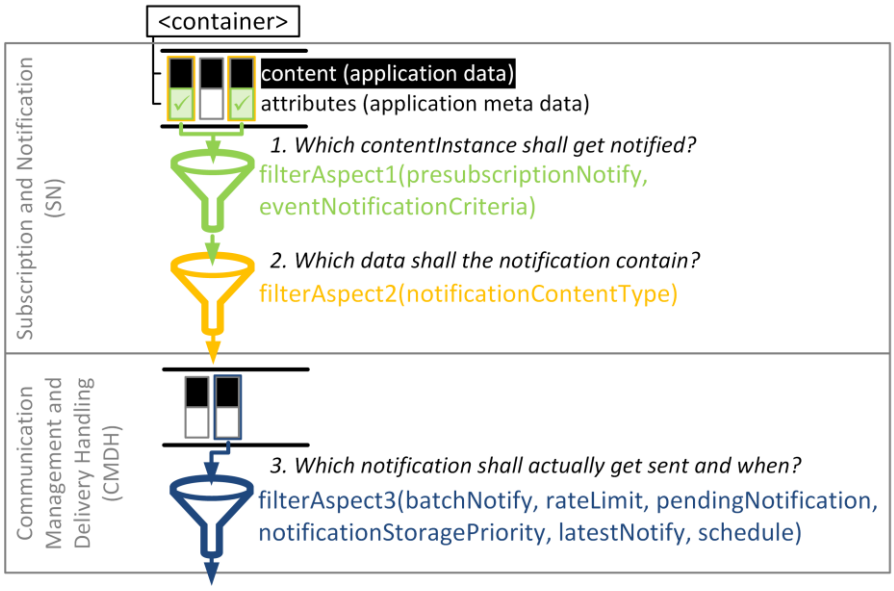

Figure 5.9: Concatenation and interaction of three filter aspects of subscribe/notify mechanism

Furthermore, Figure 5.9 shows, that filterAspect1 and filterAspect2 are realised within the Subscription and Notification (SN) CSF, and that filterAspect3 is part of the Communication Management and Delivery Handling (CMDH) CSF (see Section 4.3.2).

The functionalities of these three filters are discussed afterwards in more detail, focusing their usage for conditional application data exchange by means of the subscribe/notify mechanism. With respect to differentiation and understanding, capabilities beyond data exchange and the respective resources are indicated, where appropriate.

FilterAspect1: Event Notification Criteria

Due to the fact that the actual application data is currently handled opaque within the oneM2M CSE (see Section 5.3.1), the filtering constraints can only address application meta-data (see Figure 5.6), which means they can only refer to attributes other than the content attribute itself. Such application meta-data criteria to trigger a notification can be configured by means of the eventNotificationCriteria attribute.

To comprehensively explain the existing notification capabilities, Figure 5.10 illustrates the supported list of notification criteria of the eventNotificationCriteria attribute and related

attributes (of the container or contentInstance resources) to which they may be applied to, indicated through the same colouring.

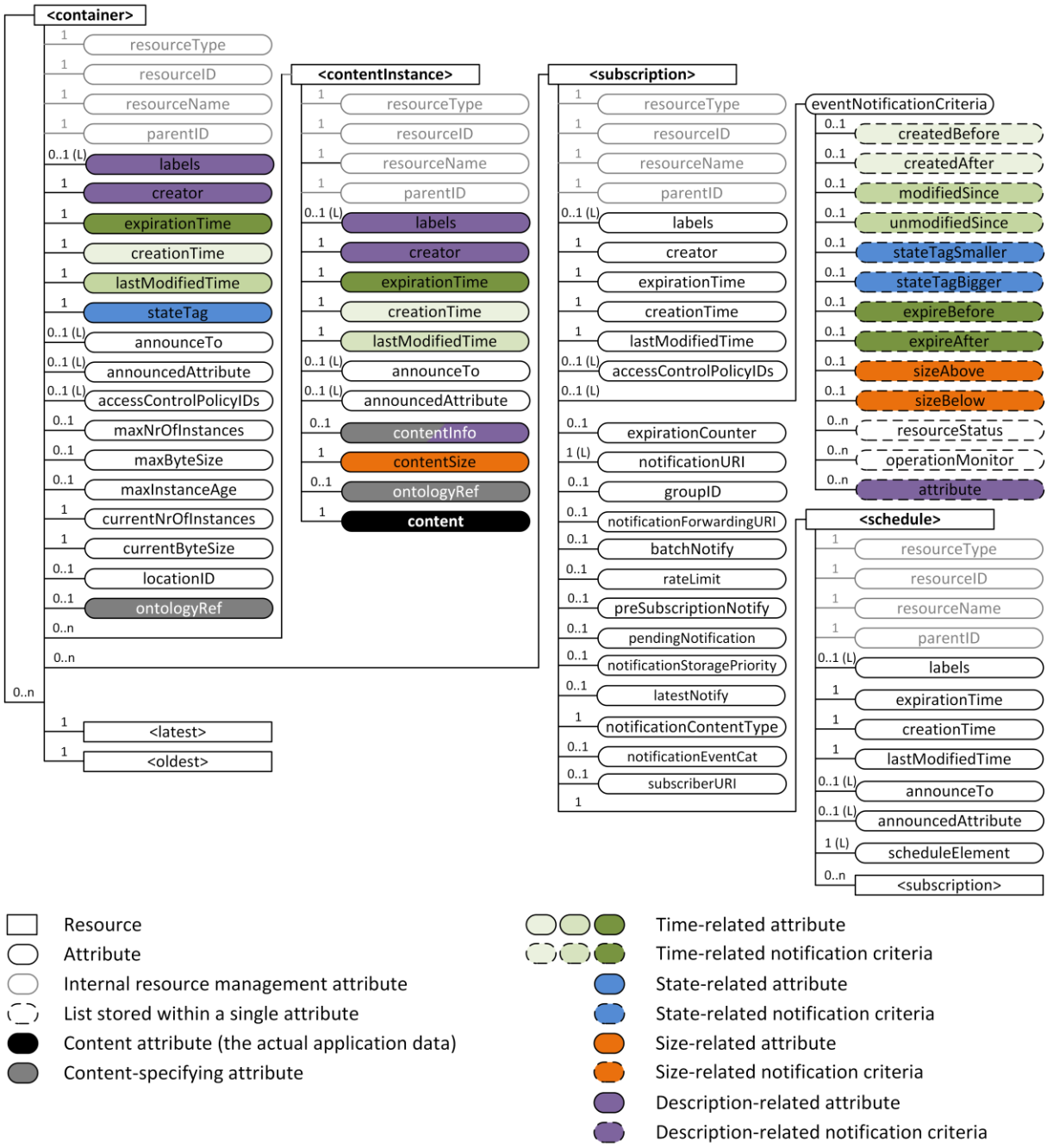

Figure 5.10: Compound resource structure of container, contentInstance, subscription, eventNotificationCriteria, and schedule including relations of the notification criteria

Those criteria that finally determine whether a resource event trigger triggers a notification or not are described subsequently in more detail, according to (oneM2M TS-0001, 2015). Criteria related to the resourceStatus:

- This value relates to the operation that has been performed on the subscribed-to resource. In the context of application data exchange, an appropriate value is “child created”, which puts the focus on the contentInstance (or schedule) resources. Alternatively, it could be set, e.g., to “updated” and hence focus/monitors the container resource itself. If the resourceStatus value puts the monitoring toward child resources, the other eventNotificationCriteria are applied to child resources, too (i.e., the contentInstance resources). Otherwise they are partially applied to the resource container. For the exchange of application data which is stored within the content attribute of a contentInstance, the configuration “child created” is appropriate to detect the creation of new contentInstance resources (including application data), and to trigger notifications depending on other notification criteria.

Criteria related to the operationMonitor:

- This attribute is similar to the resourceStatus, but it monitors every operation (create, retrieve, update, delete) that is attempted to be performed, although it might not be successful. This is useful for finding malicious AEs. However, for the purpose of application data exchange, this is of minor significance.

Time-related criteria/conditions (green coloured attributes and notification criteria):

- The criteria createdBefore and createdAfter refer to the attribute creationTime of the resource container or contentInstance.
- The criteria modifiedSince and unmodifiedSince refer to the attribute lastModifiedTime of the resource container or contentInstance.
- The criteria expireBefore and expireAfter refer to the attribute expirationTime of the resource container or contentInstance.

State-related criteria/conditions (blue coloured attributes and notification criteria):

- Each time a resource is modified, the stateTag is increased. This offers a simplified analysis of resource changes in addition to time-related analysis capabilities. The variables stateTagSmaller and stateTagBigger are criteria that refer to the attribute stateTag of the resource container. Since the update of existing contentInstance resources is not supported, they do not have a stateTag.

Size-related criteria/conditions (orange coloured attributes and notification criteria):

- The criteria sizeAbove and sizeBelow refer to the attribute contentSize of the resource contentInstance. This attribute always refers to the contentSize, independent from the configured resourceStatus.

Description-related criteria/conditions (purple coloured attributes and notification criteria):

- This value facilitates "full-text filtering" on resource attributes, which is, besides others, particularly suitable for the following purposes:
    - Descriptive information for containers and contentInstance resources can be provided within the attribute labels which might contain a list of values. For example, a possible configuration of the labels attribute criteria might be 'labels=(type=temperatureValue;device=vehicle)'[16].
    - The creator of a container or contentInstance is stored within the attribute creator. The attribute criteria can be used to filter accordingly, such as 'creator=VehicleDataProvider'[16].
    - Finally, the attribute criteria can also be used to filter notifications to supported content encodings such as 'contentInfo=application/xml:1'.

The attribute preSubscriptionNotify of the subscription resource specifies, if (and how many) notifications shall also be triggered for past resource events which may have occurred prior to the subscription creation and are still stored. Further constraints, such as the previously described eventNotificationCriteria, may also be applied to these earlier events, and hence finally decide if past resource events trigger notifications.

FilterAspect2: Notification Content

The filterAspect1 has selected the resource that is to be notified, such as a contentInstance resource in the context of application data exchange. The filterAspect2 specifies the resource attribute subset that is included in the notification. For example, it specifies which subset of the contentInstance resource to be notified is actually included in the notification message. This is specified by use of the notificationContentType of the subscription resource (see Figure 5.11).

However, since the actual application data, stored within the content attribute, remains opaque at CSE level (see Section 5.3.1), filtering with the filterAspect2 cannot address the actual content (i.e., the value) of attributes. This means if a content attribute of a single contentInstance provides, for instance, the position (latitude, longitude, heading), the speed, and ESP control intervention status, filtering to a subset, e.g., that the content attribute only contains the speed value, is not supported.

[16] The semicolon is used here as separator for the list items, which might not be the syntax of oneM2M.

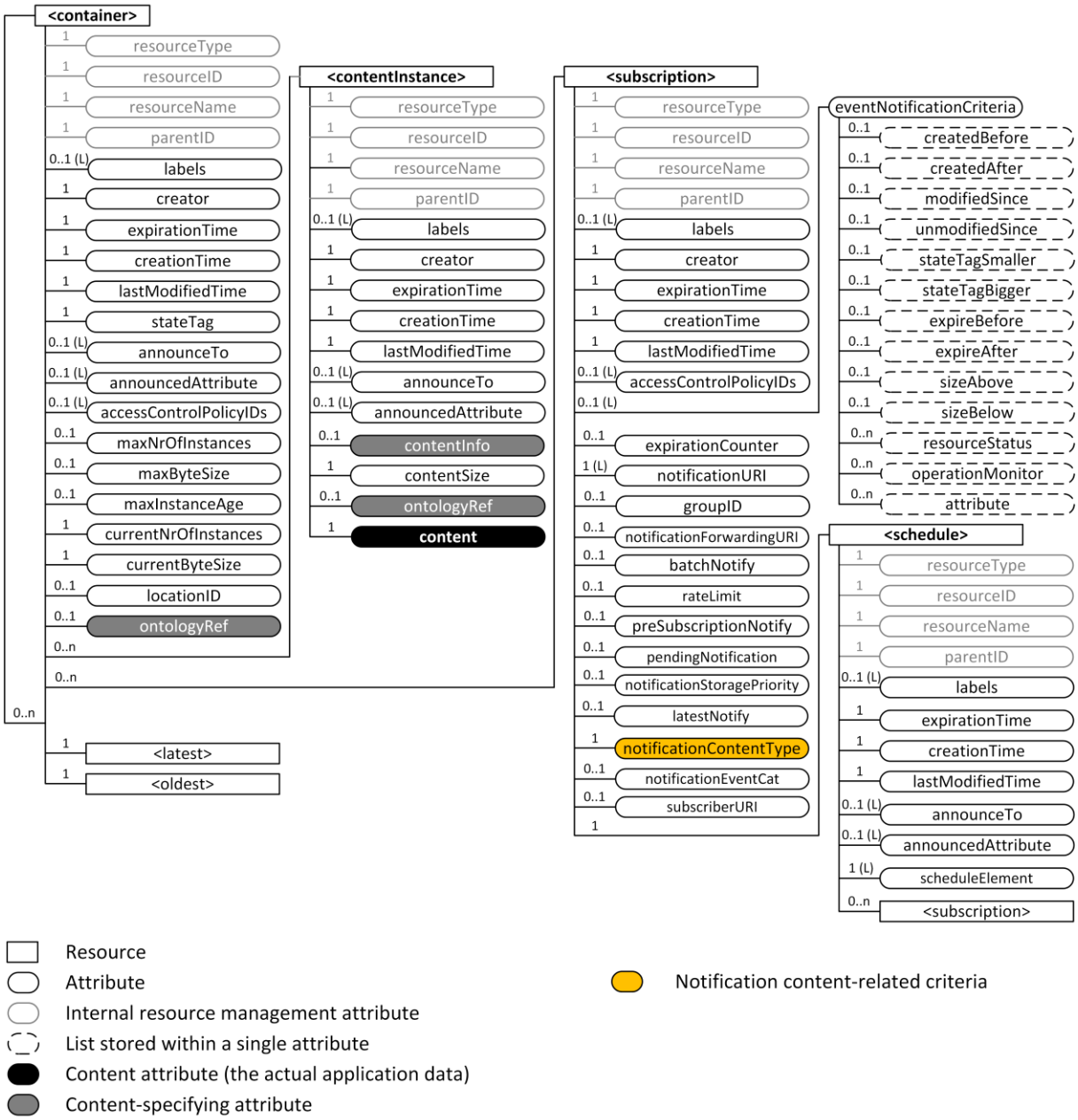

Figure 5.11: Compound resource structure of container, contentInstance, subscription, eventNotificationCriteria, and schedule including content-related notification criteria

Nonetheless, at the level of resource attributes, filtering capabilities exist: The attribute notificationContentType specifies such resource subset, actually delivered within the notification message. Possible configuration values are (oneM2M TS-0001, 2015):

- ‘modifiedAttributes’: The notification only contains the modified attributes.
- ‘wholeResource’: The notification contains the whole resource.
- ‘referenceOnly’: The notification contains only a reference to the resource, i.e., a reference to the contentInstance created.

With respect to the current release of oneM2M standards, it is not exactly specified as to whether the value 'modifiedAttributes' refers to the delta related to the most recent notification of this particular subscription or if it refers to the delta between the two latest contentInstance resources. It could be assumed that the latter will be the case, which in combination with eventNotificationCriteria could possibly cause an incomplete or incorrect view of the current status of the contentInstance resource for the subscriber. Hence, the configuration value 'modifiedAttributes' might be inappropriate for certain considerations related to exchange of application data.

Besides, it must be noted that the configuration 'referenceOnly' implies that the actual application data exchange does not occur as part of the notification message: hence not as part of the subscribe/notify mechanism. The subscriber may afterwards receive the related resource (which may include the application data) within a separate retrieve operation. However, this violates the intended beneficial indirect communication characteristics of the subscribe/notify mechanism for application data exchange. Thus, this configuration has not been chosen.

In summary, it can be stated that the configuration 'wholeResource' is to date the most appropriate value for the attribute notificationContentType with respect to the direct exchange of application data by use of the subscribe/notify mechanisms. Thus, it is presumed for the following considerations.

FilterAspect3: Notification Delivery

After the selection of resources and their attributes that are part of a notification through the filterAspect1 and filterAspect2, a third filter capability (FilterAspect3) exists. It is related to the CMDH CSF (see Section 4.3.3), which means that the filterAspect3 is applied before the notification is actually being sent. Figure 5.12 illustrates the two categories of notification delivery policies consisting of time-related notification schedule criteria and delivery-related notification criteria.

The notification delivery policies were originally motivated by and derived from the network characteristics of M2M (see Section 4.2.2). For example:

- The attribute batchNotify facilitates the local collection of notifications to send them afterwards as a batch at once. Therefore, the number of notifications cached and the timespan can be specified.
- The attribute rateLimit enables the specification of a maximum number of notifications within a selected time period. If the limit is exceeded, further notifications are temporarily stored.

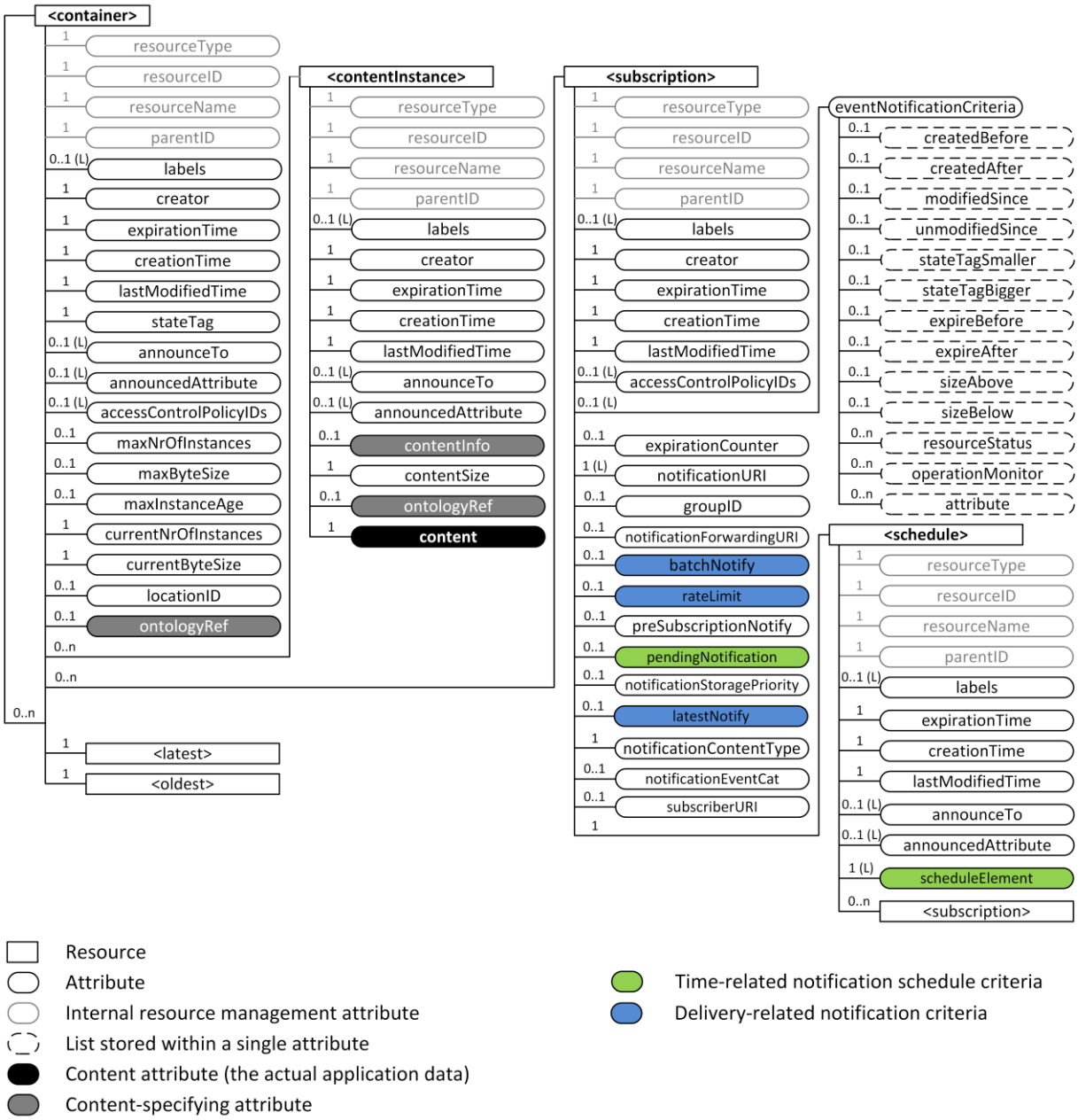

Figure 5.12: Compound resource structure of container, contentInstance, subscription, eventNotificationCriteria, and schedule including time-related notification schedule criteria

Although these mechanisms and attributes were introduced for the purpose of delivery handling, e.g., of notification messages, they generate a third filtering capability: In combination with the attribute latestNotify it is possible that only a subset of the cached notifications, i.e., the most recent notification at the time of actual notification delivery, are being sent to the subscriber. Thus, the combination of those attributes generate an additional filtering capability for notifications, and – assuming an appropriate usage of the subscription/notify mechanism – the filtering of application data that is being exchanged.

Another policy that is related to notification message delivery can be specified by means of the attribute scheduleElement as part of the schedule resource (oneM2M TS-0001, 2015). It can be

used to configure schedule policies when the subscriber wants to receive notifications from his subscription[17]. The pattern facilitates the definition of a notification schedule policy by use of second, minute, hour, day of month, month of the year, and day of the week, including asterisks ('*'), ranges ('-'), and enumerations (separated by ','). Assuming, for example, that the subscriber wants to receive notifications from the related subscription every 10 seconds, a scheduling policy could be '0,10,20,30,40,50 * * * * *'. It must be considered that this, in fact, refers to absolute points in time (in the given example the seconds 0, 10, 20, 30, 40, 50 of a minute) and does not specify relative time differences between notifications. In comparison to batchNotify and rateLimit, a notification schedule hence does not depend on the time when notifications become eligible for delivery. The attribute pendingNotification specifies how to handle notifications that have been cached because they became eligible for delivery outside of a permissible time window, specified by the notification schedule. Possible values are 'sendAllPending', which leads to the sending of all cached notifications, and 'sendLatest', which leads to the sending of the latest notification being cached. The latter again might lead to the selection of a subset of the overall available notifications or contentInstance resources and hence is another indirect filtering capability.

Within the current release of oneM2M, the mechanisms related to batchNotify and rateLimit, although specified as resource attributes, are not supported (oneM2M TS-0004, 2015, p. 144). However, the general idea of the two accompanying delivery constraints on eligible notifications that are batchNotify and rateLimit on one hand and subscription notification schedule (and node reachability schedule) on the other hand are understandable. However, all possible permutations, e.g., with respect to eventNotificationCriteria, preSubscriptionNotify, pendingNotification, might require detailed consideration and specification through oneM2M.

Concluding Considerations Regarding Conditional Application Data Exchange

The filterAspect1 related to the eventNotificationCriteria provide limited capabilities for the subscribe/notify mechanism:

- Since the Update method is not supported on contentInstance resources, creationTime and lastModifiedTime values are equal. Thus, createdBefore actually equals unmodifiedSince, and createdAfter actually equals modifiedSince. All four criteria specify a particular moment in time, which means once the time is reached, these criteria are either true or false for all future contentInstance creations, and hence provide limited filter capabilities with respect to application data exchange.

[17] Furthermore, there is the capability of optionally specifying a scheduling policy for an entire node, which is referred to as the reachability schedule (in contrast to the notification schedule of a single subscription) and which is located below a CSEBase resource. The CMDH functionality may need to consider both scheduling policies in combination before the sending of notifications. Within the scope of the examples here this does not provide an added-value and hence is not part of further consideration.

- The stateTagSmaller and stateTagBigger criteria are similar, since these policies, once fulfilled, apply either only for all prior contentInstance resources or for all future contentInstance resources.
- The expireBefore, expireAfter, sizeAbove, sizeBelow may provide certain filter capabilities, since these criteria can at least vary independently from the timeline. In this regard, they provide the capabilities of selecting a subset of contentInstance resources over time. However, expirationTime-related and size-related filters are considered as less relevant with respect to data exchange requirements of distributed use cases.
- The filter capabilities related to the attribute criteria of the eventNotificationCriteria attribute provide certain filter capabilities with respect to full-text searches within attributes of contentInstance resources. Nevertheless, it excludes the content attribute itself. Indeed, there are some examples, where such attribute criteria may provide a useful filter, e.g., for filtering with respect to the sensor type that is unveiled in the label attribute. However, these descriptions are not standardised: thus they lack syntactical and semantic standardisation (cf. Section 4.2.1). This means extending the filter capabilities of the attribute criteria towards a generic and interoperable capability raises similar requirements to the oneM2M service platform evolution as the enhancements necessary for direct filtering against the content attribute itself. This is why firstly it does not provide an adequate solution for application data today, and secondly it might face limited importance for the future, if full semantic interoperability is achieved (see Section 6.3.3).

The notificationContentType attribute (filterAspect2) specifies what is actually included in a notification but has no impact on whether a notification is triggered or not. In this regard its functionality is of minor importance with respect to conditional application data exchange.

Finally, the filterAspect3 relates to the CMDH CSF (see Section 4.3.3) and shall enable adequate capabilities regarding time-related filtering and related policies. In this regard, it could be used to implement time-triggered data exchange policies.

In summary, it can be stated that the current form of the subscribe/notify mechanism of oneM2M regarding application data exchange corresponds at most to the topic-based (alternatively referred to as subject-based) model of the publish/subscribe system, because AEs basically subscribe to resources, which can be considered as topics (Coulouris et al., 2012; Mühl, Fiege, & Pietzuch, 2006). Additionally, the subscribe/notify mechanism has certain capabilities of the type-based model for a publish/subscribe system (Coulouris et al., 2012; Mühl et al., 2006): The current eventNotificationCriteria can refer to the contentInfo (i.e., data type), respectively the label (that can be used, e.g., for the description of the type or class).

5.3.3 Analysis with ASDP scenario

Above, the principle considerations related to data exchange within the oneM2M service platform were discussed. This Section analyses these capabilities by means of the ASDP scenario introduced in Section 5.1. Following the previous consideration, the data exchange is performed by use of the subscribe/notify mechanism.

Bootstrap

Step 1: The CSEVehicle<ID>Base registers at the CSEOEMServer<ID>Base.

Step 2: The AE1 registers at the ASN-CSE as 'VehicleDataProvider' through the creation of respective AE resource.

Step 3: The AE1 creates the data container 'VehicleData'.

Step 4: The AE2 registers at the IN-CSE as 'ExtendedFloatingCarData' through the creation of respective AE resource.

Step 5: The AE3 registers at the IN-CSE as 'VehicleMaintenance' through the creation of respective AE resource.

Subscription Setup

It is assumed that all necessary bootstrap procedures are successfully performed. Furthermore, this example presumes that AE2 and AE3 already have the necessary information that the Vehicle<ID> is available and that it has an application AE1 'VehicleDataProvider' that is capable of providing data for the implementation of the use cases of AE2 and AE3. This information can be obtained by use of discovery functionalities of the oneM2M service platform or through a-priory configuration. However, this research focuses the data exchange capabilities. Thus, discovery considerations are bypassed here (cf. DIS CSF in Sections 4.3.2 and 6.3.3).

The most important aspects for data exchange between the AEs by use of the subscribe/notify mechanism are the notification criteria. This is where the requirements of AEs, respectively (distributed) use case, meets the capabilities of the platform (i.e., the oneM2M CSE). Diverse policies could be beneficial for the AEs according to their application logic (see Section 5.1.2). However, according to the existing oneM2M capabilities (see Section 5.3.2), time-related criteria are defined for the ASDP scenario:

- SubscriptionAE2: The AE2 creates this subscription resource at the VehicleData container of the VehicleDataProvider AE of the CSEVehicle<ID>Base to receive notifications in case of the creation of new contentInstance resources. For time-related constraints, AE2 includes a schedule resource into the subscription, with a scheduleElement, configured to receive notifications at a maximum of every 10 seconds. Additionally, the configuration sendlatest is applied to receive only the most recent contentInstance resource. These subscriptions are sent through the local

IN-CSE, where they are both re-targeted to the target ASN-CSE. For the opposite notifications, local callbacks (within the IN-CSE) are created to route them to the AEs.

- SubscriptionAE3: The AE3 creates this subscription resource at the VehicleData container of the VehicleDataProvider AE of the CSEVehicle<ID>Base to receive notifications in case of the creation of new contentInstance resources. For time-related constraints, AE3 includes a schedule resource into the subscription, with a scheduleElement, configured to receive notifications at a maximum of every 5 seconds. Additionally, the configuration sendlatest is applied to receive only the most recent contentInstance resource. These subscriptions are sent through the local IN-CSE, where they are both re-targeted to the target ASN-CSE. For the opposite notifications, local callbacks (within the IN-CSE) are created to route them to the AEs.

Data Exchange

Figure 5.13 illustrates the current data exchange capabilities of oneM2M with the given condensed ASDP scenario (see Section 5.1). It is shown, how two AEs (AE2 and AE3) that registered with two subscriptions (SubscriptionAE2 and SubscriptionAE3) for VehicleData of AE1 get notified with VehicleData in a time frame of about 15 seconds.

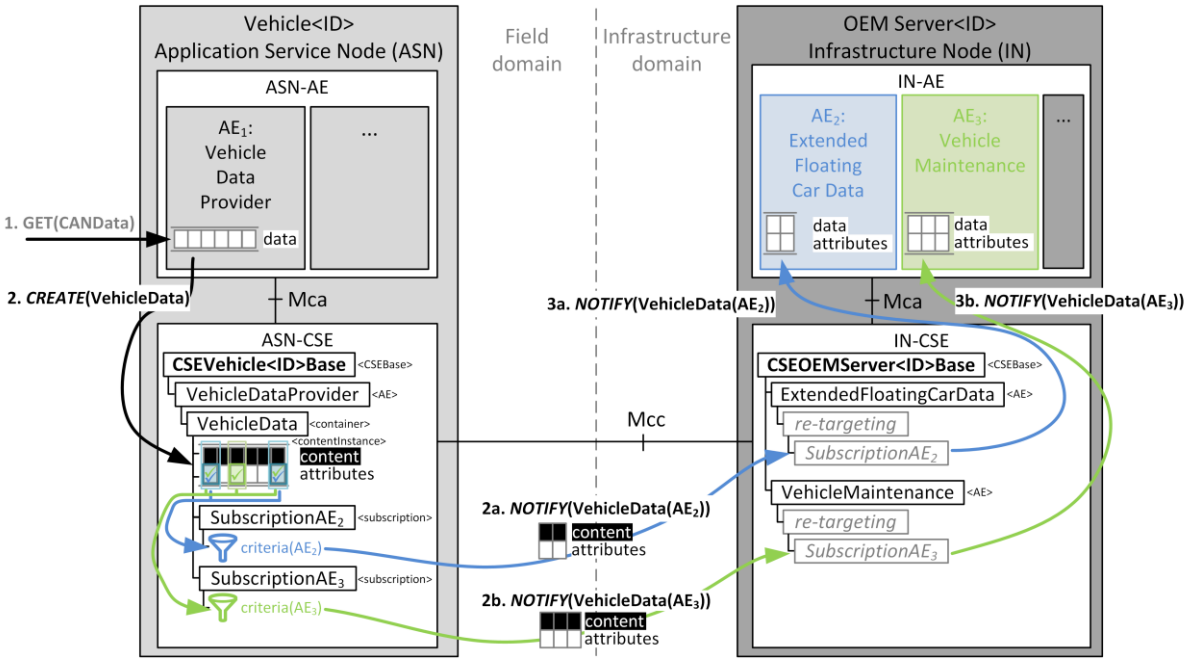

Figure 5.13: Application data exchange example of ASDP scenario with current oneM2M data exchange capabilities

- Step 1. GET(CANData):
  The AE1 proprietary obtains the vehicle data from the in-vehicle CAN-bus.
  This step is performed six times within this example, generating the six vehicle data samples, as illustrated.

- Step 2. CREATE(VehicleData):
  The AE1 provides this data without filtering (cf. Section 5.2.2) as new contentInstance resource to the VehicleData container within its local ASN-CSE. Thereby, the VehicleData is transformed to a vendor-independent application/xml representation (cf. Section 5.1).
  This step is also performed six times within this example, generating the six contentInstance resources including VehicleData, as illustrated.

The ASN-CSE checks at every contentInstance creation event to determine if the notification criteria of existing subscriptions are met and if a notification should actually be sent (SN and CMDH, see Section 5.3.2).

- Step 2a. NOTIFY(VehicleData(AE2)):
  If the notification criteria of the subscription of AE2 is fulfilled, the related notification is sent. This policy leads to a filtering of two of six VehicleData contentInstance resources which are being sent from the ASN-CSE to the IN-CSE as single notification messages.
- Step 2b. NOTIFY(VehicleData(AE3)):
  If the notification criteria of the subscription of AE3 is fulfilled, the related notification is sent. This policy leads to a filtering of three of six VehicleData contentInstance resources which are sent from the ASN-CSE to the IN-CSE as single notification messages.
- Step 3a. NOTIFY(VehicleData(AE2)):
  Each of the two single notification messages is re-targeted at the IN-CSE to the (server-capable) AE2.
- Step 3b. NOTIFY(VehicleData(AE3)):
  Each of the three single notification messages is re-targeted at the IN-CSE to the (server-capable) AE3.

The detailed assessment of this scenario is part of the following section.

5.4 Shortcomings of the Current oneM2M Data Exchange and Their Impacts

The exemplified ASDP scenario reveals two significant shortcomings of current oneM2M data exchange capabilities, namely the absence of application-data-dependent criteria for data exchange and missing aggregation of different subscriptions. These shortcomings and their impacts are discussed in the following.

5.4.1 No Application-Data-Dependent Criteria for Data Exchange

Although the attribute contentInfo of the contentInstance resource is introduced to facilitate the specification and provision of syntactical information of the application data (see Section 5.3.1), by design-choice of the current oneM2M standard, it is not used at the CSE. Hence the application data remains opaque at the CSE level. This has a significant impact on the available notification criteria (i.e., filter conditions) of the subscribe/notify mechanism. These criteria cannot depend on application-data (stored and transferred within the content attribute) but can only depend on application meta data (see Section 5.3.1).

Accordingly, reasonable application-data-dependent criteria for data exchange that are directly derived from the distributed use case to be implemented, cannot be specified, such as:

- Notification, when remaining fuel range is smaller than 100 km. (see Vehicle Maintenance / Fleet Management, Section 3.2.2)
- Notification, when heavy rain is detected. (see Extended Floating Car Data, Section 3.2.1)
- Notification, when there is a risk of freezing rain. (see Extended Floating Car Data, Section 3.2.1)
- Notification, when there is an electronic stability control intervention. (see Extended Floating Car Data, Section 3.2.1)
- Notification, when a strong deceleration is detected. (see Extended Floating Car Data, Section 3.2.1)

Reduced Network Efficiency

In the absence of application-data-dependent filter criteria for data exchange, it has to be assumed that the filtering that can be applied at CSE level (filterPosition2 and filterPosition3, see Section 5.2.2) only enables insufficient tailoring of the application data exchanged (see Section 5.3.2).

The absence of application-data-dependent criteria for data exchange prevent the subscription to application-data-dependent events such as a threshold exceedance which is only reflected within the opaque structure of a related content attribute. To avoid unwanted loss of such data, the filter must be courser, which in general leads to the transmission and acquisition of more data than required (see Section 5.2.1). Consequently, typically, further filtering at the receiving AE (filterPosition4, see Section 5.2.2) is required. Hence it must be assumed that the actual network bandwidth consumption of the distributed M2M applications is above the theoretical requirements of the use case.

Moreover, limited capabilities to tailor the data acquisition according to the requirements of the consuming AE causes that the actually exchanged data to depend to a great extent on the data providing AE or sensor resolution, and not on the implemented use case. This, for example,

could result in different network utilisations for a single use case, depending on the data providing Node, AE, or container. Particularly against the background of inter-vendor and inter-domain applications where the data provider could conceivably be controlled from another party and might not be known a-priori, this effect is a clear drawback.

In summary, it could be anticipated that all named aspects lead to the reduction of network efficiency for many distributed usage scenarios of the oneM2M service platform in general, and a oneM2M-based ASDP in particular. Hence, it is also a shortcoming with regard to the REQ 6.

Reduced Privacy
Missing application-data-dependent filter criteria for data exchange in general cause the receiving AEs to get more data than required to implement the use case. However, this does not prevent the receiving AE from consuming and analysing the additional data. In this regard, the absence of application-data-dependent filtering capabilities cause privacy concerns.

Thus, with respect to privacy, inappropriate data exchange capabilities of the oneM2M service platform may also lead to a conflicting with legislation. The German Federal Data Protection Act[18], in addition to others, regulates the principle of "data reduction and data economy" in Section 3a (Federal Ministry of Justice and Consumer Protection, 2009): "Personal data are to be collected, processed and used, and processing systems are to be designed in accordance with the aim of collecting, processing and using as little personal data as possible. In particular, personal data are to be aliased or rendered anonymous as far as possible and the effort involved is reasonable in relation to the desired level of protection." Section 19 further requires "provision of information to the data subject": "[…] information shall be provided only in so far as the data subject supplies particulars making it possible to locate the data and the effort needed to provide the information is not out of proportion to the interest in such information expressed by the data subject. […]" (Federal Ministry of Justice and Consumer Protection, 2009). To which extent vehicular data must be considered as personal data depends on several aspects, and as the technical capabilities and usage scenarios evolve, this will continue to be a subject of discussion in society and law (Cacilo et al., 2016). However, in summary it can be stated that inadequate filtering capabilities empower AEs to gain more data than necessary which in general also reduces privacy. Despite the potential of law violations, this fact is quite likely unwanted and may reduce user acceptance of oneM2M-based ASDP in particular, and M2M and IoT solutions in general.

5.4.2 No Aggregation of Subscriptions

The current oneM2M standard deals with every subscription individually. In this regard, different subscriptions to the same resource are not aggregated or harmonised. This applies to

[18] Bundesdatenschutzgesetz

the hosting CSE, as well as to transit CSE(s): At the hosting CSE, notifications are generated and sent individually, as well as transit CSE(s) individually forward the notifications.

Reduced Network Efficiency
This could potentially lead to redundant data transmissions between nodes, as the ASDP scenario, illustrated in Figure 5.13 unveils: Here, five notification messages with the respective latest contentInstance resource are transmitted from the ASN-CSE to the IN-CSE. Only three of these notification messages contain different contentInstance resources: hence, two of five notifications are transmitting redundant data. This diminishes network efficiency of distributed applications, too. It is also a shortcoming with regard to the REQ 6.

5.5 Summary

This chapter delves deeply into application data handling and data exchange capabilities of the current oneM2M service platform, which is of major importance with regard to the feasible functional splits and distribution of AEs and network efficiency.

Based on the reference configuration of the oneM2M service platform for the distributed ASDP concept, firstly a condensed scenario is introduced which is used for the analysis. Afterwards, principle consideration has been given about the source and sinks of application data and the resulting data flows are performed. It has been found that the filterPosition2, which is the hosting CSE of the originating AE is the most appropriate to perform data filtering.

The subscribe/notify mechanism of oneM2M is particularly suitable for distributed event-based systems and the ASDP, and provides decoupling regarding space, time, and synchronisation. At the same time, its oneM2M implementation provides the most comprehensive filtering capabilities of exiting oneM2M data exchange mechanism. For this reason, it is utilised for the implementation of the scenario and the data exchange analysis.

The analysis shows that the totality of filtering capabilities can be considered as the concatenation of three single filter aspects, of which two are directly related to the SN CSF, and one indirect capability through the CMDH CSF. The detailed analysis based on the condensed scenario identifies shortcomings which are the absence of application-data-dependent filter criteria for subscribe/notify mechanism and the lack of subscription aggregations. Both potentially reduce the network efficiency of the distributed application, as it is also the case for the condensed scenario. Furthermore, the absence of application-data-dependent notification criteria may also reduce the privacy since the data acquisition cannot be exactly tailored to the necessities, and this according to the use case typically leads to the reception of more data than essentially required.

For the efficient realisation of distributed functionalities within the oneM2M service platform in general and functionalities related to the automotive ASDP in particular, novel enhanced data exchange capabilities are required. These are introduced in the next chapter.

6 Proposal of Novel Data Exchange Capabilities for the oneM2M Service Platform

The comprehensive analysis of the data exchange capabilities of the oneM2M service platform reveals shortcomings that have a negative impact on network efficiency and privacy of distributed functionalities. This motivates enhancements – the subject of this chapter. Hence, the next section proposes enhancements to address the identified shortcomings and derives related requirements. Afterwards, the approach to implement these requirements within the oneM2M service platform is presented and put into context, by means of related concepts and work. Finally, in advance of a prototypical implementation, two alternative architectural approaches are described and assessed.

6.1 Proposed Enhancements and Derived Requirements

To address the identified shortcomings of existing oneM2M data exchange capabilities, this Section proposes two enhancements: namely, application-data-dependent notification criteria for data exchange with subscribe/notify mechanism and the aggregation of subscriptions. Based on these enhancements, requirements are derived, which enable the enhancements.

6.1.1 Enhancement 1: Application-Data-Dependent Notification Criteria for Data Exchange with Subscribe/Notify Mechanism

The first enhancement that is proposed for the evolution of oneM2M standards is the enabling of application-data-dependent notification criteria for data exchange with subscribe/notify mechanism. It addresses the related shortcoming (see Section 5.4). Accordingly, the enhancement should increase network efficiency and privacy of oneM2M AEs. To implement this enhancement, two requirements are derived.

To facilitate application-data-dependent notification criteria, the opaque application data must be transparent at the level or component (e.g., within the CSE), where the filtering shall be applied. Here, transparent application data refers to the syntax of the application data (i.e., data structure and data types). To enable this, applications (i.e., AEs) shall be capable of providing their application data in a standardised way, including the reference to the specification. This leads to the following requirement:

REQ E.1: Applications (AEs) shall be capable of providing their application data in a standardised way.

The second requirement to enable application-data-dependent notification criteria for subscribe/notify mechanism relates to the policy language for describing the notification criteria. To enable advantageous notification criteria, the policy language shall be very comprehensive and shall support at least basic arithmetical and logical operations on common data types. Further, the policy language shall have the capability of dealing with individual data structures of common data types, reflecting the variability of the oneM2M standard given to the AEs. This is reflected in the following requirement:

REQ E.2: A comprehensive policy language shall be provided to enable the standardised description of gainful application-data-dependent notification criteria including basic arithmetical and logical operations on common data types.

6.1.2 Enhancement 2: Aggregation of Subscriptions

The novel application-data-dependent notification criteria as well as all existing notification criteria work as filter functions, selects a subset of the overall available data from the subscribed-to resource, e.g., a container of an AE. Since different subscriptions to the same resource might not select orthogonal subsets, the aggregation of these subscriptions that is a union of the respective subsets is proposed to further increase network efficiency (see Section 5.2.1). To achieve this, two requirements for two different scenarios can be derived.

Subscriptions that originate in the same local CSE shall be aggregated at this local CSE. Accordingly, the aggregated subscription shall be placed at the remote resource, which also reflects filter position considerations (see Section 5.2.2). This leads to the following requirement:

REQ E.3: Different subscriptions to the same remote resource shall be aggregated at the local CSE.

The hierarchical oneM2M configurations (see Section 4.3.6) may lead to one or more transit CSEs involved in the subscription setup and data exchange. In this regard, the aggregation of subscriptions at transit CSEs may have a high potential for optimisation, since compared to a local CSE, an even greater number of subscriptions typically get handled here. This reflects in the following requirement:

REQ E.4: Different subscriptions to the same remote resource should be aggregated at transit CSE(s).

However, it must be considered that the technical impact of the REQ E.4 is also greater with respect to the fact that the transit CSE would have to interpose the subscription and data exchange message flows, which may have negative implications for (end-to-end) encryption. Further, more AEs or CSEs that are connected via a transit CSE will most likely lead to more (different) subscriptions and data exchange messages that would then need to be handled which may lead to less stable subscription setups. As a result, the organisational overhead may mitigate the potential for subscription aggregations. Nevertheless, similar improvements may be achievable with the REQ E.3, if the transit CSE is substituted by the remote CSE, implementing a back-to-back approach (see Section 4.6.2).

6.2 Approach

This Section presents the general approach(es) to implement the proposed enhancements within the oneM2M service platform. Firstly, the intention is to mainly limit the description of the approaches without considering architectural design decisions or trade-offs that might be necessary. Moreover, the approaches are discussed here at an abstracted technology level, which means the approaches do not considering concrete products for implementing the technologies within a prototype.

As the previous analysis shows, current criteria for data exchange are the concatenation of three filter criteria which in total decide whether a notification is sent or not (see Section 5.3.2). Continuing this, the application-data-dependent notification criteria can be considered and realised as an additional filter (named filterAspectNew), concatenated with the existing filters (named filterCombined), see Figure 6.1.

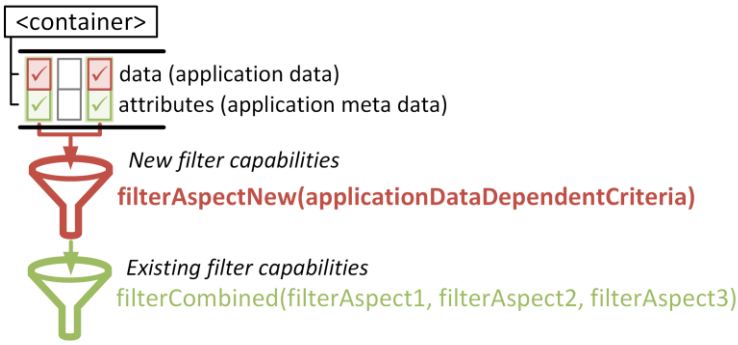

Figure 6.1: Enhanced data exchange capabilities as concatenation of new filter capabilities with existing filter capabilities

In the following, the approaches dealing with how such application-data dependent filterAspectNew can be realised are presented.

6.2.1 XML and XSD to Enable Transparent Application Data

As discussed previously (see Section 5.3.1), the oneM2M service platform currently handles application data in an opaque manner outside of AEs. Nevertheless, to enable interoperability

of AEs across vendors and domains in the context of an IoT as unbounded system (see Sections 3.4.2 and 4.2.1), generic and machine-readable data specifications have to be provided. Hence, the current version of the oneM2M standard already introduces two attributes, namely contentInfo and ontologyRef, which facilitate exchange of syntactic specifications and semantic descriptions of the content (see Section 5.3.1). Although the means to enable semantic interoperability are currently not yet sufficient (see Section 6.3.3), the envisaged enhancements can be enabled solely upon transparent data syntax. This means: It is not necessary to understand the semantics (i.e., the meaning, such as that the value is a temperature) of a content, but it is sufficient to know its data type (i.e., that it is an integer), because logical or arithmetical operations depend solely on the data type (i.e., value is greater or equal to 100).

The oneM2M standard already enables generic description of the content syntax by means of the contentInfo attribute. In particular, if the content is an Extensible Markup Language (XML) document, the combination with its inbuilt reference to an XML Schema Definition (XSD) provides sufficient syntactic information about the elements and structure of the XML document, and thus finally about the content (cf. Salminen & Tompa, 2011). Accordingly, the approach to enable transparent application data utilises XML, which means that it requires a contentType “application/xml:1” (or “application/xml:0”). This addresses the REQ E.1.

6.2.2 Complex Event Processing to Enable Application-Data-Dependent Notification Criteria for Data Exchange With Subscribe/Notify Mechanism

The oneM2M standard does not define the data format or structure of the application data (i.e., of the content attribute). The approach to focus on XML as the content format (see Section 6.2.1) does not define it either but provides the capability for sufficient specification by means of XML and XSD. Nevertheless, the application data exchanged remain heterogeneous and, together with the variety of conceivably useful application-data-dependent acquisition policies, so that a comprehensive but generic approach is required. As a solution, this research proposes the utilisation of Complex Event Processing (CEP) as the enabling technology. It is subsequently briefly introduced and followed by the description as to how it is utilised to address REQ E.2.

Introduction to CEP

The term Complex Event Processing (CEP) was first introduced by Luckham in (2002). Eckert and Bry in (2009) describe CEP as a traditional database system turned upside down: Instead of one-time queries against a finite set of existing data as within traditional database scenarios, in CEP the query is “’standing’ against a (conceptually) infinite stream of events [i.e., data]”(Eckert & Bry, 2009).

CEP and oneM2M: Utilisation of CEP to Enhance Subscribe/Notify Mechanism

The capability of CEP to process events while they are occurring enables its utilisation for notification policies. Following the sense & respond behaviour (Chandy & Schulte, 2010), which is the basic cycle of Event-Driven Architectures (EDA), a CEP-based application-data-dependent notification policy can (Bruns & Dunkel, 2015):

1. **Sense** the event of application data being created (i.e., contentInstance resources within) the subscribed container.
2. **Process or Analyse** the detected event, whether or not it matches the defined policy (i.e., event pattern).
3. **Respond** to the event if the policy is met. Here, this means that this filtering stage is passed, and a notification message is sent, given that other subsequent oneM2M notification criteria (i.e., filterAspect1, filterAspect2, and filterAspect3) that might also be set, are also met.

According to its term, CEP facilitates not only the detection of simple events (e.g., based on logical and arithmetical policies), but also complex events which, are e.g., based on (Bruns & Dunkel, 2015):

- **Sequential policies**, e.g., an event A that must be followed by an event B
- **Sliding windows**, e.g., a time-based or length-based window which is considered
- **Aggregation**, e.g., sum, average, min, max

The complete policy can be built by combinations of the elementary event types or pattern. It must be considered that CEP is first of all a technology and that different CEP implementations may provide slightly different but typically rather more capabilities for specification of event patterns with regard to those named here. As a result, CEP outperforms the minimum requirements of REQ E.2 and provides a comprehensive means to specify application-data-dependent policies.

Consequently, CEP requires a comprehensive language that facilitates the specification of the event patterns. This is the Event Processing Language (EPL), which is an extension of the Structured Query Language (SQL) that is known from traditional database systems for event processing purposes. The EPL is a declarative language to express event processing rules (Bruns, Dunkel, Masbruch, & Stipkovic, 2015). The condition part is used for the description of application-data-dependent notification criteria (cf. step 2: process and analysis of EDA-cycle, see above). The Table 6.1 presents five examples of meaningful application-data-dependent notification criteria in the context of the automotive scenarios used in this research. Example 5 shows that the comprehensive EPL facilitates the description of novel values (e.g., deceleration) as criteria, which are not present in the application data. Accordingly, the subscription to strong deceleration (e.g., greater than 6 m/s^2) requires no modification of the AE vehicle data provider, but only the creation of appropriate enhanced subscription.

Example	Description	EPL Statement
1	Notification, when remaining fuel range is less than 100 km.	SELECT * FROM VehicleData WHERE fuelRange < 100
2	Notification, when heavy rain is detected.	SELECT * FROM VehicleEnvironmentData WHERE rainSensor > 4
3	Notification, when there is a risk of freezing rain.	SELECT * FROM VehicleEnvironmentData WHERE temperature < 3 AND rainSensor > 0
4	Notification, when there is an electronic stability control intervention.	SELECT * FROM VehicleData WHERE ESP=true
5	Notification, when a strong deceleration of more than 6m/s^2 is detected (calculated on speed delta [km/h] and time delta [s]).	SELECT * FROM pattern[a=Position -> b=Position((b.speed - a.speed)/((b.timestamp-a.timestamp)*3.6) < -6)]

Table 6.1: Examples of beneficial automotive application-data-dependent notification criteria and their EPL statement representation.

Related Concepts and Work

The utilisation of CEP to implement enhanced data exchange capabilities for the oneM2M service platform constitutes a novelty. However, related concepts and work can be found.

Papageoriou et al. in (2013) investigates "smart m2m data filtering using domain-specific thresholds in domain-agnostic platforms", which does not include the capability of filtering on M2M devices and does not use EPL to describe filters respectively CEP to enforce them. Also Bruns et al. in (2015) considered the usage of CEP for M2M. They also emphasised its advantages for the detection of meaningful situations and events, considered the mapping or adaption of the event model, but again performed CEP only on the incoming unfiltered raw events in the backend.

In-network CEP in (wireless) sensor networks is considered by Saleh et al. in (Saleh, 2013; Saleh & Sattler, 2013) which shows significant improvements regarding network requirements and energy efficiency of the sensors because of reduced communication.

Wang in (2013), among other things, investigates approaches for privacy-preserving within CEP via event suppression, which could be used for enhanced privacy considerations of application-data-dependent notification policies with EPL statements. Furthermore, research of Cugolar et al. in (2012) about deployment strategies for distributed CEP could provide guidance for optimised usage of the proposed enhanced data exchange capabilities towards overall optimisation.

6.2.3 Aggregation of Subscriptions Enabled Through Criteria Unification

Basically, the aggregation of subscriptions means the unification of their notification criteria. As a result of the mathematically union of notification criteria, the resulting subset of the overall available data set is also the union of all subscriber subsets. For the existing filter aspects of oneM2M (i.e., filterAspect1, filterAspect2, and filterAspect3), the unification of criteria has to be manually implemented. For the filterAspectNew that is based on EPL language constructs, this language may support unification (e.g., diverse variants of joins are normal language construct in the database context). Further, different filter criteria can also be combined with a logical OR which is used for this approach. The following listing provides an example:

```
SELECT * FROM VehicleData
WHERE ESP=true;
// aggregated with
SELECT * FROM VehicleData
WHERE fuelRange < 100;
// can be unified to
SELECT * FROM VehicleData
WHERE ESP=true OR fuelRange < 100;
```

Figure 6.4 presents a flowchart for the implementation of subscription aggregations as part of the subscription create or update at the local CSE[19]. This addresses the REQ E.3. The functionalities can be integrated into the existing methods of the SN CSF.

Figure 6.3 presents a possible flowchart for the implementation of subscription aggregations as part of the subscription forwarding at transit CSE(s)[19]. It shows that the implementation of the REQ E.4 is basically feasible. The functionalities have to be integrated into the existing CMDH CSF where related methods need to be extended accordingly. However, oneM2M configurations with transit CSEs have not been given consideration within the scenarios of this research. Hence, REQ E.4 is not further investigated in the following.

[19] Deletions or expirations of subscriptions have to be implemented similarly to subscription update but are not dealt with here.

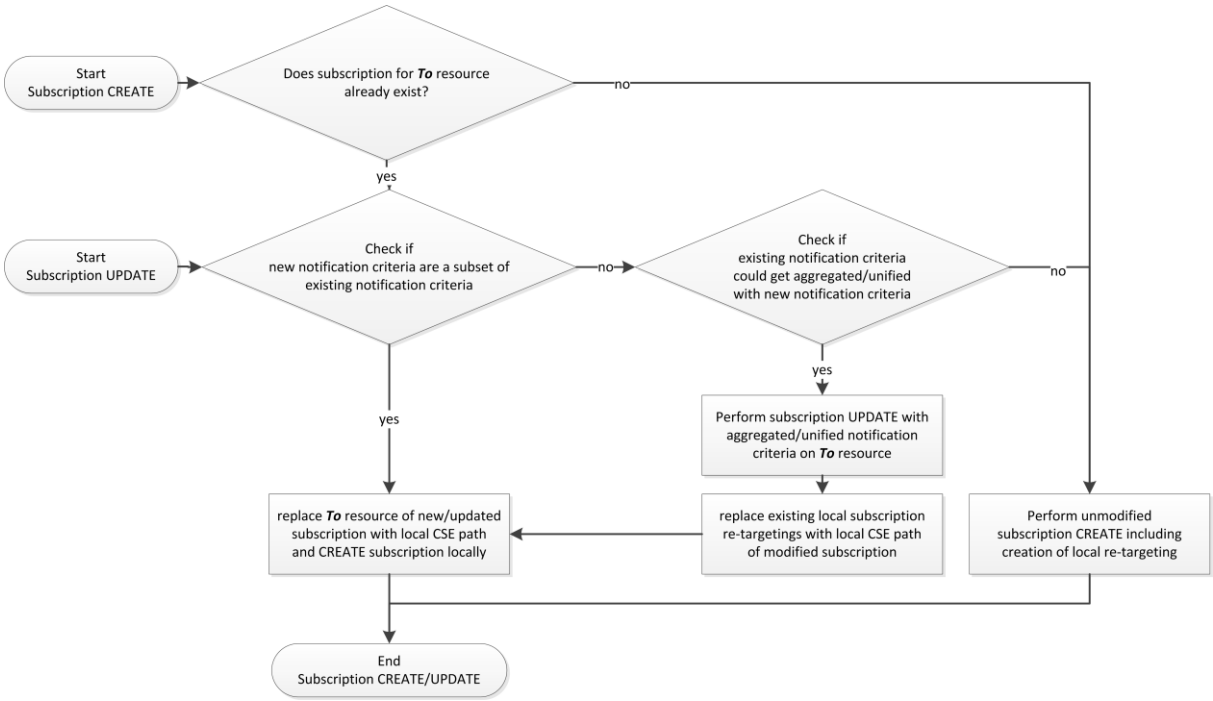

Figure 6.2: Flowchart for aggregation of subscriptions at local CSE

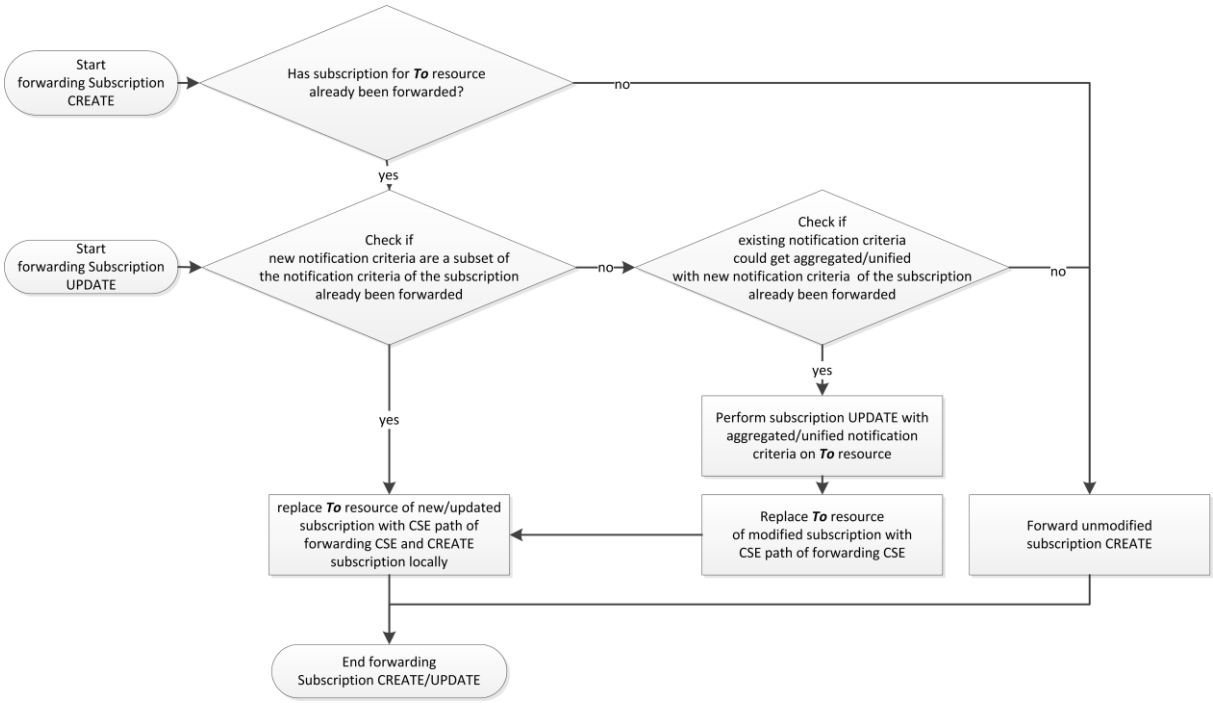

Figure 6.3: Flowchart for aggregation of subscription at transit CSE

6.3 Alternative Architectural Approaches

Different architectural approaches are possible to implement the proposed enhancements. However, the implementation of the enhancements within each single AEs is neither reasonable nor feasible due to the following reasons: REQ E.3 (and REQ E.4) cannot be appropriately fulfilled since the enhancements are individual AE parts that share no common view on the data

containers and subscriptions. Further, it would be inefficient to individually implement the enhancements in every AE, because this leads to the duplication of a significant number of functionalities. Finally, this would also violate the oneM2M design principle to implement functionalities that are shared across many AEs commonly (see Section 4.3.1), which in the end also reduces interoperability (see Section 4.2.1) due to the lack of standardisation.

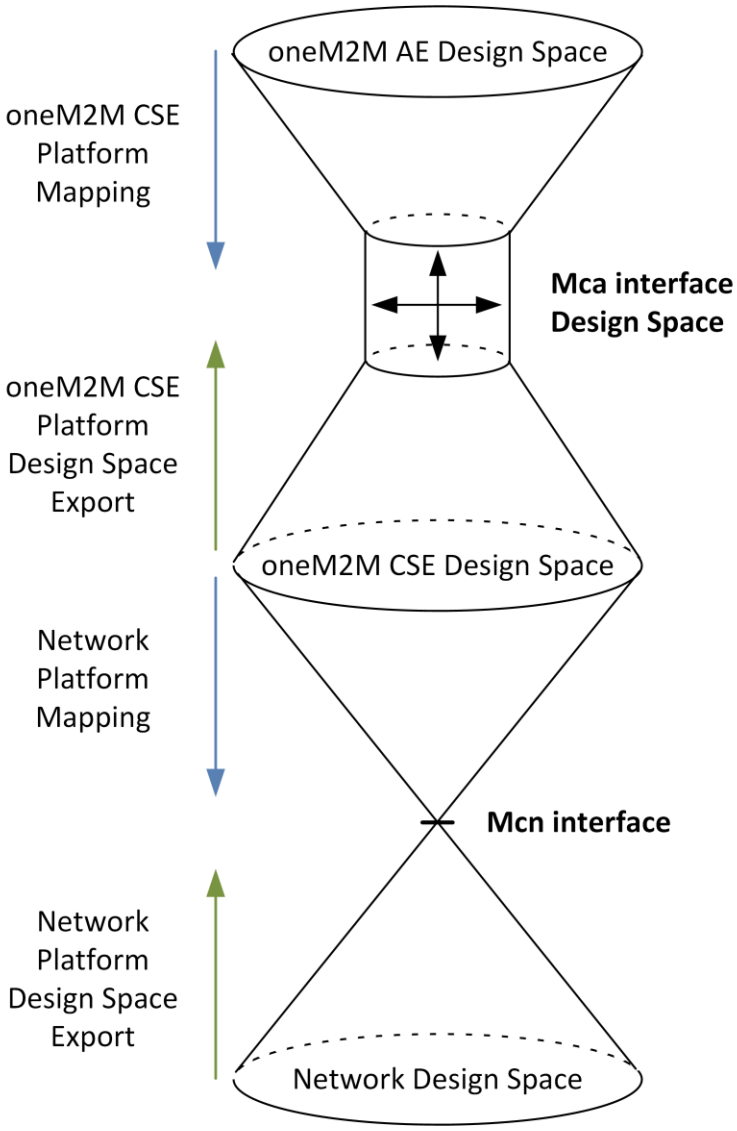

Figure 6.4: Design space for Mca interface

Nevertheless, with respect to the Platform-Based Design approach (see Section 2.4.5), the central architectural design decision is, whether or not the enhancements are implemented within the CSE layer. In more detail, the design decision is whether the Mca interface exposes the proposed enhancements to the AEs, or if the enhancements shall be implemented above the Mca interface, i.e., within the AE layer (but outside of the AEs). This architectural design decision regarding the capabilities provided by the Mca interface is illustrated in Figure 6.4.

In order to facilitate an adequate architectural design decision about these two alternative architectural approaches, they are described below in more detail and evaluated by use of the previously-introduced scenario (see Section 5.1) which, however, is now modified to reflect the enhanced capabilities provided. Afterwards, these two architectural alternatives and respective architectural trade-offs are discussed, and an architectural design decision is made in advance to a prototypical implementation as proof of concept. In addition to the investigated scenario, the anticipated future developments of oneM2M are also considered, particularly those necessary to gain full semantic interoperability.

6.3.1 Approach 1: Within oneM2M AE Layer

This architectural approach implements the proposed enhancements above the Mca interface. In order to implement the enhancements still not within the single AEs, an intermediate layer within the M2M AE layer of each Node is introduced, referred to as Enhanced Data Exchange Layer (EDEL). The Figure 6.5 illustrates the concatenation and interaction of new application-data-dependent filter aspect with existing filters.

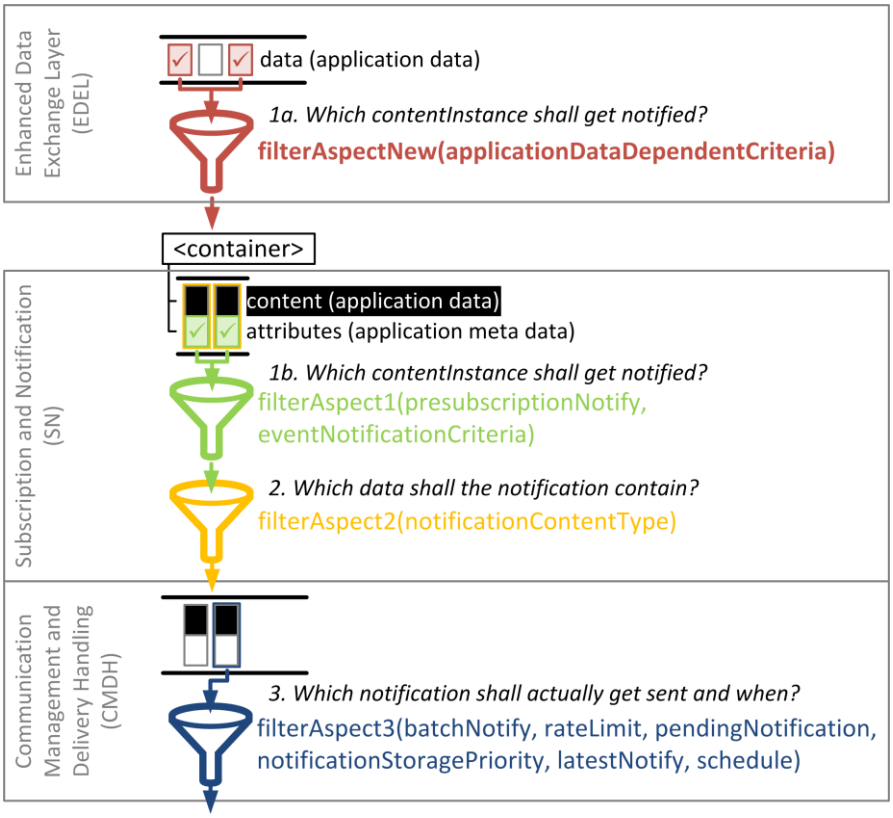

Figure 6.5: Concatenation and interaction of new application-data-dependent filter aspect with existing three filter aspects of subscribe/notify mechanism for alternative architectural approach 1

The EDEL interposes the subscribe/notify and the container-related communication primitives between the AEs and the local CSE. For this purpose, the EDEL exposes enhanced versions of the underlying interface Mca, referred to as Mca-E (see Figure 6.6). The application-data-dependent filtering is performed within the EDEL, where the data structure remains transparent

through the utilisation of the syntax-specifying attribute contentInfo provided by the AEs, according to the described approach (see Section 6.2.1).

For the scenario-based evaluation of this architectural alternative, the previously-introduced scenario is consulted again (see Section 5.3). However, at this point, the enhanced data exchange capabilities provided to the AEs have to be reflected to evaluate their benefits. In this regard, a criterion derived from the use case 'Extended Floating Car Data' is used instead of an unspecific time constraint for the notification criteria of AE_2. In this example, this criterion is a remaining fuel range below 100 km. Afterwards, the AE_3 'Vehicle Maintenance' also creates a subscription to the container VehicleData with the criteria(AE_3) that requests notifications when an interference of the ESP occurs (ESP=true). At the time of the second subscription Create, the EDEL detects that a subscription to the same remote resource already exists and aggregates the two criteria (denoted criteria(AE_2,AE_3)). The AE_2 and AE_3 work against the local stubs of their subscriptions, located within their local EDEL.

All communication and data exchange between those two EDELs can be realized with existing resource capabilities of the M2M CSE: For example, the EDEL itself can register at the unmodified oneM2M CSEs as an AE. The exchange of the enhanced data exchange policies can then be performed proprietarily between those EDEL by use of standard data exchange mechanisms of the current oneM2M standard. Figure 6.6 merely illustrates the final result of these setup procedures and the consequent data exchanges. As the figure shows, with this approach the data is already provided filtered to the unmodified oneM2M CSE.

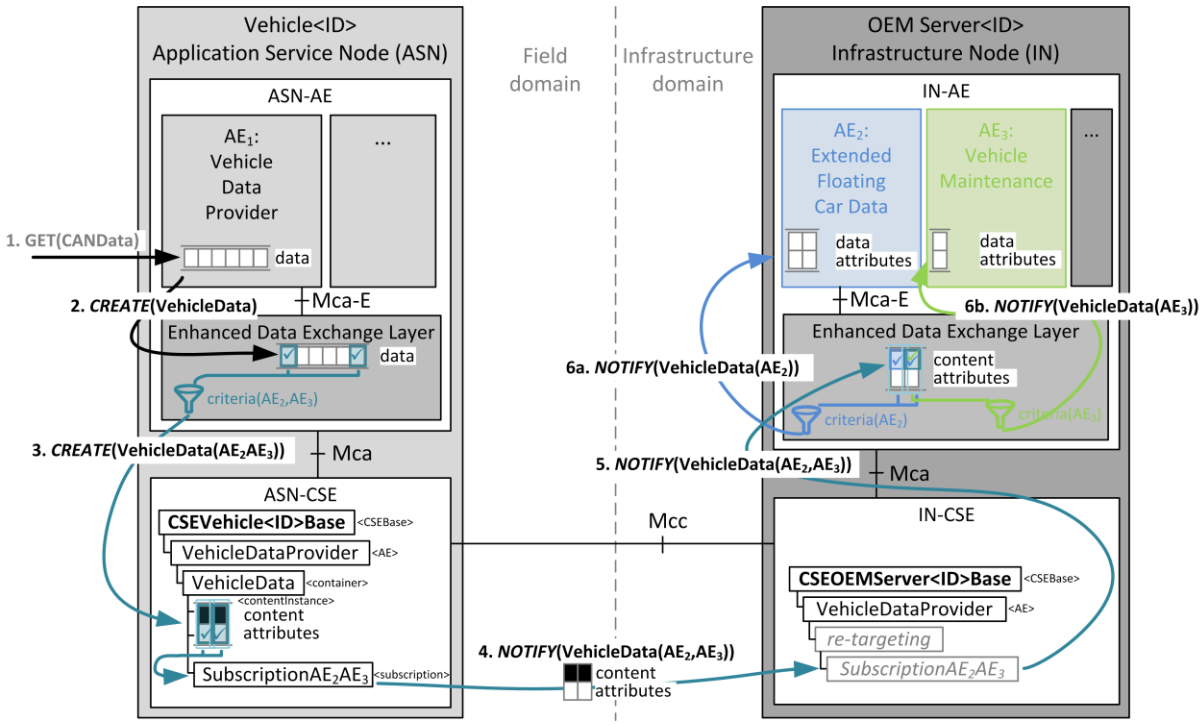

Figure 6.6: Enhanced data exchange with alternative architectural approach 1

The resulting data exchanges are described in the following:

- Step 1. GET(CANData):
  The AE1 proprietary obtains the vehicle data from the in-vehicle CAN-bus.
  This step is performed six times within this example, generating the six vehicle data samples, as illustrated.
- Step 2. CREATE(VehicleData):
  The AE1 provides this data without filtering (cf. Section 5.2.2) via the Mca-E interface to the EDEL. Thereby the VehicleData is transformed to a vendor-independent representation, such as application/xml (cf. Section 5.1).
  This step is also performed six times within this example, generating the six data instances within the EDEL as illustrated.

The EDEL checks at every data-instance creation event whether the notification criteria of the existing subscriptions are met and whether the application data sample shall be provided to the ASN-CSE. Here, the EDEL provides the enhanced data exchange capabilities, as proposed.

- Step 3. CREATE(VehicleData(AE_2AE_3)):
  If the application-data-dependent aspect of notification criteria(AE_2,AE_3) of the subscriptionAE_2AE_3 is fulfilled, the data is provided to the VehicleData container within its local ASN-CSE as new contentInstance.
  This step is performed two times, generating two contentInstance resources including VehicleData, as illustrated.

The ASN-CSE checks at every contentInstance creation event whether the notification criteria of existing subscriptions are met and whether a notification shall actually being sent (cf. SN and CMDH, see Section 5.3.2). This utilises the existing oneM2M data exchange capabilities only.

- Step 4. NOTIFY(VehicleData(AE2,AE3)):
  When the notification criteria of the subscriptionAE2AE3 is fulfilled, the related notification is sent. This policy leads to no additional filtering here. Hence two single notification messages are being sent from the ASN-CSE to the IN-CSE.
- Step 5. NOTIFY(VehicleData(AE2,AE3)):
  Each of the two single notification messages is re-targeted at the IN-CSE to the (server-capable) EDEL. The VehicleData contentInstance resources are stored at the IN-CSE within the corresponding resource.

The EDEL checks at every contentInstance creation event to determine which of the existing subscriptions are met (according to the application-data-dependent criteria) and triggers the related notification messages.

- Step 6a. NOTIFY(VehicleData(AE2)):
  When the notification criteria of the subscription of AE2 is fulfilled, the related notification is sent. This policy leads the selection of two of two VehicleData contentInstance resources which are sent from the EDEL to the (server-capable) AE2 as single notification messages.
- Step 6b. NOTIFY(VehicleData(AE3)):
  When the notification criteria of the subscription of AE3 is fulfilled, the related notification is sent. This policy leads the selection of one of two VehicleData contentInstance resources which is sent from the EDEL to the (server-capable) AE3 as single notification message.

Both AE2 and AE3 consume the vehicle data (and attributes, if applicable) according to their use case without additional filtering, i.e., discarding of received vehicle data.

6.3.2 Approach 2: Within oneM2M CSE layer

An alternative architectural approach is the enhancement of the oneM2M service platform itself. This means that additional functionalities are implemented within the CSE. This makes modifications of the related interfaces Mca and Mcc necessary to provide the enhanced capabilities. Accordingly, they are referred to as Mca-E and Mcc-E. Hereby, the interface enhancements could be implemented to preserve downwards compatibility, i.e., the enhancements are optional usage patterns, not preventing the interface usage in accordance to the current standard.

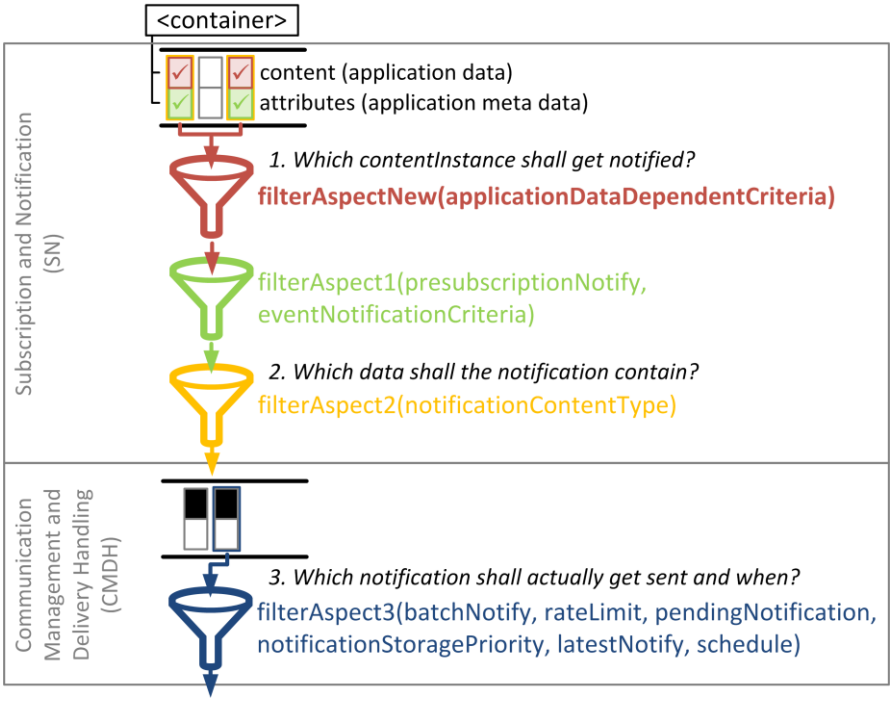

Figure 6.7: Concatenation and interaction of new application-data-dependent filter aspect with existing three filter aspects of subscribe/notify mechanism for alternative architectural approach 2

The previously introduced scenario is also used for the evaluation of this architectural alternative (see Section 5.3). Similar to the previous alternative, at this point, the enhanced data exchange capabilities provided to the AEs have to be reflected to evaluate their benefits. Hence, for the AE_2 the notification criterion "remaining fuel range below 100 km" is applied instead of an unspecific time constraint. Afterwards, the AE_3 'Vehicle Maintenance' again creates a subscription to the same container VehicleData with the criteria(AE_3) that requests notifications when an interference of the ESP occurs (ESP=true). The existence of a subscription to the same remote resource is now detected directly at the CSE, which aggregates the two criteria (denoted criteria(AE_2,AE_3)). The AE_2 and AE_3 work against the local stubs of their subscriptions, which are here located here within their local CSE.

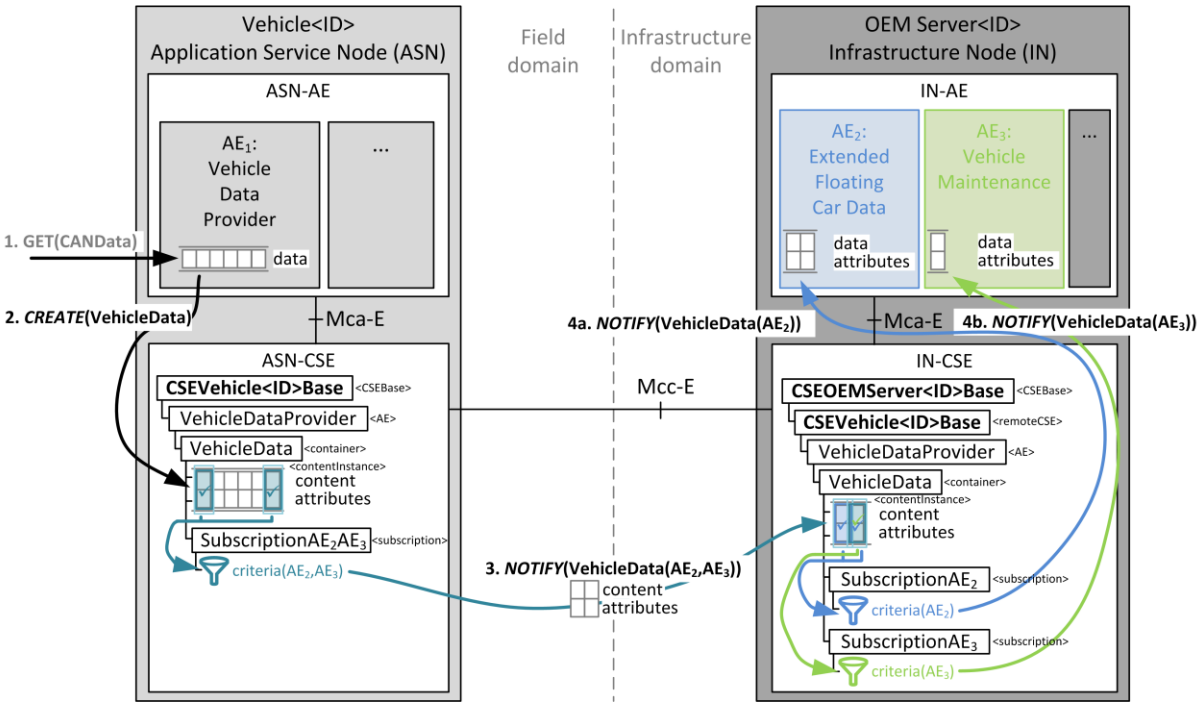

Figure 6.8: Enhanced data exchange for alternative architectural approach 2

With this approach, data is still provided to the CSE without pre-filtering at its original resolution. The existing filterCriteria resources and meta-data filter capabilities are merely enhanced with the proposed capabilities of facilitating application-data-dependent policies and aggregation according to the proposed enhancements. This results in the following steps for the exchange of application data by means of this enhanced subscribe/notify mechanism:

- Step 1. GET(CANData):
  The AE1 proprietary obtains the vehicle data from the in-vehicle CAN-bus.
  This step is performed six times within this example, generating the six vehicle data samples, as illustrated.
- Step 2. CREATE(VehicleData):
  The AE1 provides this data without filtering (cf. Section 5.2.2) as new contentInstance

resource to the VehicleData container within its local ASN-CSE. Hereby, the VehicleData is transformed to a vendor-independent application/xml representation (cf Section 5.1)
This step is also performed six times within this example, generating the six contentInstance resources including VehicleData, as illustrated.

The ASN-CSE checks at every contentInstance creation event whether the notification criteria (including application-data-dependent criteria) of existing subscriptions are met and if a notification shall actually being sent (SN including enhancements and CMDH, see Section 5.3.2).

- Step 3. NOTIFY(VehicleData(AE2,AE3)):
  If the notification criteria of the subscriptionAE2AE3 is fulfilled, the related notification is sent. This policy leads to a filtering of two of six VehicleData contentInstance resources which are being sent from the ASN-CSE to the IN-CSE as single notification messages. The VehicleData contentInstance resources are stored at the IN-CSE within the corresponding resource.

The IN-CSE checks at every contentInstance creation event whether the notification criteria (including application-data-dependent criteria) of existing subscriptions are met and whether a notification shall actually being sent (SN including enhancements and CMDH, see Section 5.3.2).

- Step 4a. NOTIFY(VehicleData(AE2)):
  When the notification criteria of the subscriptionAE2 is fulfilled, the related notification is being sent. This policy leads to the selection of two of two VehicleData contentInstance resources which are sent from the IN-CSE to the (server-capable) AE2.
- Step 4b. NOTIFY(VehicleData(AE3)):
  When the notification criteria of the subscriptionAE3 is fulfilled, the related notification is sent. This policy leads to a selection of one of two VehicleData contentInstance resources which are sent from the IN-CSE to the (server-capable) AE2.

Both AE2 and AE2 consume the vehicle data (and attributes, if applicable) according to their use case without additional filtering, i.e., discarding of received vehicle data.

6.3.3 Excursion: Towards (Full) Semantic Interoperability in oneM2M

The ultimate goal of M2M considerations is the interconnection of machines without human intervention. A simple example is that the light switch or dimmer should be able to control lights from different vendors, whose communication technology or interface might not be

known during development of the switch or dimmer. An enhanced example is that someone may be in a room and simply say "I'm freezing": A microphone from the smart home system shall captures the speech and should be able to derive related information and control commands such as the current location (i.e., room) of the person, discovers devices related to the room's temperature (e.g., a thermostat) finally give the command to increase the temperature set. In the automotive context, electric vehicles should be able to communicate with manifold charging infrastructure and vendors. In ITS scenarios, traffic management may be interested in the average speed on a certain route that should utilise every connected vehicle on that route, independent of its vendor. This information may be provided by an automatically generated mash-up AE or service, utilising the speed of the vehicles that are currently driving on that route in combination with an average calculation service.

Such scenarios, which are usually intended as M2M or IoT solutions are referred to as "smart" (e.g., Smart Home, Smart Grid, Smart City), and require that AEs and nodes are able to gain (full) semantic interoperability. Full semantic interoperability includes semantics of the application data and is not limited to interface semantics (i.e., CRUD methods). A common approach to achieve semantic interoperability is standardisation of data formats which is then explicitly implemented within each AE, node, etc. during development time. However, considering the great number of M2M nodes that should be interoperable, the lifetime of devices, the evolution of standards and technology, this common methodology reach its limits. Instead, generic methodologies that are able to gain interoperability automatically during runtime might become necessary. Coulouris et al. in (2012) described this problem area to the point: "Often the semantics of operations may vary as well as the syntax, and overcoming semantic incompatibility is in general difficult and error-prone. Then there is the scale of the problem: if there are N interfaces, then potentially N2 adaptors have to be written, and more and more interfaces will be created over time. Moreover, there is the question of how components are to acquire suitable interface adaptors as they reassociate in a volatile system. Components (or the devices that host them) cannot come preloaded with all possible N2 adaptors, so the correct adaptor has to be determined and loaded at runtime."

The oneM2M service platform so far only unifies the communication between nodes and applications through the abstraction from specific communication technologies and protocols. The RESTful interface design leads to interoperability at communication level with respect to interface (i.e., connector) semantics which are the CRUD+N methods (see Section 4.4.4). Furthermore, the standardised oneM2M resources ensures basic syntactic interoperability at the level of oneM2M resource types, e.g., AE, container, subscription. However, the current release of oneM2M lacks (full) semantic interoperability between AEs, which still requires "a-priori agreement between the two [AEs] regarding the meaning, i.e., the semantics of the data, which is then implicitly coded into the producer and the consumer." (oneM2M TR-0007, 2015). Moreover, although technically (i.e., syntactically) accessible, the meaning and hence capabilities of the AEs, containers, nodes or their services which are exposed through URIs is

not clear. Fielding stated in (Fielding, 2000) with respect to the RESTful architectural style and semantics: "Semantics are a by-product of the act of assigning resource identifiers and populating those resources with representations. At no time whatsoever do the server or client software need to know or understand the meaning of a URI – they merely act as a conduit through which the creator of a resource (a human being naming authority) can associate representations with the semantics identified by the URI. In other words, there are no resources on the server; just mechanisms that supply answers across an abstract interface defined by resources. It may seem odd, but this is the essence of what makes the Web work across so many different implementations."

Certainly, the generality and simplicity of a RESTful interface enables technical interoperability between the myriad computers, building the "traditional" Internet. However, it has to be considered that this refers to an Internet made for humans, who are capable of deriving the semantic information and manually creating the connection between resources through implementing a-priori knowledge. For example, a human being may gain a fundamental understanding of the data provided by an URI because of its naming, see URI1 example below. Together with documentation of the AEs, containers, etc., human beings can develop interconnected functionalities. But it must be considered that, towards an Internet made for machines (i.e., M2M and IoT), machines must be able to autonomously gain full semantic interoperability, even on demand during runtime.

URI1: /CSEVehicle1Base/VehicleDataProvider/VehicleData/VehicleDataInstance1/content
URI2: /CSEBase/AE/container/contentInstance/content
URI3: /AB/CD/EF/GH/content

With respect to the current status of the oneM2M standards, the machine-readable information for the URI1 remains on the level of resource types (see URI2). Since the resource naming typically has no meaning to machines, in general the URI3 provides the same amount of information to them as the URI1. This is in accordance with the RESTful architectural style as cited above.

These examples indicate the necessity of enhancements of the oneM2M standard to gain full semantic interoperability between AEs and nodes of different venders and domains. In this regard, semantic annotation of resources together with ontologies are considered as an enabling technology (Alaya, Banouar, Monteil, Chassot, & Drira, 2014; Ben Alaya et al., 2015; Colace et al., 2015; Daniel & Matera, 2014). Ben Alaya et al. propose an IoT-O ontology facilitating the description of the service, actuation, actuator, sensor, and observation model for a thing (Ben Alaya et al., 2015). Such semantic descriptions could be integrated into the oneM2M resources by reference (i.e., through a link to the providing resource) or inline, where the description is stored within an attribute of the resource (Ben Alaya et al., 2015). In combination with the integration of appropriate technologies such as semantic reasoning, novel resource discovery methodologies and dynamic resource creations as mash-ups are enabled: For

example, resources can be discovered depending on their capabilities, as described in the smart home example above, where the thermostat of the room may be identified by its capability of controlling the room's temperature. Here, discovery besides matching may also include ranking of possible service candidates and the selection of one (Domingue, Fensel, & Hendler, 2011, p. 982). Dynamic resource creation combines such discovery capabilities with mash-up approaches where, e.g., manifold vehicle speeds are combined with a resource or service that is able to calculate averages. This enables the ITS example from above, where the oneM2M platform is capable to dynamically built a resource that provides the average speed on a certain road segment.

Although the approaches to gaining (full) semantic interoperability within the oneM2M service platform to a certain extend are still part of research and standardisation, the necessity for such capabilities is widely agreed upon (Ben Alaya et al., 2015; oneM2M TR-0007, 2015). In this regard, respective enhancements of the oneM2M standard can be anticipated for future releases.

6.3.4 Assessment

In advance of a prototypical proof-of-concept realisation, the two alternative approaches are assessed considering the architectural trade-offs to be made. Besides technical aspects of the architectural design decision, the assessment also has to consider the anticipated future enhancements of the oneM2M standard such as the semantic interoperability (see Section 6.3.3) or enhanced access rights to improve privacy[20]. Furthermore, the assessment should consider the philosophy that stands behind the oneM2M standardisation efforts.

Both approaches basically fulfil the identified requirements REQ E.1 and REQ E.2, by which the implementation of application-data-dependent notification criteria is facilitated. They provide the same enhanced data exchange capabilities including application-data-dependent notification policies to the AEs through the enhanced Mca-E interface when a deployment scenario with two nodes is considered, similar to the ASDP scenario as utilised here.

Besides, REQ E.3 can also be realised with both alternatives. However, considering oneM2M deployment scenarios with deeper hierarchies where data exchange between AEs will pass more CSEs than the respective up to two hosting CSEs, functional capabilities of the two alternative approaches differ: Since forwarded messages (e.g., related to subscription and data exchange) probably will not reach the EDEL of intermediate CSEs, inter-node-AE aggregation at intermediate CSEs (REQ E.4) is not possible with approach 1. However, as already discussed, this specific situation is not being further considered within this research, although

[20] The implementation of enhanced access rights for the oneM2M service platform is beyond the scope of this research. Nevertheless, they are considered as a meaningful step to improving privacy. Nevertheless, since enhanced access rights are enabled through the proposed enhancements for data exchange, they are another aspect for the assessment of the architectural trade-off with respect to the two alternative approaches.

it also might be an argument for approach 2, where aggregation of data subscriptions and data exchange across nodes could be implemented.

However, also with respect to the wireless cellular network utilisation both approaches are equal. This shows the considered scenario where in both cases only three notification messages including the most recent contentInstance resource are transferred between the ASN-CSE and the IN-CSE. Nevertheless, although the results for AEs are equal, the architectural approaches are fundamentally different, not only because of the technical realisation, but also with respect to the philosophy behind them.

Since enhanced functionalities shall be provided to the AEs, which are exceeding the existing capabilities of the oneM2M service platform, the AEs in both cases have to utilise an enhanced interface (Mca-E). Nevertheless, the architectural approach 1 (see Section 6.3.1) is technically more complex: It requires the subscriptions to be maintained twice – at the EDEL and within the CSE. Accordingly, significant functionalities, such as the CSF SN must be implemented twice. Besides the few attributes that are different within the event notification criteria, the data to be stored and maintained at the EDEL and the CSE is quite similar. Moreover, the approach 1 requires also the doubling of access criteria management and enforcement which not only causes doubling of functionalities but may also be unwanted with respect to security considerations. Due to the above-named reasons, where functionalities are duplicated in both layers (EDEL and CSE), the architectural approach 1 is assessed as technically more complex. Both approaches extend the existing capabilities of oneM2M. Thus, it must be assumed that this increases the memory and computational requirements in case of approach 1 of the AE layer (containing the EDEL), and in case of approach 2 of the CSE layer. However, it must be assumed that approach 1 in total has a higher memory and computational requirements than the approach 1 because of less efficient implementation capabilities (looser coupling because of intermediate reference points), including duplicated functionalities.

Since both approaches increase memory and computational requirements and because the oneM2M standard currently describes functionalities which are not to be considered optional, this results in increased minimum hardware requirements. Given that the AE layer and the CSE layer are running on the same node (i.e., IN, MN, and ASN), as also proposed for the oneM2M-based ASDP, this may decrease the scalability to more constraint nodes. Following the prior considerations, the approach 1 may thereby decrease the scalability more than approach 2. In a oneM2M configuration, where the AE layer and the CSE layer are not hosted on the same node, the two approaches only affect the hardware requirements of those nodes hosting the respective layer where the enhancements are implemented: Accordingly approach 1 increases the hardware requirements to ADNs and ASNs, and to MNs or the IN, if an AE (or the AE layer containing the EDEL) is present. Approach 2 increases the hardware requirements of ASNs, MNS, and the IN. However, it could be assumed that the most constraint nodes will be realised

as ADN, where the CSE is not co-located. From this point of view, approach 1 may again have higher negative implications for scalability towards constraint nodes compared to approach 2.

Another aspect is the handling of application data at the CSE layer. So far, the actual application data is handled opaquely at the CSE (see Section 5.3.1). With approach 1, this can be continued. However, approach 2 requires that the application data becomes transparent at the CSE to enable the application-data-dependent notification criteria. Opaque application data at the CSE layer might be assessed as beneficial with respect to encapsulation and information hiding considerations. At the same time, from the perspective of the OSI reference model, it must be considered that the CSE layer already works on the level of the application layer. In this context, it must be considered, where the architectural design decisions resulting in opaque application data at CSE have their origin and what will occur to this decision in anticipation of envisaged enhancements of the oneM2M standard. This question not only has a technical dimension, but also an organisational dimension regarding the oneM2M standardisation activities. Towards a standardised platform for the interconnection of machines across vendors and domains, oneM2M at first unifies the communication with respect to communication technologies and protocols (see Section 4.4.5). For this first step, standardisation of application data (description) was out of scope, and opaque application data and subscriptions that cannot be referred to application data is a consequence (see Sections 5.3.1 and 6.3.3). Then again, oneM2M aims full semantic interoperability to enable functionalities perceived as smart (see Section 6.3.3). In this context, besides others, dynamic mash-ups, which result in the creation of virtual resources (e.g., a virtual AE that provides a data container providing the average speed on a certain route) are requirements for oneM2M standards' respectively CSE enhancements. To enable the CSE to build such virtual mash-up resources, application data must be transparent at the CSE level, even up to its semantic description. Hence, making the application data a white box through utilisation of its syntactic description, is considered as anticipation of reasonable upcoming oneM2M standards' evolvements. Thus, this is not assessed as a counterargument of approach 2.

Alone these oneM2M standardisation objectives provide further assessment criteria for the architectural trade-off between the two alternative approaches: The oneM2M standardisation activities aim to overcome fragmentation of solutions arising from technologies, protocols, and domain-specific or vendor-specific approaches. As a solution, the reference points and the CSE layer are standardised. The range of functionalities by the latter shall be driven by the question, as to whether a functionality is to be shared by or is advantageous for many AEs (across many vendors and domains). As discussed earlier, the proposed enhancements are considered as beneficial beyond an ASDP. If so, the functionalities should be part of oneM2M standardisation because otherwise AEs have to again agree on several standards not harmonised or integrated. This again fosters fragmentation and hence weakens the positioning of the oneM2M standardisation activities as a means to overcome this challenge. Accordingly, approach 1 without standardisation of enhancements by oneM2M is assessed as a less beneficial trade-off.

Approach 2 inevitably has to be part of the oneM2M standardisation activities because it extends the CSE. As previously discussed this should also apply to the approach 1. If so, proposed enhancements and existing functionalities are maintained by oneM2M. In this regard, oneM2M can choose the most efficient architectural approach. This is approach 2, where functionalities must not be duplicated.

To summarise: Approach 2 is the more efficient way to implement the proposed enhancements within the oneM2M service platform. It constitutes the logical continuation of the oneM2M standardisation philosophy and anticipates future enhancements, e.g., to gain full semantic interoperability between AEs and nodes. Clearly, like most functional enhancements, also those proposed here might also increase the hardware requirements for nodes, running the CSE. Then again, against the background of the resulting improvements as well as existing and anticipated future functionalities, the additional amount of memory and computation capabilities is assessed as reasonable. Approach 1 is assessed as less efficient and contrary to the oneM2M objectives. However, if oneM2M does not intend to implement the proposed enhancements, approach 1 might be the best alternative, to not implementing these enhancements in every single AE. In conclusion, the two presented alternative approaches do not mutually exclude one another – a combination wherein certain fundamental enhancements are implemented at CSE level that are further extended with a less-common building block at application layer might be possible. Nevertheless, the prototypical implementation shall utilise the approach 2, where the enhancements are made in the related CSFs of the CSE layer.

6.4 Summary

The current data exchange capabilities of the oneM2M service platform have shortcomings which can result in reduced network efficiency and privacy of distributed applications. Novel enhancements for the data exchange capabilities for the subscribe/notify mechanism of the oneM2M service platform are proposed to address these shortcomings. These are application-data-dependent criteria for data exchange and the aggregation of different subscriptions to the same remote resource.

The approach consists of three building blocks to address the four requirements, derived from the two named enhancements. XML and XSD are used to enable AEs to provide their application data in a standardised way including syntactic specification. Complex Event Processing technologies and the related Event Processing Language are proposed to facilitate comprehensive application-data-dependent notification policies and to enforce them within the subscribe/notification mechanism. CEP, respectively the EPL, also provide means for the aggregation of different notification policies. In addition, related flowcharts for the aggregation of subscriptions at the local CSE and the remote CSE are presented, whereas the latter is not further investigated in this research.

The approach describes the technologies and fundamentals on how the enhanced data exchange capabilities can be realised. However, there are different architectural approaches which are capable of implementing the proposed enhancements within the oneM2M service architecture. With respect to the Platform-Based Design methodology, in general two different architectural design decisions can be taken, although combination might be possible. The central decision is whether the enhancements are realised within the oneM2M AE layer or within the oneM2M CSE layer. Both approaches are introduced and discussed by means of the known condensed scenario. This facilitates the final assessment of the architectural trade-off, which also takes into account the oneM2M philosophy, design principles, objectives, and anticipated developments. It is clearly explained why the approach 2 of implementing the enhancements within the oneM2M CSE layer is considered as the more appropriate and thus intended solution. Its practical feasibility is shown through a prototypical proof-of-concept implementation, which is subject of the next chapter.

7 Prototype Implementation

The applicability of the proposed enhancements for data exchange through the subscribe/notify mechanism and the appropriateness of the selected architectural alternative was validated by means of a prototypical implementation as proof of concept. This chapter introduces this prototype and related considerations, starting with the presentation of the selected technologies. Afterwards, the building blocks are described that implement the enhancements and integrate them into the oneM2M CSE, respectively the OM2M prototype. Finally, the efficiency improvements enabled through the enhanced data exchange capabilities are assessed through exemplary experiments and estimations, followed by concluding considerations.

7.1 Technologies

Basically, two technologies or projects build the basis for the prototype: The Eclipse OM2M project (Eclipse OM2M project, 2016) is used for basic functionality of the oneM2M service platform. The Esper CEP Engine (EsperTech, 2016a) is integrated into OM2M as additional core plugin to facilitate the realisation of CEP-based functionalities, see Figure 7.1. The technologies are introduced in the following.

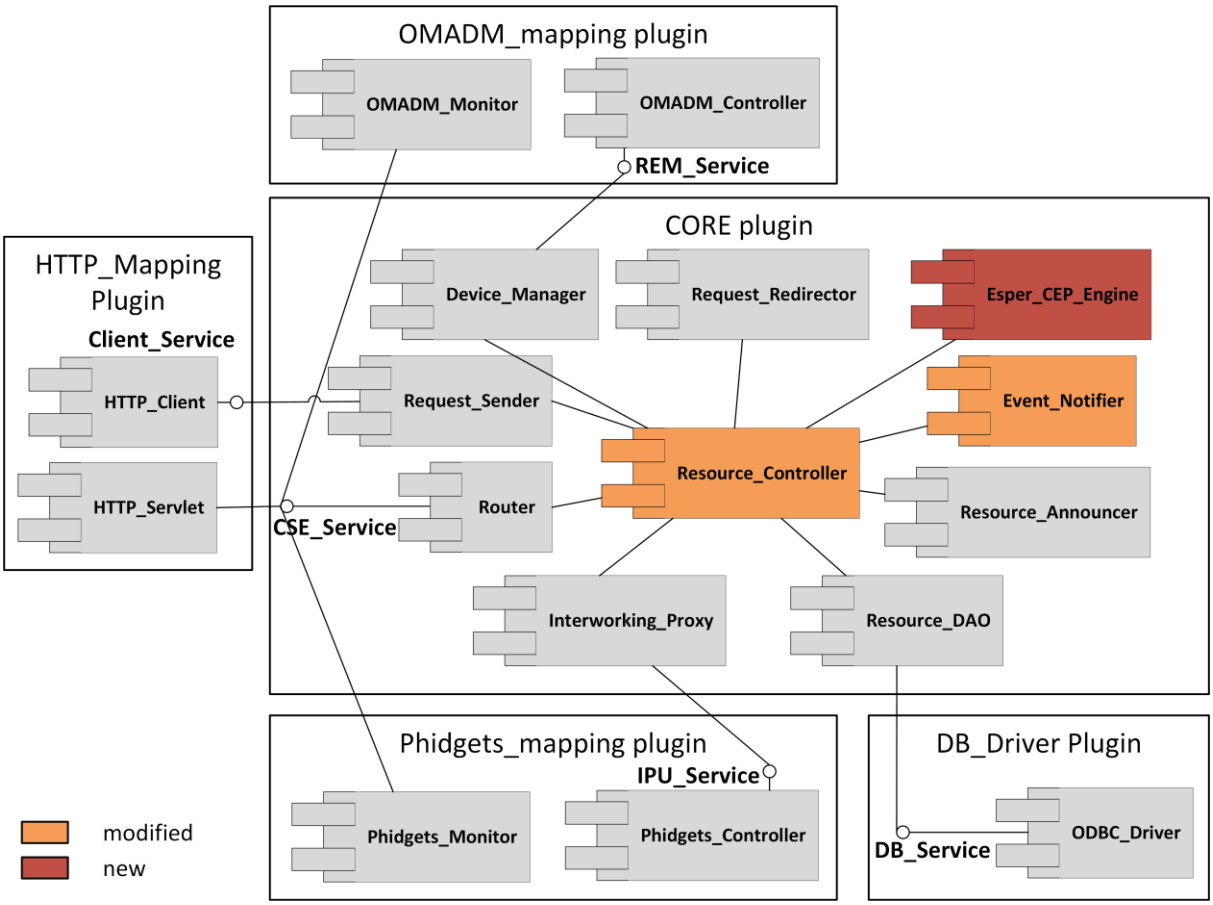

Figure 7.1: Component diagram of enhanced OM2M prototype showing modified and new CORE plugins

7.1.1 Eclipse OM2M Project

Although the release 1 of the oneM2M service platform on which the considerations of this research mostly rely on was released in January 2015 (see Chapter 4), the standard continues to evolve and still requires improvements (see Sections 4.1 and 6.3.3). Accordingly, there are currently only few implementations available which do not in any case implement the latest releases available.

One promising activity is the OM2M project (Alaya et al., 2014), hosted as open source project from the Eclipse foundation. This prototype implementation is based on the version 0.8.0 of OM2M which was the most recent version at the time of implementation but still implements the ETSI M2M standards (ETSI TS 102 690, 2013; ETSI TS 102 921, 2013). Table 7.1 shows a mapping of the oneM2M terms to the ETSI M2M standard (ETSI TS 102 690, 2013; ETSI TS 102 921, 2013) terms[21]. An additional difference are the collection resources that are still present in the ETSI M2M and OM2M prototype (see Section 4.4.3). In detail, in addition to the

[21] The description of the building blocks continues the wording according to the oneM2M standard in order to ease the understanding and traceability to previous chapters. For some details, in addition to Table 7.1, the mapping to the ETSI M2M standard is described to reflect the actual implementation in the enhanced OM2M prototype.

terms, some attributes and/or capabilities are evolved as well from the ETSI M2M to the oneM2M standard (oneM2M TS-0001, 2015; oneM2M TS-0004, 2015). However, since the ETSI standard is the main precursor of the oneM2M service platform, the relevant characteristics for data exchange such as data containers, contentInstance resources, subscriptions including the subscribe/notify mechanism, and eventNotificationCriteria (see Section 5.3.2) are comparable. This is also emphasised by the fact that an ETSI M2M platform can easily be upgraded to a oneM2M platform (Elloumi, 2014). In summary, it can be stated that this motivates the usage of OM2M as foundation for the prototypical implementation and ensures the validity of the results with respect to the proof-of-concept of the proposed enhancements for the oneM2M standards.

oneM2M	ETSI M2M
AE (see Section 4.3.1)	Device Application (DA), Gateway Application (GA), Network Application (NA)
CSE (see Section 4.3.1)	Device Service Capability Layer (DSCL), Gateway Service Capability Layer (GSCL), Network Service Capability Layer (NSCL)
Mca (see Section 4.3.2)	dIa (between DA and DSCL, or GA and GSCL), mIa (between NA and NSCL)
Mcc (see Section 4.3.2)	mId (between DSCL and NSCL, or GSCL and NSCL)
eventNotificationCriteria (see Section 5.3.2)	filterCriteria

Table 7.1: Mapping of selected oneM2M terms to ETSI M2M terms

The OM2M implementation of the oneM2M service platform is based on the OSGi Equinox framework which enables the creation of highly modular Java systems (Alliance, 2014; McAffer, VanderLei, & Archer, 2010). Hence, the CSE implementation itself follows a service-oriented approach which contributes to the extension of its capabilities such as those proposed here.

7.1.2 Esper CEP Engine

To implement CEP capabilities into the CSE layer of the prototype, the Esper CEP Engine has been selected. It is available, e.g., as a lightweight open-source Java library (EsperTech, 2016a), which eases its integration into the OM2M prototype. The Esper CEP Engine has proven its suitability in several contexts, including (automotive) embedded systems and M2M (Bruns et al., 2015; Bruns & Dunkel, 2015; Terroso-Sáenz, Valdés-Vela, Campuzano, Botia, & Skarmeta-Gómez, 2015).

Esper supports various types of event representations including XML respectively XSD. Moreover, it supports a tailored EPL to express the description of event conditions (Bruns & Dunkel, 2015).

7.2 Building Blocks

The OM2M prototype is enhanced to implement the proposed enhanced data exchange capabilities for the subscribe/notify mechanism of the oneM2M service architecture (see Section 6).

This section describes the building blocks which implement the related functionalities for the enhanced data exchange. These building blocks are integrated into the existing CSE handlers of the methods (see Section 4.4.4), provided by the respective reference points (see Section 4.3.2). In this regard, the reference points as well become enhanced (i.e., Mca-E[22] and Mcc-E, see Section 6.3).

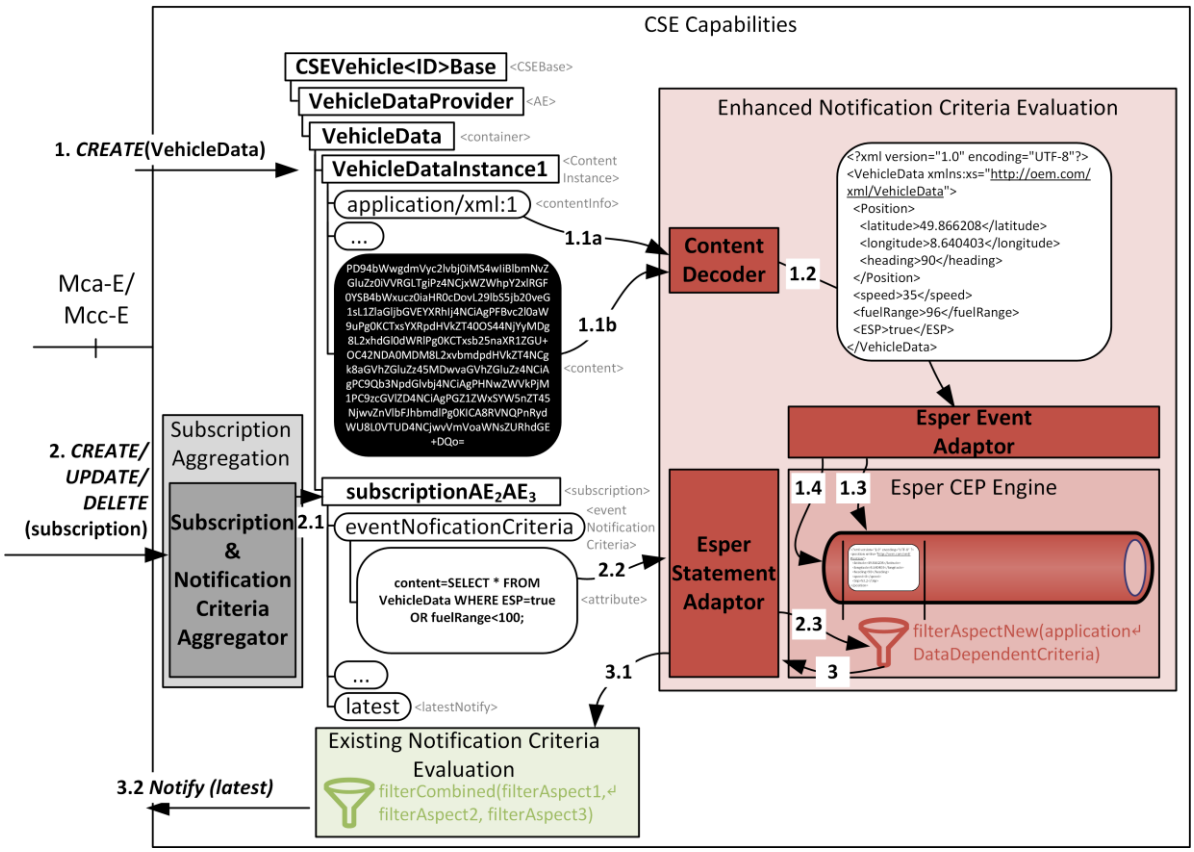

Figure 7.2: Integration of enhanced data exchange capabilities for subscribe/notify within CSE layer of oneM2M service platform

Figure 7.2 shows at a glance the integration and interplay of the new building blocks (colour-coded red) to implement enhanced data exchange capabilities for the oneM2M CSE. In regard

[22] In the enhanced OM2M prototype, this refers to dIa-E or mIa-E.

to enhanced notification criteria evaluation, three building blocks have been added: Content Decoder, Esper Event Adaptor, and Esper Statement Adaptor. Furthermore, it is indicated by the grey building block Subscription & Notification Criteria Aggregator, where such aggregation of subscriptions (see Section 6.2.3) could be added. However, since OM2M in version 0.8.0 has too few retargeting capabilities across CSEs, this is not part of this enhanced OM2M prototype. Nevertheless, to provide further implementation considerations, it is illustrated in Figure 7.2 and detailed afterwards. For the experiments, it is expected that already aggregated eventNotificiationCriteria are provided.

The Content Decoder and Esper Event Adaptor have been integrated into the CREATE method for contentInstance resources (Figure 7.2, steps 1.x). The Esper Statement Adaptor has been integrated into the CREATE, UPDATE, and DELETE methods of the subscription resource (Figure 7.2, steps 2.2-2.3). Further, parts of the Esper Statement Adaptor are integrated into the Notify method (Figure 7.2, steps 3.x).

In the following, these building blocks and the Subscription & Notification Criteria Aggregator are introduced in more detail, including the description of the steps in which they are involved by means of an example.

7.2.1 Content Decoder

The oneM2M CSE currently handles application data opaque at the CSE (see Section 5.3.1). This on the one hand preserves maximum variability for the M2M applications with respect to the structuring of application data and formats. Nevertheless, on the other hand, towards interoperability in the context of unbounded systems (see Sections 3.4.2 and 4.2.1), generic and machine-readable data specifications should be provided. The prototypical implementation follows the proposed approach of utilising XML and XSD to gain transparent data syntax of the content (see Section 6.2.1). For this purpose, the existing oneM2M attribute contentInfo to decode a base64-encoded content attribute is used.

The starting point is the creation of a new contentInstance within a container. Figure 7.2 illustrates an example where the contentInstance VehicleDataInstance1 within the container VehicleData is created (Figure 7.2, step 1). The CREATE method handler checks whether the contentInfo is set to ‘application/xml:1’ (which indicates a base64 encoded string of an XML document). If so, this encoding information together with the related content is forwarded to the Content Decoder (Figure 7.2, step 1.1a, step 1.1b). Then the decoded application data (i.e., an XML document) is passed on to the Esper Event Adaptor (Figure 7.2, step 2).

7.2.2 Esper Event Adaptor

The Esper Event Adaptor implements two functionalities: First, it enables the registration of an event type for the respective application data format. Second, it enables the propagation of

incoming application data (i.e., content) as event. For both, the decoded XML document from the Content Decoder is the input.

First, the Esper Event Adaptor verifies whether the XML document includes a link to at least one XML Schema Definition (XSD), formally describing the XML document with respect to its structure, elements, and their simple or complex data types, etc. (Salminen & Tompa, 2011). If this is the case, the Esper Event Adaptor determines whether the XSD document is already known. If not, the XSD document is obtained (i.e., downloaded) and passed on to the Esper CEP Engine to register an associated event type (Figure 7.2, step 1.3). In the example of Figure 7.2, the XSD is located at http://oem.com/xml/VehicleData. Thus, the Esper CEP Engine is aware of the application data format by means of syntax and data types which is the foundation for the specification, verification, and processing of related event queries. Given that the XSD does not change, this step 1.3 is only executed once at the first creation of a new contentInstance within the container.

Second, the XML document is passed on to the Esper CEP Engine as new event (Figure 7.2, step 1.4) of the respective event type. This step is repeated for every new contentInstance creation, given that a subscription to the respective container exists.

7.2.3 Subscription and Notification Criteria Aggregator

In order to implement the aggregation of different subscriptions to the same remote resource before forwarding the subscription to transit or remote CSEs[23], the subscription CREATE method handling at the origin oneM2M CSE has to be enhanced (Figure 7.2, step 2.1). It utilises the fact that the oneM2M CSE interposes such subscription creation requests in any case to enable the subscription routing across one or many CSEs and notification retargeting, if applicable. This implementation would follow the selected approach and architectural alternative (see Sections 6.2 and 6.3.2). To implement the related enhancement, the related methodology has to be enhanced to additionally save all relevant notification constraints such as eventNotificationCriteria including the new application-data-dependent notification criteria (i.e., the EPL statement), schedule, etc. (cf. Section 5.3.2). If an additional subscription to the same remote resource runs through the CSE (this is a given for the CSE as origin or transit CSE), such subscription could be detected and the aggregation workflow according to Section 6.2.3 could be performed. However, as stated above, this is not part of the enhanced OM2M prototype.

[23] Since the aggregation of different subscriptions to the same resource is only reasonable for remote resources, the subscription to local resources has not been considered here.

7.2.4 Esper Statement Adaptor

The Esper Statement Adaptor implements two functionalities: The registration of EPL statements at the respective event types of the Esper CEP and the handling of related notifications in cases where the statement is fulfilled.

To implement the application-data-dependent notification policies with minimal changes to the oneM2M service platform and its reference points, and to continue preliminary design decisions of oneM2M, the existing eventNotificationCriteria attribute capabilities are utilised (see Section 5.3.2). The list of condition tags of the eventNotificationCriteria attribute already facilitates policies, addressing single attributes of the subscribed-to resource. This, in addition to others, enables full-text notification constraints which can be used, e.g., to filter on specific creator, contentInfo or labels (see Section 5.3.2). Those existing capabilities are enhanced to support criteria addressing the content attribute of the subscribed-to resource which are expressed through an EPL statement. The handler of the subscription CREATE method is enhanced to check whether the eventNotificationCriteria list includes an item following the syntax "content=[EPLstatement]"[24] such as "content=SELECT * FROM VehicleData WHERE ESP=true OR fuelRange<100;" (see example of Figure 7.2). If this is the case, the subscription is forwarded to the Esper Statement Adaptor (Figure 7.2, step 2.2).

The Esper Statement Adaptor extracts such EPL statement of a subscription and adds it to the list of events being evaluated by the Esper CEP Engine (Figure 7.2, step 2.3)[25]. Consequently, the application-data-dependent notification criteria, represented and referred to as the filterAspectNew(applicationDataDependentCriteria), is evaluated within the Esper CEP Engine according to the envisaged approach (see Section 6.2).

During the registration of an EPL statement at the Esper CEP Engine, the Esper Statement Adaptor additionally provides a callback for the related notification. This callback is integrated into the existing notification preparation functionality of the oneM2M CSE. When the EPL statement, e.g., filterAspectNew(applicationDataDependentCriteria), is fulfilled through the incoming events (see Section 7.2.2), this callback is called (Figure 7.2, step 3) and the Esper Statement Adaptor forwards it to the existing notification criteria evaluation (Figure 7.2, step 3.1), e.g., filterAspectNew(applicationDataDependentCriteria). This realises the envisaged concatenation of the filterAspectNew(applicationDataDependentCriteria) with the existing filter capabilities, namely filterCombined(filterAspect1, filterAspect2, filterAspect3), following the approach (see Section 6.2).

[24] In this enhanced OM2M prototype, this is realized through a check whether the filterCriteria ifMatch starts with "select".

[25] This proof-of-concept prototype requires the creation of at least one contentInstance containing the reference to the related XSD before a subscription can be created to ensure that the related event type is already registered at the Esper CEP Engine.

If a subscription is being updated or deleted, the Esper Statement Adaptor is responsible for the adoption or deletion of the related EPL statements and callbacks at the Esper CEP engine.

7.3 Experiments and Estimations

Section 7.2 detailed the implementation of the CSE prototype and therefore used a white-box view of the enhanced CSE. To reflect the actual usage of the CSE layer, the experiments are performed using the CSE as black-box. This means, the CSE is used through the respective enhanced reference points. This section describes one possible experimentation setup, which facilitates the proof-of-concept of the proposed enhancements and the assessment of the related results.

7.3.1 Setup and Test Strategy

This section describes the setup of the experiments to enable the assessment of the prototype and estimations about the benefits of the proposed enhancements. It starts with the explanation of the general setup, followed by the description of the bootstrap. Finally, the approach to execute the experiments and to gather the results for further assessment is presented.

General

Figure 7.3 illustrates the entities and their relationship for the setup of the experiment. Here, the data exchange is broken down to its minimum setup by means of one AE connected to a single CSE. This facilitates the assessment of the enhanced data exchange capabilities by means of enhanced subscribe/notify of a single CSE which provides a good basis for further considerations and estimations.

In order to build a generic AEPrototype that is able to mimic the behaviour of various AEs, the "Simple REST Client" is used. It enables a high degree of freedom with regard to the http requests sent to the CSE, while at the same time ensuring full http compliance. The AEPrototype is used as a kind of test harness that sets up and executes the experiments (i.e., test cases). Figure 7.4 shows the usage of "Simple REST Client" as AEPrototype showing the creation/registration of a VehicleDataProvider application at the CSE/NSCL.

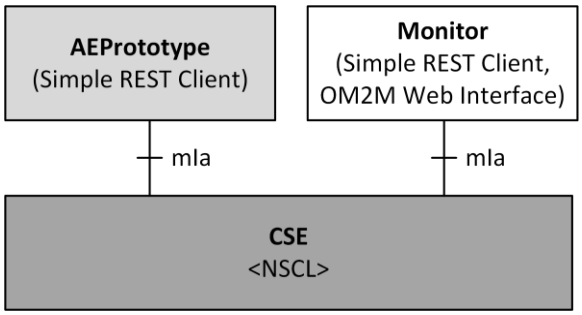

Figure 7.3: Experiment setup

The Monitor is used to supervise the experiments and to collect the results. This means, that the current state of the CSE (that is manifested in the resource tree of the CSE) can be investigated through the enhanced mIa-E interface with the appropriate degree of detail (e.g., various retrieve operations at different resource (sub-)trees) at any time during the experiment execution. Accordingly, again either the “Simple REST Client” or the built-in OM2M Web Interface can be used to realize this Monitor functionality. Figure 7.5 shows the usage of OM2M Web Interface as Monitor showing the evaluation of the resource tree after the creation/registration of a VehicleDataProvider application at the CSE/NSCL. Similar to the AEPrototype, the Monitor is a passive entity in this experiment setup, which means that it does not provide its own server capabilities. However, considering the design of the oneM2M service platform together with the setup of the experiments, this does not limit the validity of the results and their transferability to the condensed ASDP scenario (see Section 5.1).

Simple REST Client

Simple REST Client | chrome-extension://fhjcajmcbmldlhcimfajhfbgofnpcjmb/index.html

Simple REST Client

Request

URL: http://127.0.0.1:8080/om2m/nscl/applications

Method: GET POST PUT DELETE HEAD OPTIONS

Headers: Authorization: Basic YWRtaW46YWRtaW4=

Data:

```
<om2m:application xmlns:om2m="http://uri.etsi.org/m2m" appId="VehicleDataProvider">
</om2m:application>
```

Clear Send

Response

Status: 201 Created

Headers:

```
location: nscl/applications/VehicleDataProvider
server: Jetty(8.1.16.v20140903)
content-length: 1410
content-type: application/xml;charset=ISO-8859-1
```

Data:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<om2m:application xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" appId="Ve
    <om2m:accessRightID>nscl/accessRights/AR_ADMIN</om2m:accessRightID>
    <om2m:searchStrings>
        <om2m:searchString>ResourceType/Application</om2m:searchString>
        <om2m:searchString>ResourceID/VehicleDataProvider</om2m:searchString>
    </om2m:searchStrings>
    <om2m:expirationTime>2049-05-17T22:54:58.212+02:00</om2m:expirationTime>
    <om2m:creationTime>2017-09-08T21:08:19.214+02:00</om2m:creationTime>
    <om2m:lastModifiedTime>2017-09-08T21:08:19.214+02:00</om2m:lastModifiedTime>
    <om2m:announceTo>
        <om2m:activated>false</om2m:activated>
        <om2m:global>false</om2m:global>
    </om2m:announceTo>
    <om2m:containersReference>nscl/applications/VehicleDataProvider/containers</om2m:containersReference>
    <om2m:groupsReference>nscl/applications/VehicleDataProvider/groups</om2m:groupsReference>
    <om2m:accessRightsReference>nscl/applications/VehicleDataProvider/accessRights</om2m:accessRightsReference>
    <om2m:subscriptionsReference>nscl/applications/VehicleDataProvider/subscriptions</om2m:subscriptionsReferenc
    <om2m:notificationChannelsReference>nscl/applications/VehicleDataProvider/notificationChannels</om2m:notific
</om2m:application>
```

Copyright © 2010 Jeremy Selier - Source code licensed under the Apache License - icon by Jason Rayner

Figure 7.4: Usage of Simple REST Client as AEPrototype showing the creation/registration of a VehicleDataProvider application at the CSE/NSCL

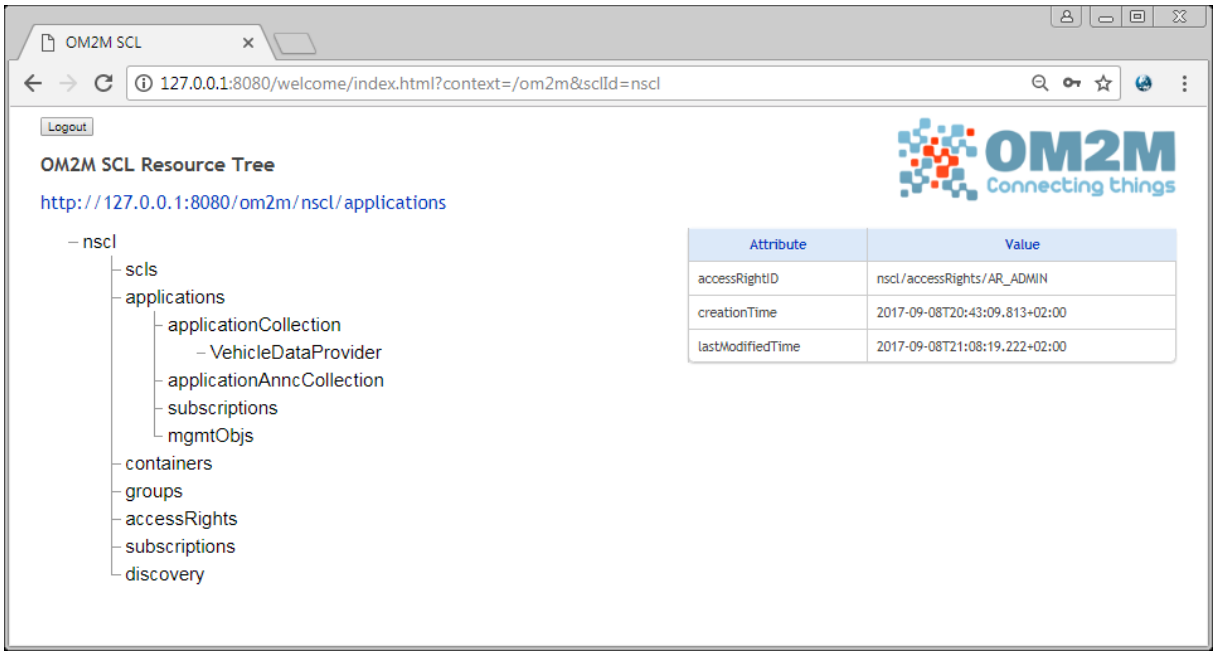

Figure 7.5: Usage of OM2M Web Interface as Monitor showing the evaluation of the resource tree after the creation/registration of a VehicleDataProvider application at the CSE/NSCL

The CSE is realized by the enhanced OM2M prototype as explained in Section 7.2. The NSCL (representing the example CSE) is started on the localhost on port 8080. Accordingly, it can be accessed at http://127.0.0.1:8080/om2m/nscl/. The OM2M NSCL by default is protected by basic access control which was also kept for this enhanced prototype. Default username/password are admin/admin. When accessing the OM2M Web Interface, these can be entered in the logon form. Every request through the mIa-E reference point has to provide the login data base64 encoded separated by a colon: base64(admin:admin) = YWRtaW46YWRtaW4=. Following this, the complete http header should be: Authorization: Basic YWRtaW46YWRtaW4=.

Finally, the XSD file specifying the XML application data of experiment has to be accessible at the specified location. In this setup, it is provided by an apache webserver running on localhost on port 80. Thus, the XSD reference within the XML application data (e.g., VehicleData) specifies http://127.0.0.1:80/xsd/VehicleData.xsd.

Test Strategy

The basic idea of the three experiments performed and described in the following are to compare the number of NOTIY messages resulting from the standard oneM2M data exchange to the number of NOTIFY messages resulting from the enhanced data exchange capabilities. Various test strategies are possible: For example, the test can be completely performed and afterwards the number of NOTIFY messages (respectively contentInstance resources in the related containers) can be counted. Besides, boundary tests (cf. Kossiakoff et al., 2011) derived from the standard and the enhanced data subscriptions can be used, as it is done with the

following experiments. The boundary tests facilitate a proper test of the CSE and ensure that the results can be used as basis for further estimations.

Bootstrap

This section details which steps are performed to bootstrap the experiment. Figure 7.6 shows the related sequence diagram of the interactions between the AEPrototype, the CSE, and the Monitor. The steps are afterwards explained in detail including the requests performed and responses received.

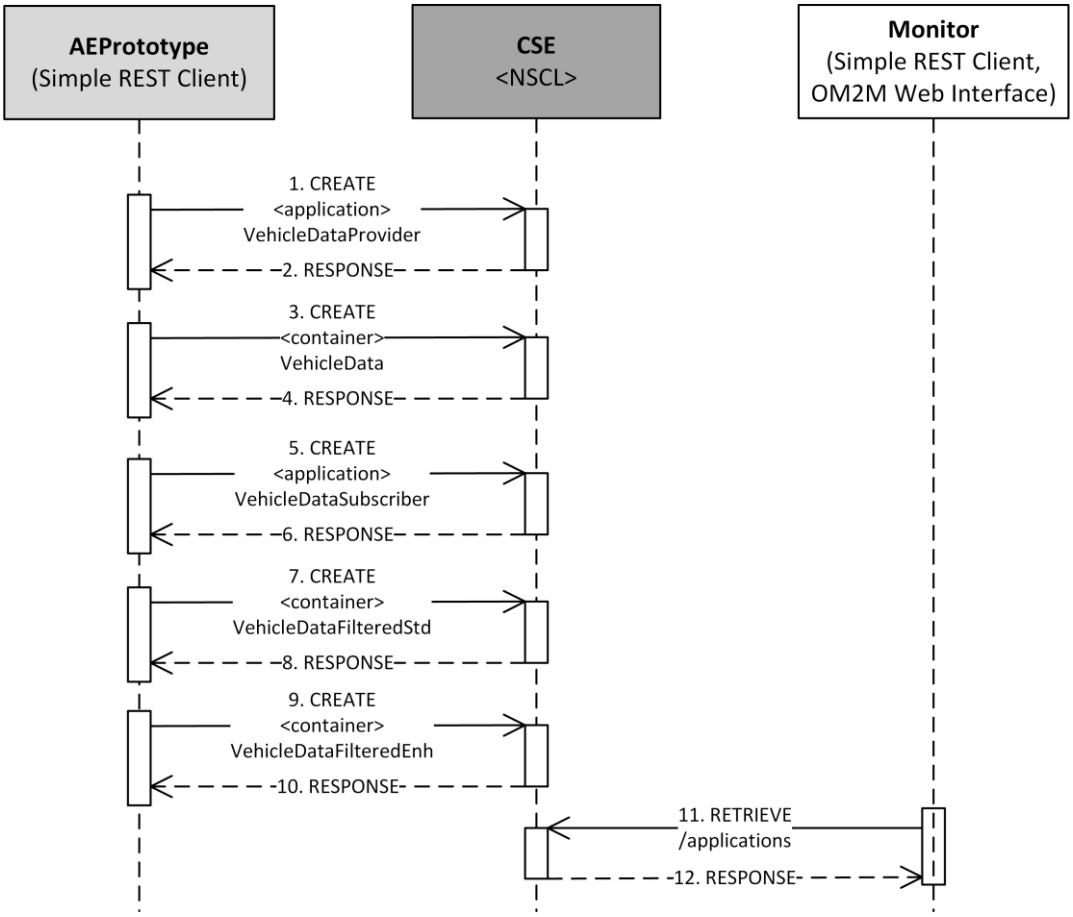

Figure 7.6: Sequence Diagram of Experiment Bootstrap

Step 1: The CREATE of an application resource VehicleDataProvider at nscl/applications is requested by use of the AEPrototype.

URL:	http://127.0.0.1:8080/om2m/nscl/applications
Method:	POST
Headers:	Authorization: Basic YWRtaW46YWRtaW4=
Data:	<om2m:application xmlns:om2m="http://uri.etsi.org/m2m" appId="VehicleDataProvider"> </om2m:application>

Step 2: The NSCL sends a RESPONSE with status 201 Created to the AEPrototype.

Status:	201 Created
Headers:	location: nscl/applications/VehicleDataProvider server: Jetty(8.1.16.v20140903) content-length: 1410

	content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:application xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" appId="VehicleDataProvider"> <om2m:accessRightID>nscl/accessRights/AR_ADMIN</om2m:accessRightID> <om2m:searchStrings> <om2m:searchString>ResourceType/Application</om2m:searchString> <om2m:searchString>ResourceID/VehicleDataProvider</om2m:searchString> </om2m:searchStrings> <om2m:expirationTime>2049-05- 18T15:05:47.823+02:00</om2m:expirationTime> <om2m:creationTime>2017-09-09T13:19:08.825+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09- 09T13:19:08.825+02:00</om2m:lastModifiedTime> <om2m:announceTo> <om2m:activated>false</om2m:activated> <om2m:global>false</om2m:global> </om2m:announceTo> <om2m:containersReference>nscl/applications/VehicleDataProvider/containers</om2m:containersReference> <om2m:groupsReference>nscl/applications/VehicleDataProvider/groups</om2m:groupsReference> <om2m:accessRightsReference>nscl/applications/VehicleDataProvider/accessRights</om2m:accessRightsReference> <om2m:subscriptionsReference>nscl/applications/VehicleDataProvider/subscriptions</om2m:subscriptionsReference> <om2m:notificationChannelsReference>nscl/applications/VehicleDataProvider/notificationChannels</om2m:notificationChannelsReference> </om2m:application>

Step 3: The CREATE of a container resource VehicleData at nscl/applications/VehicleDataProvider/containers is requested by use of the AEPrototype.

URL:	http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataProvider/containers
Method:	POST
Headers:	Authorization: Basic YWRtaW46YWRtaW4=
Data:	<om2m:container xmlns:om2m="http://uri.etsi.org/m2m" om2m:id="VehicleData"> </om2m:container>

Step 4: The NSCL sends a RESPONSE with status 201 Created to the AEPrototype.

Status:	201 Created
Headers:	location: nscl/applications/VehicleDataProvider/containers/CONT_923464169 server: Jetty(8.1.16.v20140903) content-length: 1179 content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:container xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" om2m:id="CONT_923464169"> <om2m:accessRightID>nscl/accessRights/AR_ADMIN</om2m:accessRightID> <om2m:searchStrings> <om2m:searchString>ResourceType/Container</om2m:searchString> <om2m:searchString>ResourceID/CONT_923464169</om2m:searchString>

	</om2m:searchStrings> <om2m:expirationTime>2049-05-18T15:07:44.908+02:00</om2m:expirationTime> <om2m:creationTime>2017-09-09T13:21:05.908+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T13:21:05.908+02:00</om2m:lastModifiedTime> <om2m:announceTo> <om2m:activated>false</om2m:activated> <om2m:global>false</om2m:global> </om2m:announceTo> <om2m:maxNrOfInstances>100</om2m:maxNrOfInstances> <om2m:contentInstancesReference>nscl/applications/VehicleDataProvider/containers/CONT_923464169/contentInstances</om2m:contentInstancesReference> <om2m:subscriptionsReference>nscl/applications/VehicleDataProvider/containers/CONT_923464169/subscriptions</om2m:subscriptionsReference> </om2m:container>

Step 5: The CREATE of an application resource VehicleDataSubscriber at nscl/applications is requested by use of the AEPrototype.

URL:	http://127.0.0.1:8080/om2m/nscl/applications
Method:	POST
Headers:	Authorization: Basic YWRtaW46YWRtaW4=
Data:	<om2m:application xmlns:om2m="http://uri.etsi.org/m2m" appId="VehicleDataSubscriber"> </om2m:application>

Step 6: The NSCL sends a RESPONSE with status 201 Created to the AEPrototype.

Status:	201 Created
Headers:	location: nscl/applications/VehicleDataSubscriber server: Jetty(8.1.16.v20140903) content-length: 1424 content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:application xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" appId="VehicleDataSubscriber"> <om2m:accessRightID>nscl/accessRights/AR_ADMIN</om2m:accessRightID> <om2m:searchStrings> <om2m:searchString>ResourceType/Application</om2m:searchString> <om2m:searchString>ResourceID/VehicleDataSubscriber</om2m:searchString> </om2m:searchStrings> <om2m:expirationTime>2049-05-18T15:09:44.953+02:00</om2m:expirationTime> <om2m:creationTime>2017-09-09T13:23:05.953+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T13:23:05.953+02:00</om2m:lastModifiedTime> <om2m:announceTo> <om2m:activated>false</om2m:activated> <om2m:global>false</om2m:global> </om2m:announceTo> <om2m:containersReference>nscl/applications/VehicleDataSubscriber/containers</om2m:containersReference> <om2m:groupsReference>nscl/applications/VehicleDataSubscriber/groups</om2m:groupsReference>

	<om2m:accessRightsReference>nscl/applications/VehicleDataSubscriber/accessRights</om2m:accessRightsReference> <om2m:subscriptionsReference>nscl/applications/VehicleDataSubscriber/subscriptions</om2m:subscriptionsReference> <om2m:notificationChannelsReference>nscl/applications/VehicleDataSubscriber/notificationChannels</om2m:notificationChannelsReference> </om2m:application>

Step 7: The CREATE of a container resource VehicleDataFilteredStd at nscl/applications/VehicleDataSubscriber/containers is requested by use of the AEPrototype.

URL:	http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataSubscriber/containers
Method:	POST
Headers:	Authorization: Basic YWRtaW46YWRtaW4=
Data:	<om2m:container xmlns:om2m="http://uri.etsi.org/m2m" om2m:id="VehicleDataFilteredStd"> </om2m:container>

Step 8: The NSCL sends a RESPONSE with status 201 Created to the AEPrototype.

Status:	201 Created
Headers:	location: nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredStd server: Jetty(8.1.16.v20140903) content-length: 1215 content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:container xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" om2m:id="VehicleDataFilteredStd"> <om2m:accessRightID>nscl/accessRights/AR_ADMIN</om2m:accessRightID> <om2m:searchStrings> <om2m:searchString>ResourceType/Container</om2m:searchString> <om2m:searchString>ResourceID/VehicleDataFilteredStd</om2m:searchString> </om2m:searchStrings> <om2m:expirationTime>2049-05-18T15:12:38.110+02:00</om2m:expirationTime> <om2m:creationTime>2017-09-09T13:25:59.110+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T13:25:59.110+02:00</om2m:lastModifiedTime> <om2m:announceTo> <om2m:activated>false</om2m:activated> <om2m:global>false</om2m:global> </om2m:announceTo> <om2m:maxNrOfInstances>100</om2m:maxNrOfInstances> <om2m:contentInstancesReference>nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredStd/contentInstances</om2m:contentInstancesReference> <om2m:subscriptionsReference>nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredStd/subscriptions</om2m:subscriptionsReference> </om2m:container>

Step 9: The CREATE of a container resource VehicleDataFilteredEnh at nscl/applications/VehicleDataSubscriber/containers is requested by use of the AEPrototype.

URL:	http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataSubscriber/containers
Method	POST
Headers:	Authorization: Basic YWRtaW46YWRtaW4=
Data:	<om2m:container xmlns:om2m="http://uri.etsi.org/m2m" om2m:id="VehicleDataFilteredEnh"> </om2m:container>

Step 10: The NSCL sends a RESPONSE with status 201 Created to the AEPrototype.

Status:	201 Created
Headers:	location: nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredEnh server: Jetty(8.1.16.v20140903) content-length: 1215 content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:container xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" om2m:id="VehicleDataFilteredEnh"> <om2m:accessRightID>nscl/accessRights/AR_ADMIN</om2m:accessRightID> <om2m:searchStrings> <om2m:searchString>ResourceType/Container</om2m:searchString> <om2m:searchString>ResourceID/VehicleDataFilteredEnh</om2m:searchString> </om2m:searchStrings> <om2m:expirationTime>2049-05- 18T15:14:12.561+02:00</om2m:expirationTime> <om2m:creationTime>2017-09-09T13:27:33.562+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09- 09T13:27:33.562+02:00</om2m:lastModifiedTime> <om2m:announceTo> <om2m:activated>false</om2m:activated> <om2m:global>false</om2m:global> </om2m:announceTo> <om2m:maxNrOfInstances>100</om2m:maxNrOfInstances> <om2m:contentInstancesReference>nscl/applications/VehicleDataSubscriber/contai ners/VehicleDataFilteredEnh/contentInstances</om2m:contentInstancesReference> <om2m:subscriptionsReference>nscl/applications/VehicleDataSubscriber/container s/VehicleDataFilteredEnh/subscriptions</om2m:subscriptionsReference> </om2m:container>

Step 11: The successful bootstrap of the experiment can be assessed by use of the Monitor through the RETRIEVE of certain (sub-)trees of the resource tree. In the given example, a RETRIEVE of the nscl/applications is performed.

URL:	http://127.0.0.1:8080/om2m/nscl/applications
Method:	GET
Headers:	Authorization: Basic YWRtaW46YWRtaW4=

Step 12: The NSCL sends a RESPONSE with status 200 OK to the AEPrototype. It shows the successful creation of the applications VehicleDataProvider and VehicleDataSubscriber.

Status:	200 OK
Headers:	content-length: 922 server: Jetty(8.1.16.v20140903) content-type: application/xml;charset=ISO-8859-1

Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:applications xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime"> <om2m:accessRightID>nscl/accessRights/AR_ADMIN</om2m:accessRightID> <om2m:creationTime>2017-09-09T13:17:39.513+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T13:23:05.956+02:00</om2m:lastModifiedTime> <om2m:applicationCollection> <om2m:namedReference id="VehicleDataProvider">nscl/applications/VehicleDataProvider</om2m:namedReference> <om2m:namedReference id="VehicleDataSubscriber">nscl/applications/VehicleDataSubscriber</om2m:namedReference> </om2m:applicationCollection> <om2m:applicationAnncCollection/> <om2m:subscriptionsReference>nscl/applications/subscriptions</om2m:subscriptionsReference> <om2m:mgmtObjsReference>nscl/applications/mgmtObjs</om2m:mgmtObjsReference> </om2m:applications>

This kind of bootstrap and the AEs and containers created are rather universal and enable the execution of diverse experiments. However, if required, the bootstrap could also be modified in order to evaluate different experiments and estimations.

Experiment Execution and Result Retrieve

This section details which steps are performed to execute the experiment and to retrieve the results. Figure 7.7 shows the related sequence diagram of the interactions between the AEPrototype, the CSE, and the Monitor. The steps are afterwards explained in detail including the requests performed and responses received.

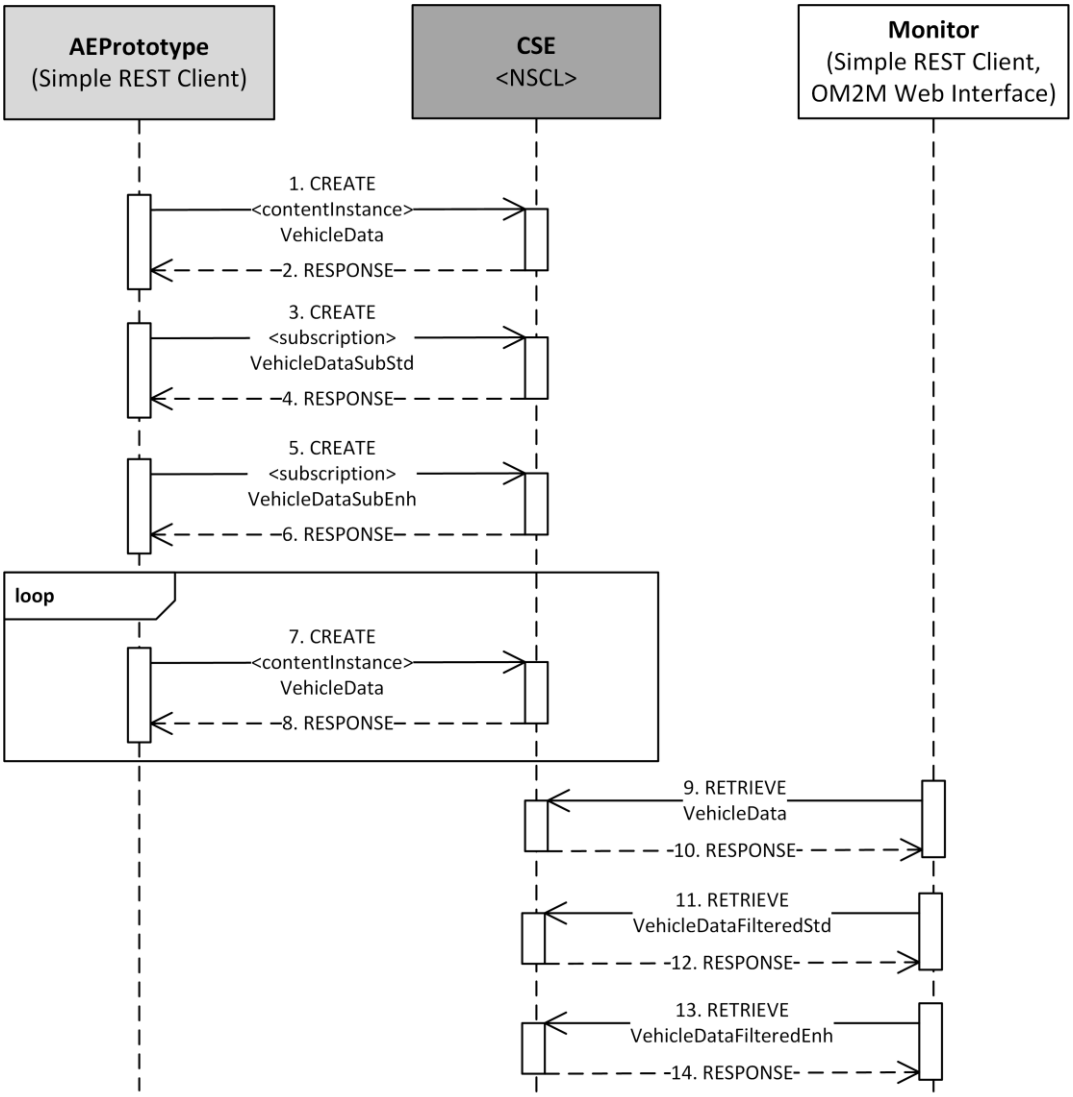

Figure 7.7: Sequence Diagram of Experiment Execution and Result Retrieve

Step 1: The CREATE of a contentInstance resource (type: VehicleData XML) at nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstances is requested by use of the AEPrototype. This is done to trigger the registration of a new event type related to the referenced XSD, see Section 7.2.2.

URL:	http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataProvider/↩ containers/VehicleData/contentInstances
Method:	POST
Headers:	Authorization: Basic YWRtaW46YWRtaW4=
Data:	<?xml version="1.0" encoding="UTF-8"?> <VehicleData xmlns:xs="http://127.0.0.1:80/xsd/VehicleData.xsd"> <Position> <latitude>49.866208</latitude> <longitude>8.640403</longitude> <heading>90</heading> </Position> <speed>35</speed> <fuelRange>750</fuelRange> <ESP>false</ESP> </VehicleData>

Step 2: The NSCL sends a RESPONSE with status 201 Created to the AEPrototype.

Status:	201 Created
Headers:	location: nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstances/CI_787760297 server: Jetty(8.1.16.v20140903) content-length: 1073 content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:contentInstance xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" om2m:id="CI_787760297" href="nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstances/CI_787760297"> <om2m:creationTime>2017-09-09T13:45:42.392+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T13:45:42.392+02:00</om2m:lastModifiedTime> <om2m:delayTolerance>2017-09-09T17:05:42.392+02:00</om2m:delayTolerance> <om2m:contentSize>310</om2m:contentSize> <om2m:content xmime:contentType="application/xml">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPFZlaGljbGVEYXRhIHhtbG5zOnhzPSJodHRwOi8vMTI3LjAuMC4xOjgwL3hzZC9WZWhpY2xlRGF0YS54c2QiPgogIDxQb3NpdGlvbj4KICAgIDxsYXRpdHVkZT40OS44NjYyMDg8L2xhdGl0dWRlPgogICAgPGxvbmdpdHVkZT44LjY0MDQwMzwvbG9uZ2l0dWRlPgogICAgPGhlYWRpbmc+OTA8L2hlYWRpbmc+CiAgPC9Qb3NpdGlvbj4KICA8c3BlZWQ+MzU8L3NwZWVkPgogIDxmdWVsUmFuZ2U+NzUwPC9mdWVsUmFuZ2U+CiAgPEVTUD5mYWxzZTwvRVNQPgo8L1ZlaGljbGVEYXRhPg==</om2m:content> </om2m:contentInstance>

Step 3: The CREATE of a subscription resource VehicleDataSubStd[26] at nscl/applications/VehicleDataProvider/containers/VehicleData/↩ contentInstances/subscriptions is requested by use of the AEPrototype. This subscription uses just the standard subscription capabilities without the enhancements. In this example, it subscribes to every VehicleData contentInstance creation. To ease the assessment of the result, the target for the NOTIFY requests is the container resource nscl/applications/VehicleDataSubscriber/↩ containers/VehicleDataFilteredStd.

URL:	http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataProvider/↩ containers/VehicleData/contentInstances/subscriptions
Method:	POST
Headers:	Authorization: Basic YWRtaW46YWRtaW4=
Data:	<om2m:subscription xmlns:om2m="http://uri.etsi.org/m2m"> <om2m:contact>http://127.0.0.1:8080/om2m/nscl/applications/↩ VehicleDataSubscriber/containers/VehicleDataFilteredStd/contentInstances↩ </om2m:contact> </om2m:subscription>

Step 4: The NSCL sends a RESPONSE with status 201 Created to the AEPrototype.

Status:	201 Created

[26] The subscription name VehicleDataSubStd serves identification and reference purposes within the experiments. Actually, the subscription resource names are generated automatically by the NSCL and cannot be set externally.

Headers:	location: nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstances/subscriptions/SUB_556424748 server: Jetty(8.1.16.v20140903) content-length: 654 content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:subscription xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" om2m:id="SUB_556424748"> <om2m:expirationTime>2049-05-18T15:39:47.184+02:00</om2m:expirationTime> <om2m:creationTime>2017-09-09T13:53:08.184+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T13:53:08.184+02:00</om2m:lastModifiedTime> <om2m:filterCriteria/> <om2m:subscriptionType>ASYNCHRONOUS</om2m:subscriptionType> <om2m:contact>http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredStd/contentInstances</om2m:contact> </om2m:subscription>

Step 5: The CREATE of a subscription resource VehicleDataSubEnh[27] at nscl/applications/VehicleDataProvider/containers/VehicleData/↩ contentInstances/subscriptions is requested by use of the AEPrototype. This subscription uses the enhanced subscription capabilities as proposed and developed with this research. In this example, it subscribes to every VehicleData contentInstance creation where the remaining fuelRange is lower than 100. To ease the assessment of the result, the target for the NOTIFY requests is the container resource nscl/applications/VehicleDataSubscriber/↩ containers/VehicleDataFilteredEnh.

URL:	http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataProvider/↩ containers/VehicleData/contentInstances/subscriptions
Method:	POST
Headers:	Authorization: Basic YWRtaW46YWRtaW4=
Data:	<om2m:subscription xmlns:om2m="http://uri.etsi.org/m2m"> <om2m:filterCriteria> <ifMatch>select * from VehicleData where fuelRange < 100</ifMatch> </om2m:filterCriteria> <om2m:contact>http://127.0.0.1:8080/om2m/nscl/applications/↩ VehicleDataSubscriber/containers/VehicleDataFilteredEnh/contentInstances↩ </om2m:contact> </om2m:subscription>

Step 6: The NSCL sends a RESPONSE with status 201 Created to the AEPrototype.

Status:	201 Created
Headers:	location: nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstances/subscriptions/SUB_960200273 server: Jetty(8.1.16.v20140903) content-length: 785 content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

[27] The subscription name VehicleDataSubEnh serves identification and reference purposes within the experiments. Actually, the subscription resource names are generated automatically by the NSCL and cannot be set externally.

	<om2m:subscription xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" om2m:id="SUB_712784690"> <om2m:expirationTime>2049-05-19T09:27:10.398+02:00</om2m:expirationTime> <om2m:creationTime>2017-09-10T07:40:31.398+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-10T07:40:31.398+02:00</om2m:lastModifiedTime> <om2m:filterCriteria> <ifMatch>select * from VehicleData where fuelRange < 100</ifMatch> </om2m:filterCriteria> <om2m:subscriptionType>ASYNCHRONOUS</om2m:subscriptionType> <om2m:contact>http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredEnh/contentInstances</om2m:contact> </om2m:subscription>

Step 7: The CREATE of a contentInstance resource (type: VehicleData XML) at nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstances is requested by use of the AEPrototype. This step 7 (incl. step 8) can be repeated as often as necessary, in order to execute the experiment and generate the related results. In this example, it is repeated three times, with changed fuelRange of 749, 100, 99 in order to check the boundary values of the experiment (see Section 7.3.2).

URL:	http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataProvider/↩ containers/VehicleData/contentInstances
Method:	POST
Headers:	Authorization: Basic YWRtaW46YWRtaW4=
Data:	<?xml version="1.0" encoding="UTF-8"?> <VehicleData xmlns:xs="http://127.0.0.1:80/xsd/VehicleData.xsd"> <Position> <latitude>49.866208</latitude> <longitude>8.640403</longitude> <heading>90</heading> </Position> <speed>35</speed> <fuelRange>749</fuelRange> <ESP>false</ESP> </VehicleData>

Step 8: The NSCL sends a RESPONSE with status 201 Created to the AEPrototype.

Status:	201 Created
Headers:	location: nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstances/CI_537453077 server: Jetty(8.1.16.v20140903) content-length: 1073 content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:contentInstance xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime" om2m:id="CI_537453077" href="nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstances/CI_537453077"> <om2m:creationTime>2017-09-09T14:03:23.449+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T14:03:23.449+02:00</om2m:lastModifiedTime> <om2m:delayTolerance>2017-09-09T17:23:23.449+02:00</om2m:delayTolerance>

```
   <om2m:contentSize>310</om2m:contentSize>
   <om2m:content
xmime:contentType="application/xml">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNv
ZGluZz0iVVRGLTgiPz4KPFZlaGljbGVEYXRhIHhtbG5zOnhzPSJodHRwOi8vM
TI3LjAuMC4xOjgwL3hzZC9WZWhpY2xlRGF0YS54c2QiPgogIDxQb3NpdGlvbj
4KICAgIDxsYXRpdHVkZT40OS44NjYyMDg8L2xhdGl0dWRlPgogICAgPGxvb
mdpdHVkZT44LjY0MDQwMzwvbG9uZ2l0dWRlPgogICAgPGhlYWRpbmc+OT
A8L2hlYWRpbmc+CiAgPC9Qb3NpdGlvbj4KICA8c3BlZWQ+MzU8L3NwZWV
kPgogIDxmdWVsUmFuZ2U+NzQ5PC9mdWVsUmFuZ2U+CiAgPEVTUD5mYW
xzZTwvRVNQPgo8L1ZlaGljbGVEYXRhPg==</om2m:content>
</om2m:contentInstance>
```


Step 9: To collect the results of the experiment, firstly the AEPrototype requests to RETRIEVE the nscl/applications/VehicleDataProvider/containers/↩ VehicleData/contentInstances resource that constitutes the VehicleData initially provided by the AEPrototype.

URL:	http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataProvider/↩ containers/VehicleData/contentInstances
Method:	GET
Headers:	Authorization: Basic YWRtaW46YWRtaW4=

Step 10: The RESPONSE in addition to others contains the currentNrOfInstances, which can be used as basis to calculate the savings of the VehicleDataSubStd and VehicleDataSubEnh in contrast to the VehicleData totally provided.

Status:	200 OK
Headers:	content-length: 4629 server: Jetty(8.1.16.v20140903) content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:contentInstances xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime"> <om2m:creationTime>2017-09-09T14:43:23.560+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T15:01:20.957+02:00</om2m:lastModifiedTime> <om2m:currentNrOfInstances>4</om2m:currentNrOfInstances> <om2m:currentByteSize>1239</om2m:currentByteSize> <om2m:contentInstanceCollection> <om2m:contentInstance om2m:id="CI_66122943" href="nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstances/CI_66122943"> <om2m:creationTime>2017-09-09T15:00:29.953+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T15:00:29.954+02:00</om2m:lastModifiedTime> <om2m:delayTolerance>2017-09-09T18:20:29.953+02:00</om2m:delayTolerance> <om2m:contentSize>310</om2m:contentSize> <om2m:content xmime:contentType="application/xml">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz4KPFZlaGljbGVEYXRhIHhtbG5zOnhzPSJodHRwOi8vMTI3LjAuMC4xOjgwL3hzZC9WZWhpY2xlRGF0YS54c2QiPgogIDxQb3NpdGlvbj4KICAgIDxsYXRpdHVkZT40OS44NjYyMDg8L2xhdGl0dWRlPgogICAgPGxvbmdpdHVkZT44LjY0MDQwMzwvbG9uZ2l0dWRlPgogICAgPGhlYWRpbmc+OTA8L2hlYWRpbmc+CiAgPC9Qb3NpdGlvbj4KICA8c3BlZWQ+MzU8L3NwZWVkPgogIDxmdWVsUmFuZ2U+NzUwPC9mdWVsUmFuZ2U+CiAgPEVTUD5mYWxzZTwvRVNQPgo8L1ZlaGljbGVEYXRhPg==</om2m:content> </om2m:contentInstance>

```
<om2m:contentInstance om2m:id="CI_669633475"
href="nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstanc
es/CI_669633475">
<om2m:creationTime>2017-09-
09T15:01:11.779+02:00</om2m:creationTime>
<om2m:lastModifiedTime>2017-09-
09T15:01:11.779+02:00</om2m:lastModifiedTime>
<om2m:delayTolerance>2017-09-
09T18:21:11.779+02:00</om2m:delayTolerance>
<om2m:contentSize>310</om2m:contentSize>
<om2m:content
xmime:contentType="application/xml">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNv
ZGluZz0iVVRGLTgiPz4KPFZlaGljbGVEYXRhIHhtbG5zOnhzPSJodHRwOi8vM
TI3LjAuMC4xOjgwL3hzZC9WZWhpY2xlRGF0YS54c2QiPgogIDxQb3NpdGlvbj
4KICAgIDxsYXRpdHVkZT40OS44NjYyMDg8L2xhdGl0dWRlPgogICAgPGxvb
mdpdHVkZT44LjY0MDQwMzwvbG9uZ2l0dWRlPgogICAgPGhlYWRpbmc+OT
A8L2hlYWRpbmc+CiAgPC9Qb3NpdGlvbj4KICA8c3BlZWQ+MzU8L3NwZWV
kPgogIDxmdWVsUmFuZ2U+NzQ5PC9mdWVsUmFuZ2U+CiAgPEVTUD5mYW
xzZTwvRVNQPgo8L1ZlaGljbGVEYXRhPg==</om2m:content>
</om2m:contentInstance>
<om2m:contentInstance om2m:id="CI_277229607"
href="nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstanc
es/CI_277229607">
<om2m:creationTime>2017-09-
09T15:01:16.913+02:00</om2m:creationTime>
<om2m:lastModifiedTime>2017-09-
09T15:01:16.913+02:00</om2m:lastModifiedTime>
<om2m:delayTolerance>2017-09-
09T18:21:16.913+02:00</om2m:delayTolerance>
<om2m:contentSize>310</om2m:contentSize>
<om2m:content
xmime:contentType="application/xml">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNv
ZGluZz0iVVRGLTgiPz4KPFZlaGljbGVEYXRhIHhtbG5zOnhzPSJodHRwOi8vM
TI3LjAuMC4xOjgwL3hzZC9WZWhpY2xlRGF0YS54c2QiPgogIDxQb3NpdGlvbj
4KICAgIDxsYXRpdHVkZT40OS44NjYyMDg8L2xhdGl0dWRlPgogICAgPGxvb
mdpdHVkZT44LjY0MDQwMzwvbG9uZ2l0dWRlPgogICAgPGhlYWRpbmc+OT
A8L2hlYWRpbmc+CiAgPC9Qb3NpdGlvbj4KICA8c3BlZWQ+MzU8L3NwZWV
kPgogIDxmdWVsUmFuZ2U+MTAwPC9mdWVsUmFuZ2U+CiAgPEVTUD5mY
WxzZTwvRVNQPgo8L1ZlaGljbGVEYXRhPg==</om2m:content>
</om2m:contentInstance>
<om2m:contentInstance om2m:id="CI_761829680"
href="nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstanc
es/CI_761829680">
<om2m:creationTime>2017-09-
09T15:01:20.933+02:00</om2m:creationTime>
<om2m:lastModifiedTime>2017-09-
09T15:01:20.933+02:00</om2m:lastModifiedTime>
<om2m:delayTolerance>2017-09-
09T18:21:20.933+02:00</om2m:delayTolerance>
<om2m:contentSize>309</om2m:contentSize>
<om2m:content
xmime:contentType="application/xml">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNv
ZGluZz0iVVRGLTgiPz4KPFZlaGljbGVEYXRhIHhtbG5zOnhzPSJodHRwOi8vM
TI3LjAuMC4xOjgwL3hzZC9WZWhpY2xlRGF0YS54c2QiPgogIDxQb3NpdGlvbj
4KICAgIDxsYXRpdHVkZT40OS44NjYyMDg8L2xhdGl0dWRlPgogICAgPGxvb
mdpdHVkZT44LjY0MDQwMzwvbG9uZ2l0dWRlPgogICAgPGhlYWRpbmc+OT
A8L2hlYWRpbmc+CiAgPC9Qb3NpdGlvbj4KICA8c3BlZWQ+MzU8L3NwZWV
kPgogIDxmdWVsUmFuZ2U+OTk8L2Z1ZWxSYW5nZT4KICA8RVNQPmZhbH
NlPC9FU1A+CjwvVmVoaWNsZURhdGE+</om2m:content>
</om2m:contentInstance>
```

	</om2m:contentInstanceCollection> <om2m:subscriptionsReference>nscl/applications/VehicleDataProvider/containers/VehicleData/contentInstances/subscriptions</om2m:subscriptionsReference> </om2m:contentInstances>

Step 11: To collect the results of the experiment, secondly the AEPrototype requests to RETRIEVE the nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredStd/contentInstances resource that reflects the filtering of the VehicleDataSubStd.

URL:	http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataSubscriber/↩ containers/VehicleDataFilteredStd/contentInstances
Method:	GET
Headers:	Authorization: Basic YWRtaW46YWRtaW4=

Step 12: The RESPONSE beside others contains the currentNrOfInstances, which can be used as basis to calculate the savings of the VehicleDataSubEnh in contrast to the VehicleData totally provided and the VehicleDataSubStd. According to the request and the subscription VehicleDataSubStd, the response also provides the received NOTIFY messages within contentInstance resources in a contentInstanceCollection. These are the base64 encoded content (i.e., the XML VehicleData), nested in a base64 encoded contentInstance (e.g., nscl/applications/VehicleDataProvider/ containers/VehicleData/contentInstances/CI_669633475), nested in a base64 encoded NOTIFY message.

Status:	200 OK
Headers:	content-length: 10068 server: Jetty(8.1.16.v20140903) content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:contentInstances xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime"> <om2m:creationTime>2017-09-09T14:44:14.291+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T15:01:44.969+02:00</om2m:lastModifiedTime> <om2m:currentNrOfInstances>3</om2m:currentNrOfInstances> <om2m:currentByteSize>5705</om2m:currentByteSize> <om2m:contentInstanceCollection> <om2m:contentInstance om2m:id="CI_175375118" href="nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredStd/contentInstances/CI_175375118"> <om2m:creationTime>2017-09-09T15:01:11.821+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T15:01:11.821+02:00</om2m:lastModifiedTime> <om2m:delayTolerance>2017-09-09T18:21:11.820+02:00</om2m:delayTolerance> <om2m:contentSize>1903</om2m:contentSize> <om2m:content xmime:contentType="application/xml">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiIHN0YW5kYWxvbmU9InllcyI/Pgo8b20ybTpub3RpZnkgeG1sbnM6b20ybT0iaHR0cDovL3VyaS5ldHNpLm9yZy9tMm0iIHhtbG5zOnhtaW1lPSJodHRwOi8vd3d3LnczLm9yZy8yMDA1LzA1L3htbG1pbWUiPgogICAgPG9tMm06c3RhdHVzQ29kZT5TVEFUVVNfQ1JFQVRFRDwvb20ybTpzdGF0dXNDb2RlPgogICAgPG9tMm06cmVwcmVzZW50YXRpb24geG1pbWU6Y29udGVudFR5cGU9ImFwcGxpY2F0aW9uL3htbCI+UEQ5NGJXd2dkbVZ5YzJsdmJqMGlNUzR3SWlCbGJtTnZaR2x1WnowaVZWUkdMVGdpSUhOMFlXNWtZV3h2Ym1VOUlubGxjeUkvUGdvOGIyMHliVHBqYjI1MFpXNTBTVzV6ZEdGdVkyVWdlRzFzYm5NNmIyMHliVDBpYUhSMGNEb3ZMM1Z5YVM1bGRITnBMbTl5Wnk5dE1tMGlJSGh0YWtlMDBpYUhSMGNEb3ZMMUZ5YVM1bGRITnBMbTl5Wnk4eU1tMFpXNTBTVzV6ZEdGdVkyVWdlRzFzYm5NNmIyMHliVDBpYUhSMGNEb3ZMM1Z5YVM1bGRITnBMbTl5Wnk5dE1tMGlJSGh0Ykc1ek9uaHRhVzFsUFNKb2RIUndPaTh2ZDNkM0xuY3pMbTl5Wnk4eU1EQTFMekExTDNodGJHMXBiV1VpSUc5dE1tMDZhV1E5SWtOSlh6WTJPVFl6TXpRM05TSWdhSEpsWmowaWJuTmpiQzloY0hCc2FXTmhkR2x2Ym5NdlZtVm9hV05zWlVSaGRHRlFj

bTkyYVdSbGNpOWpiMjUwWVdsdVpYSnpMMVpsYUdsamJHVkVZWFJoTDJOdmJuU
mxiblJKYm5OMFlXNWpaWE12UTBsZk5qWTVOak16TkRjMUlqNEtJQ0FnSUR4dmJUS
nRPbU55WldGMGFXOXVWR2x0WlQ0eU1ERTNMVEE1TFRBNVZERTFPakF4T2pFeE
xqYzNPU3N3TWpvd01Ed3ZiMjB5YlRwamNtVmhkR2x2YmxScGJXVStDaUFnSUNBOG
IyMHliVHBzWVhOMFRXOWthV1pwWldSVWFXMWxQakl3TVRjdE1Ea3RNRGxVTVR
VNk1ERTZNVEV1TnpjNUt6QXlPakF3UEM5dmJUSnRPbXhoYzNSTmIyUnBabWxsWkZ
ScGJXVStDaUFnSUNBOGIyMHliVHBrWld4aGVWUnZiR1Z5WVc1alpUNHlNREUzTFR
BNUxUQTVWREU0T2pJeE9qRXhMamMzT1Nzd01qb3dNRHd2YjIweWJUcGtaV3hoZV
ZSdmJHVnlZVzVqWlQ0S0lDQWdJRHh2YlRKdE9tTnZiblJsYm5SVGFYcGxQak14TUR3
dmIyMHliVHBqYjI1MFpXNTBVMmw2WlQ0S0lDQWdJRHh2YlRKdE9tTnZiblJsYm5RZ
2VHMXBiV1U2WTI5dWRHVnVkRlI1Y0dVOUltRndjR3hwWTJGMGFXOXVMM2h0Yk
NJK1VFUTVOR0pYZDJka2JWWjVZekpzZG1KcU1HbE5VelIzU1dsQ2JHSnRUblphUjJ4
MVdub3dhVlpXVWtkTVZHZHBVSG8wUzFCR1dteGhSMnhxWWtkV1JWbFlVbWhKU0
doMFlrYzFlazl1YUhwUVUwcHZaRWhTZDA5cE9IWk5WRWt6VEdwQmRVMUROSGh
QYW1kM1RETm9lbHBET1ZkYVYyaHdXVEo0YkZKSFJqQlpVelUwWXpKUmFWQm5i
MmRKUkhoUllqTk9jR1JIYkhaaWFqUkxTVU5CWjBsRWVITlpXRkp3WkVoV2ExcFVO
REJQVXpRMFRtcFplVTFFWnpoTU1uaG9aRWRzTUdSWFVteFFaMjluU1VOQloxQkhlS
FppYldSd1pFaFdhMXBVTkRSTWFsa3dUVVJSZDAxNmQzWmlSemwxV2pKc01HUlhVb
XhRWjI5blNVTkJaMUJIYUd4WlYxSndZbTFqSzA5VVFUaE1NbWhzV1ZkU2NHSnRZe
XREYVVGblVFTTVVV0l6VG5Ca1IyeDJZbW8wUzBsRFFUaGpNMEpzV2xkUkswMTZ
WVGhNTTA1M1dsZFdhMUJuYjJkSlJIaHRaRmRXYzFWdFJuVmFNbFVyVG5wUk5WQ
kRPVzFrVjFaelZXMUdkVm95VlN0RGFVRm5VRVZXVkZWRU5XMVpWM2g2V2xSM
2RsSldUbEZRWjI4NFRERmFiR0ZIYkdwaVIxWkZXVmhTYUZCblBUMDhMMjl0TW0w
NlkyOXVkR1Z1ZEQ0S1BDOXZiVEp0T21OdmJuUmxiblJKYm5OMFlXNWpaVDRLPC9
vbTJtOnJlcHJlc2VudGF0aW9uPgogICAgPG9tMm06c3Vic2NyaXB0aW9uUmVmZXJlbm
NlPm5zY2wvYXBwbGljYXRpb25zL1ZlaGljbGVEYXRhUHJvdmlkZXIvY29udGFpbmVy
cy9WZWhpY2xlRGF0YS9jb250ZW50SW5zdGFuY2VzL3N1YnNjcmlwdGlvbnMvU1VCX
zMyMjg4MzAxOTwvb20ybTpzdWJzY3JpcHRpb25SZWZlcmVuY2U+Cjwvb20ybTpub3R
pZnk+Cg==</om2m:content>
</om2m:contentInstance>
<om2m:contentInstance om2m:id="CI_349759828"
href="nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredStd/contentIns
tances/CI_349759828">
<om2m:creationTime>2017-09-09T15:01:16.943+02:00</om2m:creationTime>
<om2m:lastModifiedTime>2017-09-
09T15:01:16.943+02:00</om2m:lastModifiedTime>
<om2m:delayTolerance>2017-09-09T18:21:16.942+02:00</om2m:delayTolerance>
<om2m:contentSize>1903</om2m:contentSize>
<om2m:content
xmime:contentType="application/xml">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0
iVVRGLTgiIHN0YW5kYWxvbmU9InllcyI/Pgo8b20ybTpub3RpZnkgeG1sbnM6b20ybT0ia
HR0cDovL3VyaS5ldHNpLm9yZy9tMm0iIHhtbG5zOnhtaW1lPSJodHRwOi8vd3d3LnczLm
9yZy8yMDA1LzA1L3htbG1pbWUiPgogICAgPG9tMm06c3RhdHVzQ29kZT5TVEFUVVN
fQ1JFQVRFRDwvb20ybTpzdGF0dXNDb2RlPgogICAgPG9tMm06cmVwcmVzZW50YXR
pb24geG1pbWU6Y29udGVudFR5cGU9ImFwcGxpY2F0aW9uL3htbCI+UEQ5NGJXd2dkb
VZ5YzJsdmJqMGlNUzR3SWlCbGJtTnZaR2x1WnowaVZWUkdMVGdpSUhOMFlXNWtZ
V3h2Ym1VOUlubGxjeUkvUGdvOGIyMHliVHBqYjI1MFpXNTBTVzV6ZEdGdVkyVWdl
RzFzYm5NNmIyMHliVDBpYUhSMGNEb3ZMM1Z5YVM1bGRITnBMbTl5Wnk5dE1tM
GlJSGh0Ykc1ek9uaHRhVzFsUFNKb2RIUndPaTh2ZDNkM0xuY3pMbTl5Wnk4eU1EQTF
MekExTDNodGJHMXBiV1VpSUc5dE1tMDZhV1E5SWtOSlh6STNOekl5T1RZd055SWdh
SEpsWmowaWJuTmpiQzloY0hCc2FXTmhkR2x2Ym5NdlZtVm9hV05zWlVSaGRHRlFjbT
kyYVdSbGNpOWpiMjUwWVdsdVpYSnpMMVpsYUdsamJHVkVZWFJoTDJOdmJuUmxi
blJKYm5OMFlXNWpaWE12UTBsZk1qYzNNakk1TmpBM0lqNEtJQ0FnSUR4dmJUSnRP
bU55WldGMGFXOXVWR2x0WlQ0eU1ERTNMVEE1TFRBNVZERTFPakF4T2pFMkxqa
3hNeXN3TWpvd01Ed3ZiMjB5YlRwamNtVmhkR2x2YmxScGJXVStDaUFnSUNBOGIyM
HliVHBzWVhOMFRXOWthV1pwWldSVWFXMWxQakl3TVRjdE1Ea3RNRGxVTVRVN
k1ERTZNVFl1T1RFekt6QXlPakF3UEM5dmJUSnRPbXhoYzNSTmIyUnBabWxsWkZScG
JXVStDaUFnSUNBOGIyMHliVHBrWld4aGVWUnZiR1Z5WVc1alpUNHlNREUzTFRBN
UxUQTVWREU0T2pJeE9qRTJMamt4TXlzd01qb3dNRHd2YjIweWJUcGtaV3hoZVZSdmJ
HVnlZVzVqWlQ0S0lDQWdJRHh2YlRKdE9tTnZiblJsYm5SVGFYcGxQak14TUR3dmIyM
HliVHBqYjI1MFpXNTBVMmw2WlQ0S0lDQWdJRHh2YlRKdE9tTnZiblJsYm5RZ2VHM
XBiV1U2WTI5dWRHVnVkRlI1Y0dVOUltRndjR3hwWTJGMGFXOXVMM2h0YkNJK1V
FUTVOR0pYZDJka2JWWjVZekpzZG1KcU1HbE5VelIzU1dsQ2JHSnRUblphUjJ4MVdub3
dhVlpXVWtkTVZHZHBVSG8wUzFCR1dteGhSMnhxWWtkV1JWbFlVbWhKU0doMFlrY
zFlazl1YUhwUVUwcHZaRWhTZDA5cE9IWk5WRWt6VEdwQmRVMUROSGhQYW1k
M1RETm9lbHBET1ZkYVYyaHdXVEo0YkZKSFJqQlpVelUwWXpKUmFWQm5iMmRK
UkhoUllqTk9jR1JIYkhaaWFqUkxTVU5CWjBsRWVITlpXRkp3WkVoV2ExcFVOREJQV

XpRMFRtcFplVTFFWnpoTU1uaG9aRWRzTUdSWFVteFFaMjluU1VOQloxQkhlSFppYld
Sd1pFaFdhMXBVTkRSTWFsa3dUVVJSZDAxNmQzWmlSemwxV2pKc01HUlhVbXhRW
jI5blNVTkJaMUJIYUd4WlYxSndZbTFqSzA5VVFUaE1NbWhzV1ZkU2NHSnRZeXREY
VVGblVFTTVVV0l6VG5Ca1IyeDJZbW8wUzBsRFFUaGpNMEpzV2xkUkswMTZWVGh
NTTA1M1dsZFdhMUJuYjJkSlJIaHRaRmRXYzFWdFJuVmFNbFVyVFZSQmQxQkRPVz
FrVjFaelZXMUdkVm95VlN0RGFVRm5VRVZXVkZWRU5XMVpWM2g2V2xSM2RsSld
UbEZRWjI4NFRERmFiR0ZIYkdwaVIxWkZXVmhTYUZCblBUMDhMMjl0TW0wNlkyO
XVkR1Z1ZEQ0S1BDOXZiVEp0T21OdmJuUmxiblJKYm5OMFlXNWpaVDRLPC9vbTJt
OnJlcHJlc2VudGF0aW9uPgogICAgPG9tMm06c3Vic2NyaXB0aW9uUmVmZXJlbmNlPm5
zY2wvYXBwbGljYXRpb25zL1ZlaGljbGVEYXRhUHJvdmlkZXIvY29udGFpbmVycy9WZ
WhpY2xlRGF0YS9jb250ZW50SW5zdGFuY2VzL3N1YnNjcmlwdGlvbnMvU1VCXzMyMj
g4MzAxOTwvb20ybTpzdWJzY3JpcHRpb25SZWZlcmVuY2U+Cjwvb20ybTpub3RpZnk+C
g==</om2m:content>
</om2m:contentInstance>
<om2m:contentInstance om2m:id="CI_193870689"
href="nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredStd/contentIns
tances/CI_193870689">
<om2m:creationTime>2017-09-09T15:01:20.992+02:00</om2m:creationTime>
<om2m:lastModifiedTime>2017-09-
09T15:01:20.992+02:00</om2m:lastModifiedTime>
<om2m:delayTolerance>2017-09-09T18:21:20.992+02:00</om2m:delayTolerance>
<om2m:contentSize>1899</om2m:contentSize>
<om2m:content
xmime:contentType="application/xml">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0
iVVRGLTgiIHN0YW5kYWxvbmU9InllcyI/Pgo8b20ybTpub3RpZnkgeG1sbnM6b20ybT0ia
HR0cDovL3VyaS5ldHNpLm9yZy9tMm0iIHhtbG5zOnhtaW1lPSJodHRwOi8vd3d3LnczLm
9yZy8yMDA1LzA1L3htbG1pbWUiPgogICAgPG9tMm06c3RhdHVzQ29kZT5TVEFUVVN
fQ1JFQVRFRDwvb20ybTpzdGF0dXNDb2RlPgogICAgPG9tMm06cmVwcmVzZW50YXR
pb24geG1pbWU6Y29udGVudFR5cGU9ImFwcGxpY2F0aW9uL3htbCI+UEQ5NGJXd2dkb
VZ5YzJsdmJqMGlNUzR3SWlCbGJtTnZaR2x1WnowaVZWUkdMVGdpSUhOMFlXNWtZ
V3h2Ym1VOUlubGxjeUkvUGdvOGIyMHliVHBqYjI1MFpXNTBTVzV6ZEdGdVkyVWdl
RzFzYm5NNmIyMHliVDBpYUhSMGNEb3ZMM1Z5YVM1bGRITnBMbTl5Wnk5dE1tM
GlJSGh0Ykc1ek9uaHRhVzFsUFNKb2RIUndPaTh2ZDNkM0xuY3pMbTl5Wnk4eU1EQTF
MekExTDNodGJHMXBiV1VpSUc5dE1tMDZhV1E5SWtOSlh6YzJNVGd5T1RZNE1DSW
dhSEpsWmowaWJuTmpiQzloY0hCc2FXTmhkR2x2Ym5NdlZtVm9hV05zWlVSaGRHRlFj
bTkyYVdSbGNpOWpiMjUwWVdsdVpYSnpMMVpsYUdsamJHVkVZWFJoTDJOdmJuU
mxiblJKYm5OMFlXNWpaWE12UTBsZk56WXhPREk1Tmpnd0lqNEtJQ0FnSUR4dmJUSn
RPbU55WldGMGFXOXVWR2x0WlQ0eU1ERTNMVEE1TFRBNVZERTFPakF4T2pJd0x
qa3pNeXN3TWpvd01Ed3ZiMjB5YlRwamNtVmhkR2x2YmxScGJXVStDaUFnSUNBOGIy
MHliVHBzWVhOMFRXOWthV1pwWldSVWFXMWxQakl3TVRjdE1Ea3RNRGxVTVRV
Nk1ERTZNakF1T1RNekt6QXlPakF3UEM5dmJUSnRPbXhoYzNSTmIyUnBabWxsWkZSc
GJXVStDaUFnSUNBOGIyMHliVHBrWld4aGVWUnZiR1Z5WVc1alpUNHlNREUzTFRB
NUxUQTVWREU0T2pJeE9qSXdMamt6TXlzd01qb3dNRHd2YjIweWJUcGtaV3hoZVZSd
mJHVnlZVzVqWlQ0S0lDQWdJRHh2YlRKdE9tTnZiblJsYm5SVGFYcGxQak13T1R3dmI
yMHliVHBqYjI1MFpXNTBVMmw2WlQ0S0lDQWdJRHh2YlRKdE9tTnZiblJsYm5RZ2V
HMXBiV1U2WTI5dWRHVnVkRlI1Y0dVOUltRndjR3hwWTJGMGFXOXVMM2h0YkNJ
K1VFUTVOR0pYZDJka2JWWjVZekpzZG1KcU1HbE5VelIzU1dsQ2JHSnRUblphUjJ4MV
dub3dhVlpXVWtkTVZHZHBVSG8wUzFCR1dteGhSMnhxWWtkV1JWbFlVbWhKU0doM
FlrYzFlazl1YUhwUVUwcHZaRWhTZDA5cE9IWk5WRWt6VEdwQmRVMUROSGhQY
W1kM1RETm9lbHBET1ZkYVYyaHdXVEo0YkZKSFJqQlpVelUwWXpKUmFWQm5iM
mRKUkhoUllqTk9jR1JIYkhaaWFqUkxTVU5CWjBsRWVITlpXRkp3WkVoV2ExcFVORE
JQVXpRMFRtcFplVTFFWnpoTU1uaG9aRWRzTUdSWFVteFFaMjluU1VOQloxQkhlSFpp
YldSd1pFaFdhMXBVTkRSTWFsa3dUVVJSZDAxNmQzWmlSemwxV2pKc01HUlhVbXh
RWjI5blNVTkJaMUJIYUd4WlYxSndZbTFqSzA5VVFUaE1NbWhzV1ZkU2NHSnRZeXR
EYVVGblVFTTVVV0l6VG5Ca1IyeDJZbW8wUzBsRFFUaGpNMEpzV2xkUkswMTZWV
GhNTTA1M1dsZFdhMUJuYjJkSlJIaHRaRmRXYzFWdFJuVmFNbFVyVDFSck9Fd3lXak
ZhVjNoVFdWYzFibHBVTkV0SlEwRTRVbFpPVVZCdFdtaGlTRTVzVUVNNVJsVXhRU
3REYW5kMlZtMVdiMkZYVG5OYVZWSm9aRWRGS3p3dmIyMHliVHBqYjI1MFpXNT
BQZ284TDI5dE1tMDZZMjl1ZEdWdWRFbHVjM1JoYm1ObFBnbz08L29tMm06cmVwcm
VzZW50YXRpb24+CiAgICA8b20ybTpzdWJzY3JpcHRpb25SZWZlcmVuY2U+bnNjbC9hc
HBsaWNhdGlvbnMvVmVoaWNsZURhdGFQcm92aWRlci9jb250YWluZXJzL1ZlaGljbGV
EYXRhL2NvbnRlbnRJbnN0YW5jZXMvc3Vic2NyaXB0aW9ucy9TVUJfMzIyODgzMDE5
PC9vbTJtOnN1YnNjcmlwdGlvblJlZmVyZW5jZT4KPC9vbTJtOm5vdGlmeT4K</om2m:co
ntent>
</om2m:contentInstance>
</om2m:contentInstanceCollection>

	<om2m:subscriptionsReference>nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredStd/contentInstances/subscriptions</om2m:subscriptionsReference> </om2m:contentInstances>

Step 13: To collect the results of the experiment, thirdly the AEPrototype requests to RETRIEVE the nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredEnh/contentInstances resource that reflects the filtering of the VehicleDataSubEnh.

URL:	http://127.0.0.1:8080/om2m/nscl/applications/VehicleDataSubscriber/↩ containers/VehicleDataFilteredEnh/contentInstances
Method:	GET
Headers:	Authorization: Basic YWRtaW46YWRtaW4=

Step 14: The RESPONSE in addition to others contains the currentNrOfInstances, which can be used as basis to calculate the savings of the VehicleDataSubEnh in contrast to the VehicleData totally provided and the VehicleDataSubStd. According to the request and the subscription VehicleDataSubEnh, the response also provides the received NOTIFY message within a contentInstance resource in a contentInstanceCollection. This is the base64 encoded content (i.e., the XML VehicleData), nested in a base64 encoded contentInstance (e.g., nscl/applications/VehicleDataProvider/ containers/VehicleData/contentInstances/CI_761829680), nested in a base64 encoded NOTIFY message.

Status:	200 OK
Headers:	content-length: 3818 server: Jetty(8.1.16.v20140903) content-type: application/xml;charset=ISO-8859-1
Data:	<?xml version="1.0" encoding="UTF-8" standalone="yes"?> <om2m:contentInstances xmlns:om2m="http://uri.etsi.org/m2m" xmlns:xmime="http://www.w3.org/2005/05/xmlmime"> <om2m:creationTime>2017-09-09T14:44:37.206+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T15:01:20.995+02:00</om2m:lastModifiedTime> <om2m:currentNrOfInstances>1</om2m:currentNrOfInstances> <om2m:currentByteSize>1899</om2m:currentByteSize> <om2m:contentInstanceCollection> <om2m:contentInstance om2m:id="CI_23657356" href="nscl/applications/VehicleDataSubscriber/containers/VehicleDataFilteredEnh/contentInstances/CI_23657356"> <om2m:creationTime>2017-09-09T15:01:20.988+02:00</om2m:creationTime> <om2m:lastModifiedTime>2017-09-09T15:01:20.988+02:00</om2m:lastModifiedTime> <om2m:delayTolerance>2017-09-09T18:21:20.988+02:00</om2m:delayTolerance> <om2m:contentSize>1899</om2m:contentSize> <om2m:content xmime:contentType="application/xml">PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiIHN0YW5kYWxvbmU9InllcyI/Pgo8b20ybTpub3RpZnkgeG1sbnM6b20ybT0iaHR0cDovL3VyaS5ldHNpLm9yZy9tMm0iIHhtbG5zOnhtaW1lPSJodHRwOi8vd3d3LnczLm9yZy8yMDA1LzA1L3htbG1pbWUiPgogICAgPG9tMm06c3RhdHVzQ29kZT5TVEFUVVNfQ1JFQVRFRDwvb20ybTpzdGF0dXNDb2RlPgogICAgPG9tMm06cmVwcmVzZW50YXRpb24geG1pbWU6Y29udGVudFR5cGU9ImFwcGxpY2F0aW9uL3htbCI+UEQ5NGJXd2dkbVZ5YzJsdmJqMGlNUzR3SWlCbGJtTnZaR2x1WnowaVZWUkdMVGdpSUhOMFlXNWtZV3h2Ym1VOUlubGxjeUkvUGdvOGIyMHliVHBqYjI1MFpXNTBTVzV6ZEdGdVkyVWdlRzFzYm5NNmIyMHliVDBpYUhSMGNEb3ZMM1Z5YVM1bGRITnBMbTl5Wnk5dE1tMGlJSGh0Ykc1ek9uaHRhVzFsUFNKb2RIUndPaTh2ZDNkM0xuY3pMbTl5Wnk4eU1EQTFMekExTDNodGJHMXBiV1VpSUc5dE1tMDZhV1E5SWtOSlh6YzJNVGd5T1RZNE1DSWdhSEpsWmowaWJuTmpiQzloY0hCc2FXTmhkR2x2ZHNkM0xuY3pMbTl5Wnk4eU1EQTFMekExTDNodGJHMXBiV1VpSUc5dE1tMDZhV1E5SWtOSlh6YzJNVGd5T1RZNE1DSWdhSEpsWmowaWJuTmpiQzloY0hCc2FXTmhkR2x2ZHNkM0xuY3pMbTl5Wnk4eU1EQTFMekExTDNodGJHMXBiV1VpSUc5dE1tMDZhV1E5SWtOSlh6YzJNVGd5T1RZNE1DSWdhSEpsWmowaWJuTmpiQzloY0hCc2FXTmhkR2x2ZHNkM0xuY3pMbTl5Wnk4eU1EQTFMekExTDNodGJHMXBiV1VpSUc5dE1tMDZhV1E5SWtOSlh6YzJNVGd5T1RZNE1DSWdhSEpsWmowaWJuTmpiQzloY0hCc2FXTmhkR2x2ZHNkM0xuY3pMbTl5Wnk4dE1tMGlJSGRhRlFjbTkyYVdSbGNpOWpiMjUwWVdsdVpYSnpMMVpsYUdsamJHVkVZWFJoTDJOdmJuUmxiblJKYm5OMFlXNWpaWE12UTBsZk56WXhPREk1Tmpnd0lqNEtJQ0FnSUR4dmJUSn

RPbU55WldGMGFXOXVWR2x0WlQ0eU1ERTNMVEE1TFRBNVZERTFPakF4T2pJd0x
qa3pNeXN3TWpvd01Ed3ZiMjB5YlRwamNtVmhkR2x2YmxScGJXVStDaUFnSUNBOGIy
MHliVHBzWVhOMFRXOWthV1pwWldSVWFXMWxQakl3TVRjdE1Ea3RNRGxVTVRV
Nk1ERTZNakF1T1RNekt6QXlPakF3UEM5dmJUSnRPbXhoYzNSTmIyUnBabWxsWkZSc
GJXVStDaUFnSUNBOGIyMHliVHBrWld4aGVWUnZiR1Z5WVc1alpUNHlNREUzTFRB
NUxUQTVWREU0T2pJeE9qSXdMamt6TXlzd01qb3dNRHd2YjIweWJUcGtaV3hoZVZSd
mJHVnlZVzVqWlQ0S0lDQWdJRHh2YlRKdE9tTnZiblJsYm5SVGFYcGxQak13T1R3dmI
yMHliVHBqYjI1MFpXNTBVMmw2WlQ0S0lDQWdJRHh2YlRKdE9tTnZiblJsYm5RZ2V
HMXBiV1U2WTI5dWRHVnVkRlI1Y0dVOUltRndjR3hwWTJGMGFXOXVMM2h0YkNJ
K1VFUTVOR0pYZDJka2JWWjVZekpzZG1KcU1HbE5VelIzU1dsQ2JHSnRUblphUjJ4MV
dub3dhVlpXVWtkTVZHZHBVSG8wUzFCR1dteGhSMnhxWWtkV1JWbFlVbWhKU0doM
FlrYzFlazl1YUhwUVUwcHZaRWhTZDA5cE9IWk5WRWt6VEdwQmRVMUROSGhQY
W1kM1RETm9lbHBET1ZkYVYyaHdXVEo0YkZKSFJqQlpVelUwWXpKUmFWQm5iM
mRKUkhoUllqTk9jR1JIYkhaaWFqUkxTVU5CWjBsRWVITlpXRkp3WkVoV2ExcFVORE
JQVXpRMFRtcFplVTFFWnpoTU1uaG9aRWRzTUdSWFVteFFaMjluU1VOQloxQkhlSFpp
YldSd1pFaFdhMXBVTkRSTWFsa3dUVVJSZDAxNmQzWmlSemwxV2pKc01HUlhVbXh
RWjI5blNVTkJaMUJIYUd4WlYxSndZbTFqSzA5VVFUaE1NbWhzV1ZkU2NHSnRZeXR
EYVVGblVFTTVVV0l6VG5Ca1IyeDJZbW8wUzBsRFFUaGpNMEpzV2xkUkswMTZWV
GhNTTA1M1dsZFdhMUJuYjJkSlJIaHRaRmRXYzFWdFJuVmFNbFVyVDFSck9Fd3lXak
ZhVjNoVFdWYzFibHBVTkV0SlEwRTRVbFpPVVZCdFdtaGlTRTVzVUVNNVJsVXhRU
3REYW5kMlZtMVdiMkZYVG5OYVZWSm9aRWRGS3p3dmIyMHliVHBqYjI1MFpXNT
BQZ284TDI5dE1tMDZZMjl1ZEdWdWRFbHVjM1JoYm1ObFBnbz08L29tMm06cmVwcm
VzZW50YXRpb24+CiAgICA8b20ybTpzdWJzY3JpcHRpb25SZWZlcmVuY2U+bnNjbC9hc
HBsaWNhdGlvbnMvVmVoaWNsZURhdGFQcm92aWRlci9jb250YWluZXJzL1ZlaGljbGV
EYXRhL2NvbnRlbnRJbnN0YW5jZXMvc3Vic2NyaXB0aW9ucy9TVUJfMTUxNzE0NDU
2PC9vbTJtOnN1YnNjcmlwdGlvblJlZmVyZW5jZT4KPC9vbTJtOm5vdGlmeT4K</om2m:c
ontent>
</om2m:contentInstance>
</om2m:contentInstanceCollection>

<om2m:subscriptionsReference>nscl/applications/VehicleDataSubscriber/containers/Vehicle
DataFilteredEnh/contentInstances/subscriptions</om2m:subscriptionsReference>
</om2m:contentInstances>

The steps described are intended to support various experiments. The example used here relates to the Experiment 1, as described in the following Section 7.3.2. However, the example can also be used as basis for other experiments and estimations. These are usual modifications/variants and the steps that have to be modified:

- **Different XML application data format:**
  In order to use a different XML data format, the Step 1 has to be adopted: The data in the related request should be the new XML application data format including the reference to the location of the corresponding XSD file.
  Modified data may also require adoptions of Steps 3, 5, and 7.
- **Different data subscriptions:**
  To adopt the VehicleDataSubStd or the VehicleDataSubEnh subscription in order to perform different experiments and estimations, the Steps 3 and 5 have to be modified accordingly.

7.3.2 Experiment 1: Remaining fuelRange with Single-Steps Provider, Variant 1

The first experiment is derived from the condensed ASDP scenario (see Section 5.1): The AE3 Vehicle Maintenance subscribes to the remaining fuelRange, in order to initiate subsequent use cases, e.g., to offer nearby gas stations, when the remaining fuelRange is below a certain

threshold. In this experiment, the threshold should be less than 100 KM remaining fuelRange. For this reason, the AE3 subscribes to VehicleData that is provided by the AE1 Vehicle Data Provider.

The AEPrototype is used as test harness by means of setting up the VehicleDataProvider and respective containers, as well as two subscriptions. The notification target of the two subscriptions are the respective containers VehicleDataFilteredStd and VehicleDataFilteredEnh in the VehicleDataSubscriber application, which mimic the AE3 Vehicle Maintenance. Furthermore, the AEPrototype provides the VehicleData to the respective container resource in the CSE.

Assumptions

- The vehicle provides the VehicleData with each single-step modification of the fuelRange attribute.
- The fuelRange continuously decreases, starting from 750 KM.

Subscriptions

- **VehicleDataSubStd:** Since the event, when the remaining fuelRange falls below 100 KM must not be missed, no filter criterion can be applied, as the standard oneM2M data exchange capabilities do not facilitate appropriate filter criteria (cf. Section 5.3).
- **VehicleDataSubEnh:** The subscription VehicleDataSubEnh uses the enhanced data exchange capabilities as developed and proposed in this research by means of the following EPL statement: select * from VehicleData where fuelRange < 100;

Test and Estimations

The CSE prototype is tested with boundary values 750, 749, 100, and 99, that are derived from the VehicleDataSubEnh. The test is performed according to the setup and test strategy as presented in Section 7.3.1. Based on this, further estimations are made:

$$(1.1) \quad (750 - 99) + 1 = 652$$

According to 1.1., the VehicleDataSubStd triggers 652 NOTIFY messages until the fuelRange 99 KM is reached, which result in 652 contentInstance resources created at the VehicleDataFilteredStd.

The VehicleDataSubEnh triggers only one NOTIFY message exactly when the fuelRange 99 KM is reached, which result in only one contentInstance created at the VehicleDataFilteredEnh.

$$(1.2) \quad \left(1 - \left(\frac{1}{652}\right)\right) * 100 = 99{,}84662$$

According to 1.2, the saving of NOTIFY messages of the VehicleDataSubEnh is 99,84662 % until a fuelRange of 99 KM is reached.

$$(1.3) \quad \left(1 - \left(\frac{99 + 1}{750 + 1}\right)\right) * 100 = 86{,}68442$$

Considering the number of NOTIFY messages until a remaining fuelRange of 0 KM is reached, the VehicleDataSubStd triggers 751 NOTIFY messages, and the VehicleDataSubEnh triggers 100 NOTIFY messages, which lead to the corresponding number of contentInstance resources in the VehicleDataFilteredStd and VehicleDataFilteredEnh data containers. This results in savings of 86,68442 % for the enhanced data exchange capabilities as utilised by the VehicleDataSubEnh (see 1.3).

Figure 7.8 illustrates the number of notifications for both subscriptions during decrease of fuelRange from 750 KM to 0 KM according to the conditions of this Experiment 1.

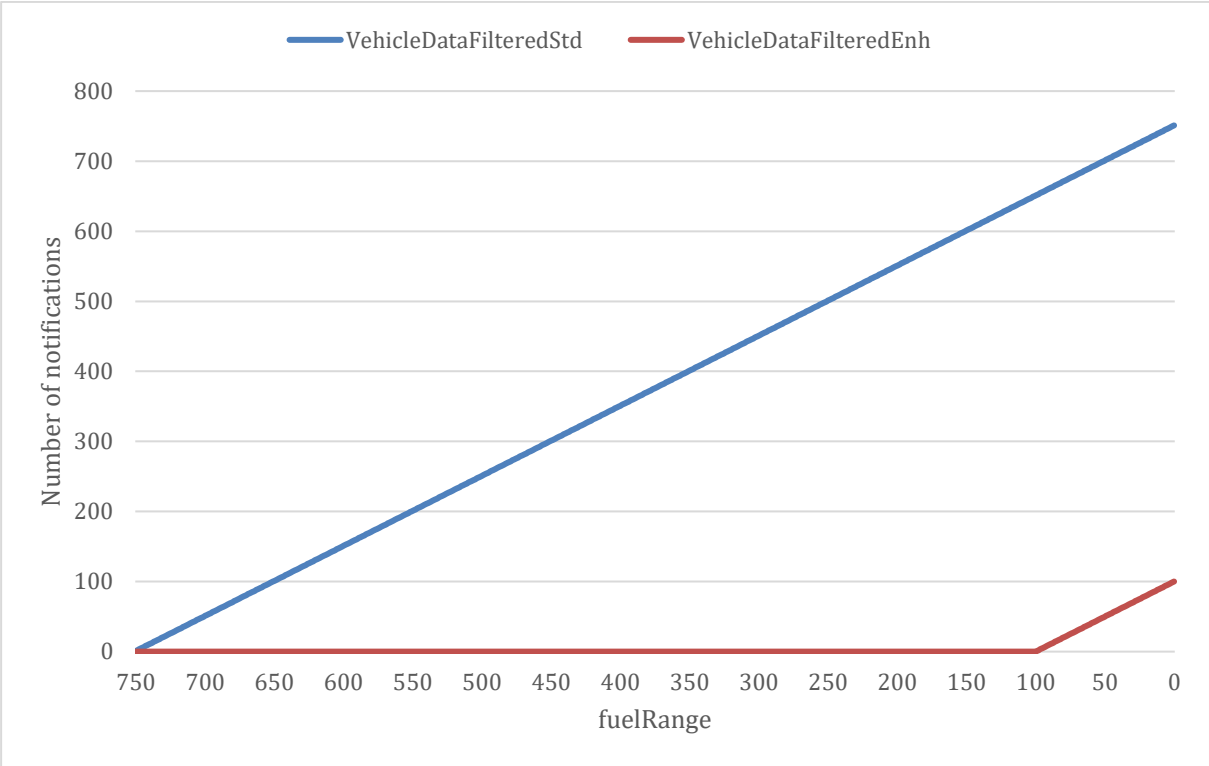

Figure 7.8: Comparison of the number of notifications for VehicleDataSubStd and VehicleDataSubEnh during fuelRange decrease according to the conditions of Experiment 1

7.3.3 Experiment 2: Remaining fuelRange with Decimal-Steps Provider

The second experiment constitutes a variation of the Experiment 1 (see Section 7.3.2). Most important difference is that the AE1 Vehicle Data Provider now provides the VehicleData only with each decimal-step modification (i.e., decrease) of the remaining fuelRange. Thus, the threshold should be less than or equal to 100 KM remaining fuelRange.

Assumptions

- The vehicle provides the VehicleData with each decimal-step modification of the fuelRange attribute.
- The fuelRange continuously decreases, starting from 750 KM.

Subscriptions

- **VehicleDataSubStd:** Since the event, when the remaining fuelRange is equal to or less than 100 KM must not be missed, no filter criterion can be applied, as the standard oneM2M data exchange capabilities do not facilitate appropriate filter criteria (cf. Section 5.3).
- **VehicleDataSubEnh:** The subscription VehicleDataSubEnh uses the enhanced data exchange capabilities as developed and proposed in this research by means of the following EPL statement: select * from VehicleData where fuelRange <= 100; In order to avoid getting the first notification at 90 KM, the EPL statement is modified.

Test and Estimations

The CSE prototype is tested with boundary values 750, 740, 110, 100, and 90, that are derived from the VehicleDataSubEnh. The test is performed according to the setup and test strategy as presented in Section 7.3.1. Based on this, further estimations are made:

$$(2.1) \quad \left(\frac{750 - 100}{10}\right) + 1 = 66$$

According to 2.1., the VehicleDataSubStd triggers 66 NOTIFY messages until the fuelRange 100 KM is reached, which result in 656 contentInstance resources created at the VehicleDataFilteredStd.

The VehicleDataSubEnh triggers only one NOTIFY message exactly when the fuelRange 100 KM is reached, which result in only one contentInstance created at the VehicleDataFilteredEnh.

$$(2.2) \quad \left(1 - \left(\frac{1}{66}\right)\right) * 100 = 98,\overline{48}$$

According to 2.2, the saving of NOTIFY messages of the VehicleDataSubEnh is 98,48 % up to a fuelRange of 100 KM.

$$(2.3) \quad \left(1 - \left(\frac{10 + 1}{75 + 1}\right)\right) * 100 = 85{,}5263$$

Considering the number of NOTIFY messages until a remaining fuelRange of 0 KM is reached, the VehicleDataSubStd triggers 76 NOTIFY messages, and the VehicleDataSubEnh triggers 11 NOTIFY message, which leads to the corresponding number of contentInstance resources

in the VehicleDataFilteredStd and VehicleDataFilteredEnh data containers. This results in savings of 85,5263 % for the enhanced data exchange capabilities as utilised by the VehicleDataSubEnh (see 2.3).

Figure 7.9 illustrates the number of notifications for both subscriptions during decrease of fuelRange from 750 KM to 0 KM according to the conditions of this Experiment 2.

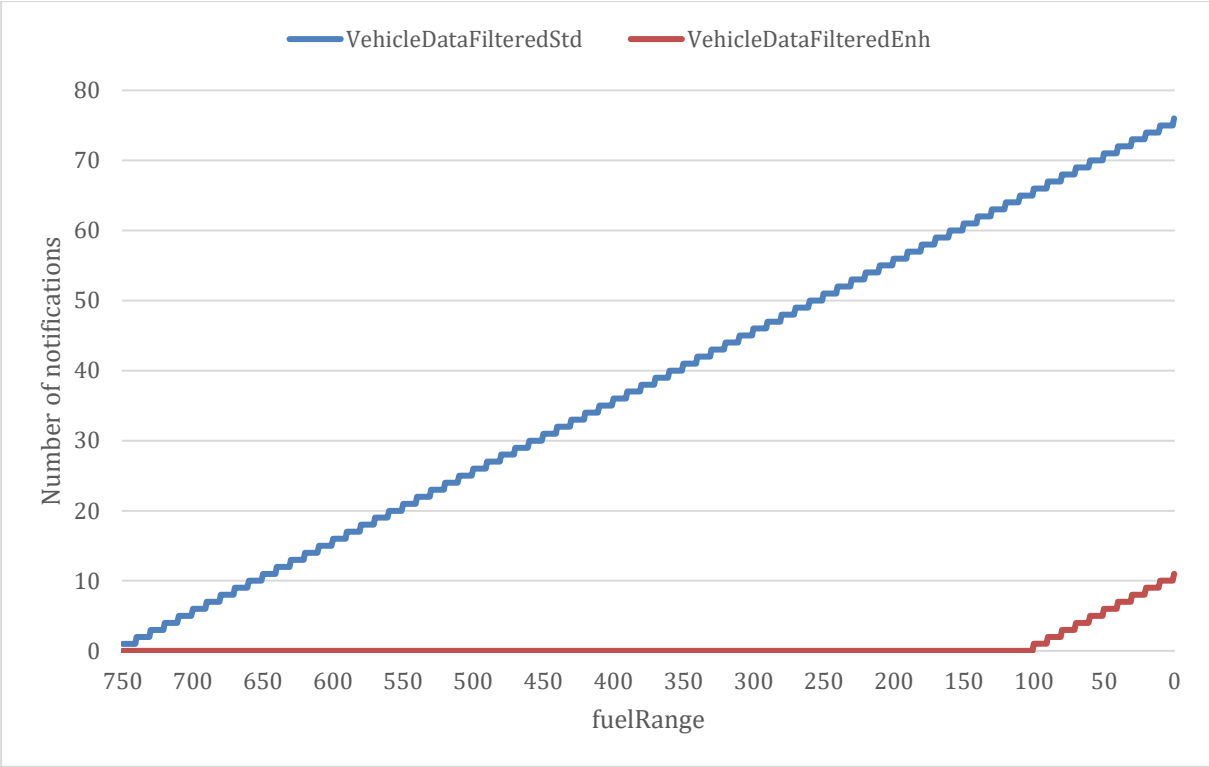

Figure 7.9: Comparison of the number of notifications for VehicleDataSubStd and VehicleDataSubEnh during fuelRange decrease according to the conditions of Experiment 2

7.3.4 Experiment 3: Remaining fuelRange with Single-Steps Provider, Variant 2

The third experiment constitutes a variation of the two previous experiments (see Sections 7.3.2 and 7.3.3). The difference is the VehicleDataSubEnh that now not only subscribes to a fuelRange lower than or equal to 100, but also wants to receive the related events only at intervals of 10 KM decrease of remaining fuelRange. This mimic the decimal-steps provider of Experiment 2, although the Vehicle Data Provider here provides single-step fuelRange changes.

Assumptions

- The vehicle provides the VehicleData with each single-step modification of the fuelRange attribute.
- The fuelRange continuously decreases, starting from 750 KM.

Subscriptions

- **VehicleDataSubStd:** Since the event, when the remaining fuelRange falls below 100 KM must not be missed, no filter criterion can be applied, as the standard oneM2M data exchange capabilities do not facilitate appropriate filter criteria (cf. Section 5.3).
- **VehicleDataSubEnh:** The subscription VehicleDataSubEnh uses the enhanced data exchange capabilities as developed and proposed in this research by means of the following EPL statement: select * from VehicleData where fuelRange <= 100 and fuelRange%10=0;

Test and Estimations

The CSE prototype is tested with boundary values 750, 110, 101, 100, 99 and 90, that are derived from the VehicleDataSubEnh. The test is performed according to the setup and test strategy as presented in Section 7.3.1. Based on this, further estimations are made:

$$(3.1) \quad (750 - 100) + 1 = 651$$

According to 3.1., the VehicleDataSubStd triggers 651 NOTIFY messages until the fuelRange 100 KM is reached, which result in 651 contentInstance resources created at the VehicleDataFilteredStd.

The VehicleDataSubEnh triggers only one NOTIFY message exactly when the fuelRange 100 KM is reached, which result in only one contentInstance created at the VehicleDataFilteredEnh.

$$(3.2) \quad \left(1 - \left(\frac{1}{651}\right)\right) * 100 = 99{,}84639$$

According to 3.2, the saving of NOTIFY messages of the VehicleDataSubEnh is 99,84639 % up to a fuelRange of 100 KM.

$$(3.3) \quad \left(1 - \left(\frac{10 + 1}{750 + 1}\right)\right) * 100 = 98{,}53528$$

Considering the number of NOTIFY messages up to when a remaining fuelRange of 0 KM is reached, the VehicleDataSubStd triggers 751 NOTIFY messages, and the VehicleDataSubEnh triggers 11 NOTIFY messages, which leads to the corresponding number of contentInstance resources in the VehicleDataFilteredStd and VehicleDataFilteredEnh data containers. This results in savings of 98,53528 % for the enhanced data exchange capabilities as utilised by the VehicleDataSubEnh (see 3.3).

Figure 7.10 illustrates the number of notifications for both subscriptions during decrease of fuelRange from 750 KM to 0 KM according to the conditions of this Experiment 3.

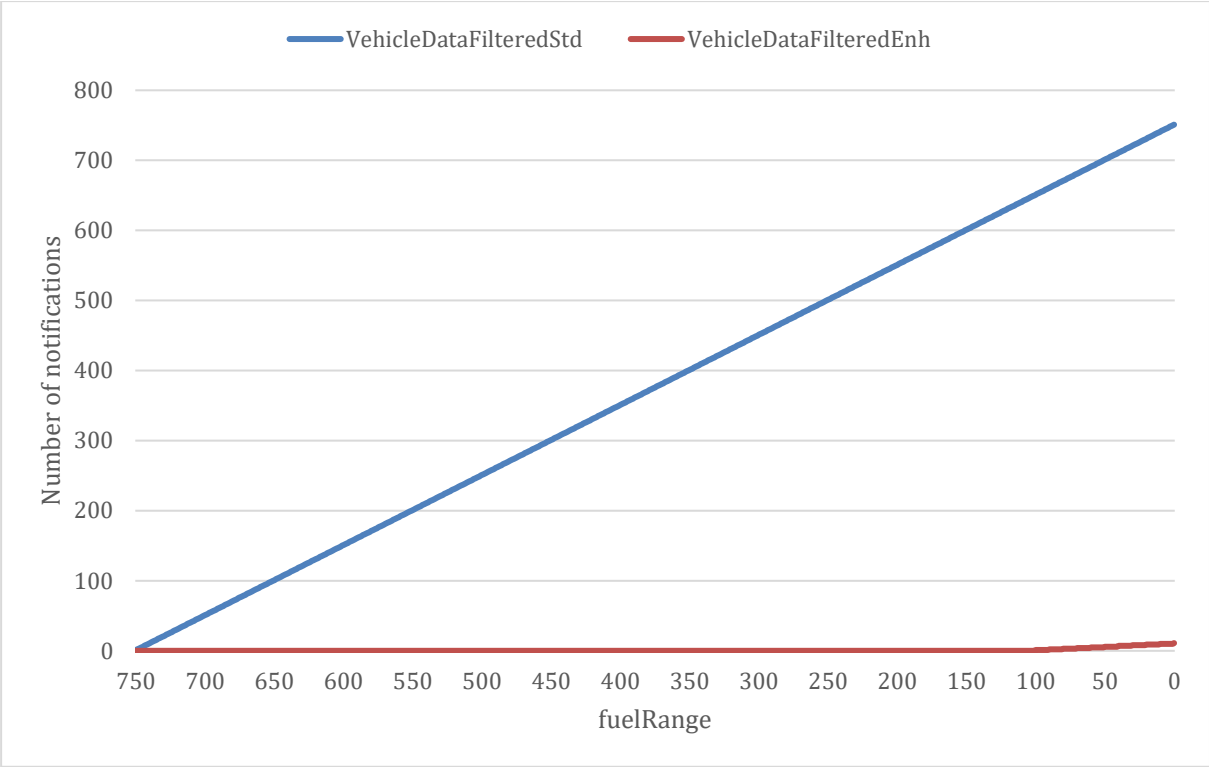

Figure 7.10: Comparison of the number of notifications for VehicleDataSubStd and VehicleDataSubEnh during fuelRange decrease according to the conditions of Experiment 3

7.4 Concluding Considerations

The focus of this research is vehicle-to-backend platform architectures (see Section 1.1). Related distributed functionalities, scenarios, and applications are introduced to derive common architecturally significant requirements that should be addressed by platform capabilities. In this regard, the scenarios and distributed functionalities do not claim to become similarly realised in real-world deployments, but they are selected as relevant considerations and abstractions for the functionalities associated with connected vehicles.

This prototypical implementation of the novel data exchange enhancements within the OM2M prototype confirmed the applicability of the proposed approaches for the oneM2M CSE (see Section 6.2). Moreover, the only minimal necessary enhancements of the reference points, and the suitable integratability of the enhancements within the CSE, support the assessment about the appropriateness of the selected architectural alternative (see Section 6.3). Hence, this prototype successfully provides a proof of the concept.

Network Efficiency and Performance

The identified shortcomings of the current oneM2M data exchange capabilities through subscribe/notify mechanism constrain the filtering capabilities at the CSE level (see Section 5.4). As a consequence, AEs have limited capabilities to tailor the data acquisition according to the requirements of the use case. Since this in general leads to the transmission of more data

than actually required, this results in reduced network efficiency (see Section 5.4). In this regard, the existing limitations constitute a shortcoming with respect to the capabilities offered by a oneM2M-based distributed ASDP to prevent network misalignments (cf. REQ 6, Section 3.4.3).

The three experiments aimed to use a realistic scenario derived from the condensed ASDP scenario. The results showed savings of from 85,5% up to 98,5% with regard to the number of NOTIFY messages through the constant decrease of the remaining fuelRange from 750 KM to 0 KM. This demonstrates the potential of the proposed novel data exchange capabilities for the oneM2M service platform. However, the quantities strongly depend on the considered scenarios including data sources and sinks and aspects such as design of the related distributed AEs and data containers. In this regard, also the novel enhancements and improvements by means of increased platform capabilities have to be considered qualitatively as well: They enable the increase of the correlation of theoretical network requirements of a distributed functionality with its actual network requirements (cf. initial considerations about filter requirements versus capabilities in Section 5.2.1). Besides, the proposed enhancements extend the existing data exchange capabilities which in themselves remain fully available. As a result, the suitable usage of the enhancements cannot result in worse network efficiency. Thus, the novel enhancements fully contribute to the prevention of network misalignments (cf. REQ 6, Section 3.4.3) through increased CSE capabilities.

Nevertheless, adding functionalities to the CSE in general raises the computational and memory requirements of the implementation. This likewise applies to the introduced Esper CEP component (see Section 7.1.2) and related building blocks to implement the proposed enhanced data exchange capabilities (see Section 7.2). Furthermore, the enhancements may degrade the latency of the respective methods such as the creation of contentInstance resources, the creation, update, and deletion of subscription resources and the preparation of the notification messages (see Section 7.2). Furthermore, in this regard, the proposed architectural enhancements, as most architectural decisions constitute a trade-off decision (see Section 2.4.4). However, the enhanced data exchange capabilities widen the design space for distributed AEs but remain optional. Hence, despite an increased memory footprint of the CSE, data exchange is basically still possible with the performance of the unmodified oneM2M CSE. Furthermore, the creation of subscriptions can be rejected by the receiving CSE (oneM2M TS-0004, 2015). This might be appropriate when a subscription exceeds the (remaining) computational capabilities of the node, e.g., because there already exist too many subscriptions or because of too complex notification criteria (i.e., EPL statements), the evaluation of which requires too much performance. Nevertheless, this is basically not limited to subscriptions utilising the proposed enhancements, but also could occur with existing subscriptions.

However, the general applicability of CEP technologies respectively the Esper CEP Engine in the context of constraint embedded devices has been shown. Saleh et al. in (2013) have utilised

CEP in the context of Wireless Sensor Networks (WSN) to perform "in-network complex event processing". Their work showed that even sensors with very limited computational power, limited memory and battery power are capable of performing CEP queries. Moreover, they also showed the benefits of CEP with respect to a reduced number of messages (i.e., increased network efficiency of the distributed functionalities). The reduced number of messages even increased the energy efficiency of the sensor, although the processing of the CEP (sub-)queries on the sensor increase computational efforts (Saleh, 2013). This matches with the performance-related information about the Esper CEP Engine (EsperTech, 2016b).

Considering these results as well as the current and planned future functionalities of the oneM2M standard, the proposed enhancements can generally be assessed as machine-capable. Finally, the E2E approach of oneM2M facilitates the deployment of the CSE on a variety of machines, which besides constraint machines also includes more powerful MNs and INs (see Section 4.3.6). It can be assumed that these MNs and INs which are located further up in the hierarchical network are less constraint and at the same time have to deal with an increased amount of data. In this regard, this also supports the proposed enhancements and approach. Not of least importance, vehicles (respectively the related On-Board Unit) can be considered as rather powerful machines, which is the reason why the proposed enhancements are particularly suitable for the realisation of a oneM2M-based ASDP.

7.5 Summary

The applicability of the proposed enhancements for data exchange by means of the subscribe/notify mechanisms of the oneM2M service platform was proven by a prototypical implementation. With respect to the selected approach and architectural alternative, the enhancements have been integrated into the Eclipse OM2M project, which is extended by the Esper CEP Engine. In more detail, the integration of Content Decoder, Esper Event Adaptor, Subscription and Notification Criteria Aggregator, and the Esper Statement Adaptor constitute the building blocks to implement the proposed enhancements. Experiments, derived from the condensed ASDP scenario, showed significant improvements with regard to the network efficiency of distributed functionalities. Finally, these and further performance aspects are elaborated upon as part of concluding considerations.

8 Conclusions and Future Work

It is particularly software in combination with Internet connectivity (e.g., by use of wireless cellular networks), which has already kicked off the drive leading to significant change of vehicular functionalities and capabilities. The connected vehicle will offer enhanced In-Vehicle Infotainment systems (IVI), Advanced Driver Assistant Systems (ADAS) up to Highly Automated Driving (HAD) to its driver and passengers. Moreover, the connected vehicle will become an integrated part of an Intelligent Transportation System (ITS), which will facilitate increased traffic safety, traffic efficiency, and comfort.

These functionalities with heterogeneous or even contradictory requirements already make up today's cars complex software-intensive embedded systems, which is one aspect why the integration of these functionalities is a very challenging task. It is becoming even more challenging with respect to the implementation of envisaged future functionalities within the functional domains named against the background of further objectives such as increased time-to-market, customisability, and maintainability during the entire vehicle lifecycle. Furthermore, interoperability across vehicle series, Original Equipment Manufacturers (OEMs), or even with other industries or sectors beyond automotive (e.g., energy, or healthcare) is required so that the visions of an ITS can materialise. The key to building such future automotive systems that are capable of dealing with these manifold requirements are adequate software architectures and platforms which are not vendor-specific. While AUTOSAR introduced such an architectural approach for in-vehicle Electronic Control Units (ECUs) and related functionalities, others (e.g., the GENIVI alliance) are focussing on IVI; a common approach for standardised and universal vehicle-to-backend platform architecture is currently missing.

8.1 Achievements of the Research

This research has its foundation in the automotive domain and its emerging functionalities in the context of connected vehicles. The latter leads to new domains and technologies becoming relevant for automotive development. Due to this regard, this research provides a holistic and interdisciplinary view on connected vehicles at the interface of automotive, software, and telecommunication. For that reason, this research starts with a comprehensive literature research on the characteristics of the automotive domain and current challenges within automotive software development against the background of future functionalities, originating

from the functional domains IVI, ADAS, and ITS. Connectivity, in this regard, on one hand introduces new challenges but on the other hand provides new capabilities to deal with the limitations of the "traditional approach" of integrating functionalities through true installation into the vehicular ECUs. Considering this, the importance of the software and system architecture is pointed out as a key to successfully deal with these challenges. With respect to related frameworks and platforms, the necessity of a common vehicle-to-backend platform was identified.

As a solution, this research has been developed and proposed the novel concept of a distributed Automotive Service Delivery Platform (ASDP). It facilitates identified valuable principles such as the integration or offloading of vehicular functionalities at/to an OEM server in the backend as well as its usage as main hub for the integration of third parties. Related to these principles and as a further aspect for the development of an ASDP, criteria have been identified to qualitatively assess automotive functionalities which facilitates inferences regarding their suitable decomposition and distribution between the vehicle and the backend. These criteria can also be used beyond this research as a basis to assess distributed functionalities of a connected vehicle and, e.g., to derive different deployment or distribution strategies. Finally, to reflect the range and interdisciplinary of future connected vehicles with regard to the functionalities and challenges, this research has been utilised three different viewpoints to derive six architecturally significant requirements to an ASDP enabling platform. These viewpoints consider previously described principles and selected scenarios as well as the distribution considerations which have been performed to further examine the problem space of an ASDP.

Similarities with regard to the problem space motivate the consideration of the oneM2M service platform which is currently being developed and standardised, as an enabler for the implementation of an ASDP. However, conclusions about the appropriateness of the oneM2M service platform should be based on its provided solution space. For that reason, the architectural building blocks and central architectural design decisions of the oneM2M standard have been analysed in detail. Although shortcomings were identified, such as the lack of full semantic interoperability, the general suitability of oneM2M as enabler for the ASDP has been shown. Consequently, a reference configuration of the oneM2M service platform for the ASDP, respectively automotive environments, is described.

Nevertheless, the general analysis of the oneM2M standard also identified the necessity to a comprehensive investigation of the data exchange capabilities provided by the oneM2M service platform. After principle considerations about filter requirements vs. capabilities and filter positions with regard to distributed functionalities, a condensed ASDP scenario has been used for profound scenario-based analysis. It unveiled shortcomings of current data exchange with subscribe/notify mechanism, which in typical usage scenarios results in decreased network efficiency and privacy. To overcome these shortcomings, novel enhancements of the oneM2M

standard, such as application-data-dependent notification criteria for subscribe/notify mechanism and aggregation of subscriptions, are proposed. The approach proposes in particular the utilisation of Complex Event Processing to facilitate specification and evaluation of enhanced notification criteria. In advance of a prototypical implementation as proof-of-concept, the proposed enhancements are assessed with respect to the design philosophy of current oneM2M standard, anticipated future enhancements, and possible architectural alternatives considering also the Platform-Based Design methodology.

Finally, a prototypical implementation of the proposed enhancements demonstrated their feasibility and the appropriateness of the selected architectural design decisions. Experiments and estimations with a typical vehicle maintenance scenario where a backend application subscribes to a low remaining fuel range of the vehicle to initiate further activities showed significant savings of the number of notification messages sent compared to the oneM2M service platform without the proposed enhancements: Depending on the assumed scope conditions and the exact subscriptions applied, the savings are 85,5% up to 98,5% for this scenario. These network efficiency improvements as well as further performance aspects of the enhancements have been put into context within concluding considerations.

This research has developed a novel oneM2M-based Automotive Service Delivery Platform. With the introduced enhanced data exchange capabilities for the oneM2M service platform, this oneM2M-based ASDP offers a universal vehicle-to-backend platform solution that is capable of addressing the key requirements of automotive environments. In this regard, this research provides significant contributions to the field of enhanced automotive software and system architectures that will enable the functionalities and visions related to future connected vehicles to become true.

While the enhanced data exchange capabilities for the oneM2M service platform are motivated by distributed automotive functionalities, their benefits are not limited to the ASDP or automotive environments. Particularly whenever applications require only a small but specific subset of an existing high amount of sensor and application data, the enhancements are beneficial. In this regard, the contributions made to the oneM2M standard are also profitable for many other M2M scenarios and domains.

8.2 Limitations

Although the objectives of this research have been met, limitations can be identified. These basically are owing to practical reasons with regard to differentiation to on-going related research and standardisation activities, or due to given time scope for the research project. The key limitations are summarised below.

The proposed enhancements of the oneM2M service platform for enhanced data exchange aimed at most compatibility with the current version of the oneM2M standard. As a result, the

capabilities of the CEP and EPL-based notification criteria are not fully exploited. For example: The EPL statements would also enable the detailed specification about which data of a content resource shall be selected and hence would facilitate individual content of the notification message. This possibility is not utilised in this research, since the notification message remains standard-conformant and hence, e.g., includes a full contentInstance resource. Thus, the usage of the EPL statements is limited to the detection of the fulfilment of application-data-dependent notification criteria, i.e. as trigger.

The aggregation of EPL-based notification criteria among themselves and with existing notification criteria of oneM2M have not been investigated in depth by means of providing the mathematical foundation for their aggregation. However, with the selection of the comprehensive EPL, which is based on SQL, the foundation for comprehensive aggregation mathematics has been provided.

Although the massive deployment of connected vehicles is reflected within the considered scenarios and hence within the ASDP concept, the considerations of the oneM2M service platform capabilities are limited to the fundamentals. This means, e.g., the platform capabilities and building blocks are considered with one vehicle-ASN and one OEM-IN, but deployment scenarios with thousands of vehicles have not been considered. Accordingly, no considerations about regional distributions or multi-node query optimisations of EPL-based notification criteria for data exchange have been made.

8.3 Suggestions for Future Work

This research project has made valid contributions to the knowledge for future software and system architectures in the context of connected vehicles, and the applicability of oneM2M-based vehicle-to-backend platforms as enabler for such ASDP concept. However, a number of areas for future work can be identified. These suggestions are detailed below.

1. This research introduced CEP as basis for enhanced data exchange capabilities. Having such CEP technology and related EPL as comprehensive policy language available could be used for further improvements of the oneM2M service platform capabilities and mechanisms, such as:
    a. EPL-based policies can provide the foundation for more advanced accessControlPolicies for resource access: Currently, the oneM2M standard only supports black or white configuration, as to whether a certain method is allowed to perform on a resource. EPL-based accessControlPolicies could facilitate more qualified access constraints, and that, moreover, can contribute to privacy considerations (cf. He, Barman, Di Wang, & Naughton, 2011; Olumofin & Goldberg, 2010).

   b. Existing notification criteria for subscribe/notify mechanism can be completely substituted by EPL-based notification criteria, in continuation of proposed enhancements of this research.
   c. Considerations about deployment strategies of EPL-based notification criteria can be intensified. This may include multi-query optimisation of distributed CEP (Cugola & Margara, 2012). It may also include inference about geographical distribution of nodes and configurations, e.g., including Mobile Edge Computing considerations (Beck, Feld, Linnhoff-Popien, & Pützschler, 2016; cf. Y. C. Hu et al., 2015), etc.
2. With regard to the already-envisaged enhancements of the oneM2M service platform towards the enabling of full semantic interoperability (oneM2M TR-0007, 2015), the existing subscribe/notify (i.e., publish/subscribe) mechanism of the oneM2M service platform could be enhanced accordingly. The concept-based publish/subscribe approach, introduced by Cilia et al., may provide a starting point for beneficial improvements here (Cilia, Antollini, & Bornhövd, 2004).
3. The interplay of a oneM2M-based ASDP with the legacy in-vehicle components of the automotive software and system landscape could be investigated in more detail. This may include considerations about interworking in contrast to integration (oneM2M TS-0001, 2015, p. 313). Furthermore, concrete bindings to other standards could be developed, such as the realisation of the vehicle-ASN as AUTOSAR SW-C or complex device driver, or the utilisation of SOME/IP for the integration (or interworking) of AUTOSAR-conformant functionalities with a vehicle-ASN.

Bibliography

1. Abdalla, G. M., Abu-Rgheff, M. A., & Senouci, S. M. (2007). Current Trends in Vehicular Ad Hoc Networks. *Ubiquitous Computing and Communication Journal*, 1–9.

2. Accenture. (2012). *Perspectives on In-Vehicle Infotainment Systems and Telematics*. *accenture.com*.

3. ADASIS. (2017). Advancing map-enhanced driver assistance systems. Retrieved September 10, 2017, from http://adasis.org/

4. AIOTI. (2017). High Level Architecture (HLA) (3rd ed., pp. 1–40). AIOTI - Alliance for Internet of Things Innovation.

5. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. *IEEE Communications Surveys & Tutorials*, 1–33. http://doi.org/10.1109/COMST.2015.2444095

6. Alasti, M., Neekzad, B., Hui, J., & Vannithamby, R. (2010). Quality of service in WiMAX and LTE networks. *IEEE Communications Magazine*, *48*(5), 104–111. http://doi.org/10.1109/MCOM.2010.5458370

7. Alaya, M. B., Banouar, Y., Monteil, T., Chassot, C., & Drira, K. (2014). OM2M: Extensible ETSI-compliant M2M Service Platform with Self-configuration Capability. *Procedia Computer Science*, *32*, 1079–1086. http://doi.org/10.1016/j.procs.2014.05.536

8. Aldred, L., van der Aalst, W. M. P., Dumas, M., & Hofstede, ter, A. H. M. (2005). On the Notion of Coupling in Communication Middleware. In M. Akşit & S. Matsuoka (Eds.), *Formal Methods for Software Architectures* (Vol. 3761, pp. 1015–1033). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/11575801_6

9. Alliance, T. O. (2014). OSGi Core (6 ed., pp. 1–450).

10. Alljoyn. (2017). OCF - AllJoyn. Retrieved September 9, 2017, from https://openconnectivity.org/developer/reference-implementation/alljoyn

11. Alt, O. (2009). Car Multimedia Systeme Modell-basiert testen mit SysML. Wiesbaden: Vieweg+Teubner. http://doi.org/10.1007/978-3-8348-9567-7

12. Amazon Web Services. (2017). Übersicht über AWS IoT Core – Amazon Web Services. Retrieved September 7, 2017, from //aws.amazon.com/de/iot-core/

13. Android Auto. (2016). Android Auto. Retrieved December 6, 2016, from https://www.android.com/intl/en_en/auto/

14. Apple. (2016). iOS - CarPlay. Retrieved July 1, 2016, from http://www.apple.com/de/ios/carplay/

15. ARIB. (2017). Association of Radio Industries and Businesses. Retrieved September 10, 2017, from https://www.arib.or.jp/english/

16. ATIS. (2017). ATIS - The Alliance for Telecommunications Industry Solutions. Retrieved September 10, 2017, from http://www.atis.org/

17. Audi AG. (2016a). Audi connect > Audi Deutschland. Retrieved December 1, 2016, from https://www.audi.de/de/brand/de/neuwagen/layer/audi-connect-lp.html

18. Audi AG. (2016b). Everything combined, all in one place: The central driver assistance control unit. Retrieved August 24, 2016, from http://www.audi.com/com/brand/en/vorsprung_durch_technik/content/2014/10/zentraleS-fahrerassistenzsteuergeraet-zfas.html

19. AUTOSAR. (2013). AUTomotive Open System ARchitecture. Retrieved April 10, 2013, from http://www.autosar.org/

20. AUTOSAR. (2014). *Project Objectives* (4.2.1). AUTOSAR.

21. AUTOSAR. (2015a). *Layered Software Architecture* (4.2.2). AUTOSAR.

22. AUTOSAR. (2015b). *Software Component Template* (4.2.2). AUTOSAR.

23. Basole, R. C., & Karla, J. (2011). On the Evolution of Mobile Platform Ecosystem Structure and Strategy. *Wirtschaftsinformatik*, *53*(5), 301–311. http://doi.org/10.1007/s11576-011-0286-y

24. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice, 3rd edition. *Software Architecture in Practice, 3rd Edition*.

25. Bassi, A., Bauer, M., Fiedler, M., Kramp, T., Kranenburg, R., Lange, S., & Meissner, S. (2013). Enabling Things to Talk. (A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. van Kranenburg, S. Lange, & S. Meissner, Eds.). Berlin, Heidelberg: Springer. http://doi.org/10.1007/978-3-642-40403-0

26. Bauer, M. (2017). IoT Platforms for Smart Cities (pp. 1–51). Presented at the 22. VDE/ITG Fachtagung Mobilkommunikation. Retrieved from https://www.hs-osnabrueck.de/fileadmin/HSOS/Forschung/Recherche/Laboreinrichtungen_und_Versuchsbetriebe/Labor_fuer_Hochfrequenztechnik_und_Mobilkommunikation/Mobilkomtagung/2017/Vortraege/4_Martin_Bauer.pdf

27. Bauer, S. (2010). Das vernetzte Fahrzeug – Herausforderungen für die IT. *Informatik-Spektrum*, *34*(1), 38–41. http://doi.org/10.1007/s00287-010-0504-9

28. Baun, C., Kunze, M., Nimis, J., & Tai, S. (2009). Cloud Computing. Springer-Verlag. http://doi.org/10.1007/978-3-642-01594-6

29. Bechler, M., Berninger, H., Biehle, T., Bohnert, T. M., Bossom, R., Brignolo, R., et al. (2010). European ITS Communication Architecture. COMeSafety.

30. Beck, M. T., Feld, S., Linnhoff-Popien, C., & Pützschler, U. (2016). Mobile Edge Computing. *Informatik-Spektrum*, *39*(2), 108–114. http://doi.org/10.1007/s00287-016-0957-6

31. Beck, M. T., Werner, M., Feld, S., & Schimper, S. (2014). Mobile edge computing: A taxonomy. http://doi.org/10.1.1.670.9418

32. Ben Alaya, M., Medjiah, S., Monteil, T., & Drira, K. (2015). Toward semantic interoperability in oneM2M architecture. *IEEE Communications Magazine*, *53*(12), 35–41. http://doi.org/10.1109/MCOM.2015.7355582

33. Ben Alaya, M., Monteil, T., & Drira, K. (2014). The importance of the collection pattern for OneM2M architecture (pp. 1–19).

34. Bhalla, M. R., & Bhalla, A. V. (2010). Generations of mobile wireless technology: A survey. *International Journal of Computer Applications*, *5*(4). Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.5216&rep=rep1&type=pdf

35. Blair, G. S., Paolucci, M., Grace, P., & Georgantas, N. (2011). Interoperability in Complex Distributed Systems. In M. Akşit & S. Matsuoka (Eds.), *Formal Methods for Software Architectures* (Vol. 6659, pp. 1–26). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-21455-4_1

36. Blaupunkt. (2017). BLAUPUNKT: Success Story. Retrieved September 8, 2017, from http://www.blaupunkt.com/en/company/success-story/

37. BMW ConnectedDrive. (2016). BMW ConnectedDrive Online Guide. Retrieved April 17, 2016, from http://features.bmw.connecteddrive.info/en.html

38. Booysen, M. J., Gilmore, J. S., Zeadally, S., & van Rooyen, G.-J. (2012). Machine-to-Machine (M2M) Communications in Vehicular Networks. Korea Society of Internet Information (KSII). http://doi.org/10.3837/tiss.2012.02.005

39. Bormann, C., Castellani, A. P., & Shelby, Z. (2012). CoAP: An Application Protocol for Billions of Tiny Internet Nodes. *IEEE Internet Computing*, *16*(2), 62–67. http://doi.org/10.1109/MIC.2012.29

40. Bose, R., Brakensiek, J., & Park, K.-Y. (2010). Terminal mode: transforming mobile devices into automotive application platforms (pp. 148–155). Presented at the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications, New York, New York, USA: ACM. http://doi.org/10.1145/1969773.1969801

41. Bose, R., Brakensiek, J., Park, K.-Y., & Lester, J. (2011). Morphing Smartphones into Automotive Application Platforms. *IEEE Computer*, *44*(5), 53–61. http://doi.org/10.1109/MC.2011.126

42. Boswarthick, D., Elloumi, O., & Hersent, O. (2012). M2M Communications. John Wiley & Sons.

43. Broadband Forum. (2013). *TR-069* (Issue: 1 Amendment 5, V1.4) (pp. 1–228). Broadband Forum.

44. Broadband Forum. (2017). Broadband forum. Retrieved September 10, 2017, from https://www.broadband-forum.org/

45. Brownsword, L. (2004). Current Perspectives on Interoperability.

46. Broy, M. (2006a). Challenges in automotive software engineering (pp. 33–42). Presented at the 29th IEEE International Conference on Distributed Computing Systems.

47. Broy, M. (2006b). The “Grand Challenge” in Informatics: Engineering Software-Intensive Systems. *IEEE Computer*, *39*(10), 72–80. http://doi.org/10.1109/MC.2006.358

48. Broy, M., Kruger, I. H., Pretschner, A., & Salzmann, C. (2007). Engineering Automotive Software. *Proceedings of the IEEE*, *95*(2), 356–373. http://doi.org/10.1109/JPROC.2006.888386

49. Broy, M., Reichart, G., & Rothhardt, L. (2011). Architekturen softwarebasierter Funktionen im Fahrzeug: von den Anforderungen zur Umsetzung. *Informatik-Spektrum*, *34*(1), 42–59. http://doi.org/10.1007/s00287-010-0507-6

50. Bruns, R., & Dunkel, J. (2015). Complex Event Processing. Wiesbaden: Springer-Verlag. http://doi.org/10.1007/978-3-658-09899-5

51. Bruns, R., Dunkel, J., Masbruch, H., & Stipkovic, S. (2015). Intelligent M2M: Complex event processing for machine-to-machine communication. *Expert Systems with Applications*, *42*(3), 1235–1246. http://doi.org/10.1016/j.eswa.2014.09.005

52. Burns, A., & Davis, R. (2013). *Mixed criticality systems-a review*.

53. Buschmann, F., Henney, K., & Schmidt, D. C. (2007). Pattern-Oriented Software Architecture, A Pattern Language for Distributed Computing. John Wiley & Sons.

54. Cacilo, A., Schmidt, S., Wittlinger, P., Herrmann, F., Sawade, O., Doderer, H., et al. (2016). *Hochautomatisiertes Fahren auf Autobahnen – industriepolitische Schlussfolgerungen* (Dienstleistungsprojekt 15/14). Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO.

55. Car Connectivity Consortium. (2016). MirrorLink. Retrieved July 1, 2016, from http://www.mirrorlink.com/

56. Carloni, L. P., De Bernardinis, F., Pinello, C., Sangiovanni-Vincentelli, A. L., & Sgroi, M. (2005). Platform-Based Design for Embedded Systems. *Embedded Systems Handbook 2005*, *6*, 22–1–22–26. http://doi.org/10.1201/9781420038163.ch22

57. Carney, D., Fisher, D., Morris, E., & Place, P. (2005). *Some current approaches to interoperability. Integration of software intensive systems initiative* (CMU/SEI-2005-TN-033). Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

58. CCSA. (2017). China Communictions Standards Association. Retrieved September 10, 2017, from http://www.ccsa.org.cn/english/

59. CEN. (2017). European Committee for Standardization. Retrieved September 10, 2017, from https://www.cen.eu/Pages/default.aspx

60. CENELEC. (2017). Welcome to CENELEC – European Committee for Electrotechnical Standardization. Retrieved September 10, 2017, from https://www.cenelec.eu/

61. Chandy, K. M., & Schulte, W. R. (2010). Event Processing - Designing IT Systems for Agile Companies.

62. Charette, R. N. (2009). This car runs on code. *IEEE Spectrum*.

63. Cilia, M., Antollini, M., & Bornhövd, C. (2004). Dealing with heterogeneous data in pub/sub systems: The Concept-Based approach. *3rd Int'l Workshop on Distributed Event-Based Systems*.

64. Colace, F., De Santo, M., Moscato, V., Picariello, A., Schreiber, F. A., & Tanca, L. (2015). Data Management in Pervasive Systems. Springer. http://doi.org/10.1007/978-3-319-20062-0

65. COMAND Online. (2016). Mercedes-Benz TechCenter: COMAND Online. Retrieved April 17, 2016, from http://techcenter.mercedes-benz.com/_en/comand_online/detail.html

66. COMeSafety. (2013). COMeSafety 2 Project. Retrieved April 8, 2013, from http://www.comesafety.org

67. Coulouris, G. F., Dollimore, J., & Kindberg, T. (2012). Distributed Systems (Fifth Edition). Pearson Education.

68. Cugola, G., & Margara, A. (2012). Deployment strategies for distributed complex event processing. *Computing*, *95*(2), 129–156. http://doi.org/10.1007/s00607-012-0217-9

69. Daniel, F., & Matera, M. (2014). Mashups. Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-55049-2

70. Dar, K., Bakhouya, M., Gaber, J., Wack, M., & Lorenz, P. (2010). Wireless communication technologies for ITS applications. *IEEE Communications Magazine*, *48*(5), 156–162. http://doi.org/10.1109/MCOM.2010.5458377

71. Davis Paul, K., & Anderson Robert, H. (2003). *Improving the Composability of Department of Defense Models and Simulations*. Santa Monica CA.

72. Dodig-Crnkovic, G. (2002). Scientific methods in computer science. Presented at the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden.

73. Domingue, J., Fensel, D., & Hendler, J. A. (2011). Handbook of Semantic Web Technologies. Springer Science & Business Media. http://doi.org/10.1007/978-3-540-92913-0

74. Eckert, M., & Bry, F. (2009). Complex Event Processing (CEP). *Informatik-Spektrum*, 1–8.

75. Eclipse OM2M project. (2016). Eclipse OM2M - Open Source platform for M2M communication. Retrieved July 21, 2016, from http://www.eclipse.org/om2m/

76. Ekström, H. (2009). QoS control in the 3GPP evolved packet system. *IEEE Communications Magazine*, *47*(2), 76–83. http://doi.org/10.1109/MCOM.2009.4785383

77. Elloumi, O. (2014). oneM2M architecture principles and benefits (pp. 1–24). Presented at the KuVS NGSDP Expert Talk, Berlin, Germany.

78. Erl, T., Carlyle, B., Pautasso, C., & Balasubramanian, R. (2013). SOA with REST - Principles, Patterns and Constraints for Building Enterprise Solutions with REST (pp. 1–577). Prentice Hall.

79. Eskandarian, A. (2012). Handbook of Intelligent Vehicles. (A. Eskandarian, Ed.). London: Springer London. http://doi.org/10.1007/978-0-85729-085-4

80. EsperTech. (2016a). Esper - EsperTech. Retrieved July 21, 2016, from http://www.espertech.com/esper/

81. EsperTech. (2016b). EsperTech - How does Esper scale? Retrieved July 21, 2016, from http://www.espertech.com/esper/esper-faq/#scaling

82. ETSI (2017). ETSI - Welcome to the World of Standards! Retrieved September 10, 2017, from http://www.etsi.org/

83. ETSI TR 102 638. (2009). *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions* (V1.1.1). European Telecommunications Standards Institute (ETSI).

84. ETSI TR 102 898. (2013). *Machine to Machine communications (M2M); Use cases of Automotive Applications in M2M capable networks* (V1.1.1). European Telecommunications Standards Institute (ETSI).

85. ETSI TS 102 689. (2013). *Machine-to-Machine communications (M2M); M2M service requirements* (1st ed.) (pp. 1–34). European Telecommunications Standards Institute (ETSI).

86. ETSI TS 102 690. (2013). *Machine-to-Machine communications (M2M); Functional architecture* (2nd ed.) (pp. 1–332). European Telecommunications Standards Institute (ETSI).

87. ETSI TS 102 921. (2013). *Machine-to-Machine communications (M2M); mIa, dIa and mId interfaces* (2nd ed.) (pp. 1–618). European Telecommunications Standards Institute (ETSI).

88. Eugster, P. T., Felber, P., Guerraoui, R., & Kermarrec, A.-M. (2003). The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, *35*(2), 114–131. http://doi.org/10.1145/857076.857078

89. European Parliament, Council of the European Union. *Regulation (EU) 2015/758 of the European Parliament and of the Council of 29 April 2015 concerning type-approval requirements for the deployment of the eCall in-vehicle system based on the 112 service and amending Directive 2007/46/EC*, eur-lex.europa.eu.

90. European Research Cluster on Internet of Things (IERC). (2017). European Research Cluster on the Internet of Things. Retrieved September 8, 2017, from http://www.internet-of-things-research.eu/about_iot.htm

91. Faezipour, M., Nourani, M., Saeed, A., & Addepalli, S. (2012). Progress and challenges in intelligent vehicle area networks. *Communications of the ACM*, *55*(2). http://doi.org/10.1145/2076450.2076470

92. Farcas, C., Farcas, E., Krüger, I. H., & Menarini, M. (2010). Addressing the Integration Challenge for Avionics and Automotive Systems - From Components to Rich Services. *Proceedings of the IEEE*, *98*(4), 562–583. http://doi.org/10.1109/JPROC.2009.2039630

93. Federal Ministry of Justice and Consumer Protection. Federal Data Protection Act. (Translations provided by the Language Service of the Federal Ministry of the Interior, Trans.) (2009).

94. Festag, A. (2014). Cooperative intelligent transport systems standards in europe. *IEEE Communications Magazine*, *52*(12), 166–172. http://doi.org/10.1109/MCOM.2014.6979970

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

96. Fisher, D. A. (2006). *An Emergent Perspective on Interoperation in Systems of Systems* (CMU/SEI-2006-TR-003) (pp. 1–67). Carnegie Mellon Software Engineering Institute.

97. FIWARE. (2017). FIWARE | Open APIs for Open Minds. Retrieved September 7, 2017, from https://www.fiware.org/

98. Freed, N., Klensin, J., & Hansen, T. (2013). Media Type Specifications and Registration Procedures (p. 32). Internet Engineering Task Force (IETF).

99. Fuhrmann, W. F., & Brass, V. (1994). Performance aspects of the GSM radio subsystem (Vol. 82, pp. 1449–1466). Presented at the Proceedings of the IEEE. http://doi.org/10.1109/5.317088

100. Fürst, S., & Bunzel, S. (2015). AUTOSAR. In H. Winner, S. Hakuli, F. Lotz, & C. Singer (Eds.), *Handbuch Fahrerassistenzsysteme* (pp. 105–122). Wiesbaden: Springer Fachmedien Wiesbaden. http://doi.org/10.1007/978-3-658-05734-3_7

101. Gajski, D. D., Abdi, S., Gerstlauer, A., & Schirner, G. (2009). Embedded System Design. Springer Science & Business Media. http://doi.org/10.1007/978-1-4419-0504-8

102. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns. Addison-Wesley.

103. GENIVI. (2013). GENIVI Alliance. Retrieved April 16, 2013, from http://www.genivi.org/

104. Gieraths, A. (2014). Umsetzung der Anforderungen aus der ISO 26262 bei der Entwicklung eines Steuergeräts aus dem Fahrerinformationsbereich. *Automotive - Safety & Security*.

105. Glass, R. L., Ramesh, V., & Vessey, I. (2004). An analysis of research in computing disciplines. *Communications of the ACM*, *47*(6), 89–94. http://doi.org/10.1145/990680.990686

106. GlobalPlatform. (2017). GlobalPlatform. Retrieved September 10, 2017, from https://www.globalplatform.org/

107. Gryc, A. (2011, February). Making Sense of the Smartphone-Vehicle Cacophony. Retrieved April 10, 2013, from http://www.qnx.com/download/download/21914/qnx_auto_smart_phone.pdf

108. Gryc, A., & Johnson, K. (2011). *Why Automakers (Should) Care about HTML5* (pp. 1–7). QNX Software Systems. Retrieved from http://www.qnx.de/download/download/22989/qnx_auto_html5.pdf

109. Hardung, B., Kölzow, T., & Krüger, A. (2004). Reuse of software in distributed embedded automotive systems (pp. 203–210). Presented at the 4th ACM international conference on Embedded software, New York, NY, USA: ACM. http://doi.org/10.1145/1017753.1017787

110. Harrison, N. B., & Avgeriou, P. (2010). How Do Architecture Patterns and Tactics Interact? A Model and Annotation. *The Journal of Systems & Software*, *83*(10), 1735–1758. http://doi.org/10.1016/j.jss.2010.04.067

111. He, Y., Barman, S., Di Wang, & Naughton, J. F. (2011). On the complexity of privacy-preserving complex event processing (pp. 165–174). Presented at the PODS '11: Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, New York, New York, USA: ACM. http://doi.org/10.1145/1989284.1989304

112. Holle, J., Groll, A., Ruland, C., Cankaya, H., & Wolf, M. (2011). Open Platforms on the Way to Automotive Practice. *8th ITS European Congress*.

113. Horowitz, B., Liebman, J., Ma, C., Koo, T. J., Sangiovanni-Vincentelli, A., & Sastry, S. S. (2003). Platform-based embedded software design and system integration for autonomous vehicles. *Proceedings of the IEEE*, *91*(1), 198–211. http://doi.org/10.1109/JPROC.2002.805827

114. Hoymann, C., Astely, D., Stattin, M., Wikstrom, G., Cheng, J.-F., Hoglund, A., et al. (2016). LTE release 14 outlook. *IEEE Communications Magazine*, *54*(6), 44–49. http://doi.org/10.1109/MCOM.2016.7497765

115. Höhn, R., & Höppner, S. (2008). Das V-Modell XT. Springer-Verlag. http://doi.org/10.1007/978-3-540-30250-6

116. Hu, R. Q., Qian, Y., Chen, H.-H., & Jamalipour, A. (2011). Recent progress in machine-to-machine communications [Guest editorial]. *IEEE Communications Magazine*, *49*(4), 24–26. http://doi.org/10.1109/MCOM.2011.5741142

117. Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., & Young, V. (2015). Mobile Edge Computing - A Key Technology Towards 5G. *ETSI White Paper*.

118. Huber, W., Lädke, M., & Ogger, R. (1999). Extended floating-car data for the acquisition of traffic information. *6th World Congress on Intelligent Transportation Systems (ITS 1999)*.

119. IEEE Computer Society. (2007). *Systems and software engineering — Recommended practice for architectural description of software-intensive systems* (No. ISO/IEC 42010:2007(E) IEEE Std 1471-2000) (pp. c1–24). Piscataway, NJ, USA: IEEE. Retrieved from http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=4278472

120. IEEE Standards Association. (2000). *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. Piscataway, NJ, USA: IEEE.

121. Ionita, M. T., Hammer, D. K., & Obbink, H. (2002). Scenario-based software architecture evaluation methods: An overview. *Icse/Sara*.

122. IoTivity. (2017). Home | IoTivity. Retrieved September 7, 2017, from https://www.iotivity.org/

123. ISO 26262. (2011). Road vehicles functional safety. *International Standard ISO/FDIS*, *26262*.

124. ISO/IEC/IEEE. (2011). ISO/IEC/IEEE 42010:2011(E), Systems and software engineering — Architecture description, 1–46.

125. ITU-T. (2013). ITU-T Rec. Y.2060 (06/2012) Overview of the Internet of things, 1–22.

126. Jansen, A., & Bosch, J. (2005). Software Architecture as a Set of Architectural Design Decisions (pp. 109–120). Presented at the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05), IEEE. http://doi.org/10.1109/WICSA.2005.61

127. Jiang, D., & Delgrossi, L. (2008). IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments (pp. 2036–2040). Presented at the Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE. http://doi.org/10.1109/VETECS.2008.458

128. Johanning, V., & Mildner, R. (2015). Car IT kompakt. Springer-Verlag.

129. Kazman, R., Abowd, G., Bass, L., & Clements, P. (1996). Scenario-based analysis of software architecture. *Software, IEEE*, *13*(6), 47–55. http://doi.org/10.1109/52.542294

130. Kazman, R., Klein, M., & Clements, P. (2000). *ATAM: Method for Architecture Evaluation* (CMU/SEI-2000-TR-004) (pp. 1–83). Pittsburgh PA: Carnegie-Mellon University, Software Engineering Institute.

131. Kazman, R., Nielsen, C. B., & Klein, J. (2013). *Understanding Patterns for System of Systems Integration* (CMU/SEI-2013-TR-017). *SoSE* (pp. 141–146). IEEE.

132. Knirsch, A., Schnarz, P., & Wietzke, J. (2012). Prioritized access arbitration to shared resources on integrated software systems in multicore environments. *Nesea*, 1–8. http://doi.org/10.1109/NESEA.2012.6474014

133. Knirsch, A., Wietzke, J., Moore, R., & Dowland, P. (2010). An Approch for Structuring Heterogeneous Automotive Software Systems by use of Multicore Architectures. *Proceedings of the Sixth Collaborative Research Symposium on Security, E-Learning, Internet and Networking (SEIN2010)*, 19–30.

134. Knirsch, A., Wietzke, J., Moore, R., & Dowland, P. (2011). Resource Management for Multicore Aware Software Architectures of In-Car Multimedia Systems. In H.-U. Heiß, P. Pepper, H. Schlingloff, & J. Schneider (Eds.), (p. 216). Presented at the Informatik schafft Communities, Berlin.

135. Kopetz, H. (2011). Real-Time Systems. Springer Science & Business Media. http://doi.org/10.1007/978-1-4419-8237-7

136. Kosch, T., Kulp, I., Bechler, M., Strassberger, M., Weyl, B., & Lasowski, R. (2009). Communication architecture for cooperative systems in Europe. *IEEE Communications Magazine*, *47*(5), 116–125. http://doi.org/10.1109/MCOM.2009.4939287

137. Kossiakoff, A., Sweet, W. N., Seymour, S., & Biemer, S. M. (2011). Systems Engineering Principles and Practice. John Wiley & Sons.

138. Krajewski, J., Sommer, D., Trutschel, U., Edwards, D., & Golz, M. (2009). Steering wheel behavior based estimation of fatigue. Presented at the 5th International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design, Big Sky, Montana, USA.

139. Král, J., & Žemlička, M. (2000). Autonomous Components. In *SOFSEM 2000: Theory and Practice of Informatics* (Vol. 1963, pp. 375–383). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/3-540-44411-4_26

140. Krüger, I. H. (2005). Service-oriented software and systems engineering-a vision for the automotive domain (p. 143). Presented at the Proceedings Second ACM and IEEE International …. http://doi.org/10.1109/MEMCOD.2005.1487906

141. Krüger, I. H., Nelson, E. C., & Prasad, K. V. (2004). *Service-based software development for automotive applications*. SAE Technical Paper.

142. Laya, A., Alonso, L., & Alonso-Zarate, J. (2014). Is the Random Access Channel of LTE and LTE-A Suitable for M2M Communications? A Survey of Alternatives. *IEEE Communications Surveys & Tutorials*, *16*(1), 4–16. http://doi.org/10.1109/SURV.2013.111313.00244

143. Lee, E. A. (2006). The future of embedded software. Presented at the 21st International Technical Conference on the Enhanced Safety of Vehicles (ESV).

144. Lee, E. A. (2007). Computing foundations and practice for cyber-physical systems: A preliminary report. *University of California*.

145. Lee, J., Kim, Y., Kwak, Y., Zhang, J., Papasakellariou, A., Novlan, T., et al. (2016). LTE-advanced in 3GPP Rel -13/14: an evolution toward 5G. *IEEE Communications Magazine*, *54*(3), 36–42. http://doi.org/10.1109/MCOM.2016.7432169

146. Lequerica, I., Garcia Longaron, M., & Ruiz, P. M. (2010). Drive and share: efficient provisioning of social networks in vehicular scenarios. *IEEE Communications Magazine*, *48*(11), 90–97. http://doi.org/10.1109/MCOM.2010.5621973

147. Lu, Y., Li, S., & Shen, H. (2011). Virtualized Screen: A Third Element for Cloud-Mobile Convergence. *Multimedia, IEEE*, *18*(2), 4–11. http://doi.org/10.1109/MMUL.2011.33

148. Luckham, D. C. (2002). The Power of Events. Addison-Wesley Professional.

149. Maia, P., Cavalcante, E., Gomes, P., Batista, T., Delicato, F. C., & Pires, P. F. (2014). On the Development of Systems-of-Systems based on the Internet of Things (pp. 1–8). Presented at the 2014 European Conference on Software Architecture Workshops, New York, New York, USA. http://doi.org/10.1145/2642803.2642828

150. Maier, M. W. (1996). Architecting Principles for Systems-of-Systems. *INCOSE International Symposium*, *6*(1), 565–573. http://doi.org/10.1002/j.2334-5837.1996.tb02054.x

151. Marwedel, P. (2010). Embedded System Design. Springer Science & Business Media. http://doi.org/10.1007/978-94-007-0257-8

152. Masak, D. (2009). Der Architekturreview. Springer-Verlag. http://doi.org/10.1007/978-3-642-01659-2

153. McAffer, J., VanderLei, P., & Archer, S. (2010). OSGi and Equinox. Addison-Wesley Professional.

154. Medjiah, S. (2017). IoT Standards Landscaping & IoT LSP Gap Analysis (pp. 1–20). Presented at the Final STF 505 Presentation Workshop, Brussels.

155. Medvidovic, N., & Taylor, R. N. (2010). Software architecture: foundations, theory, and practice (Vol. 2). Presented at the 32nd ACM/IEEE International Conference on Software Engineering. http://doi.org/10.1145/1810295.1810435

156. Messelodi, S., Modena, C. M., Zanin, M., De Natale, F. G. B., Granelli, F., Betterle, E., & Guarise, A. (2009). Intelligent extended floating car data collection. *Expert Systems with Applications*, *36*(3), 4213–4227. http://doi.org/10.1016/j.eswa.2008.04.008

157. Microsoft. (2017). Azure IoT Suite | Microsoft Azure. Retrieved September 7, 2017, from https://azure.microsoft.com/en-gb/suites/iot-suite/

158. Mohammad, S. A., Rasheed, A., & Qayyum, A. (2011). VANET Architectures and Protocol Stacks: A Survey. *Lecture Notes in Computer Science*, *6596*(Chapter 9), 95–105. http://doi.org/10.1007/978-3-642-19786-4_9

159. Mumtaz, S., Huq, K. M. S., & Rodriguez, J. (2014). Direct mobile-to-mobile communication - paradigm for 5G. *IEEE Wireless Communications*, *21*(5), 14–23.

160. Mühl, G., Fiege, L., & Pietzuch, P. (2006). Distributed Event-Based Systems. Berlin/Heidelberg: Springer Science & Business Media. http://doi.org/10.1007/3-540-32653-7

161. Naphade, M., Banavar, G., Harrison, C., Paraszczak, J., & Morris, R. (2011). Smarter Cities and Their Innovation Challenges. *IEEE Computer*, *44*(6), 32–39. http://doi.org/10.1109/MC.2011.187

162. Natale, M. D., & Sangiovanni-Vincentelli, A. L. (2010). Moving From Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools. *Proceedings of the IEEE*, *98*(4), 603–620. http://doi.org/10.1109/JPROC.2009.2039550

163. NDS. (2017). NDS Association. Retrieved September 10, 2017, from https://www.nds-association.org/#thestandard

164. OASIS. (2006). *Reference Model for Service Oriented Architecture 1.0*. (C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, & R. Metz, Eds.).

165. Olumofin, F. G., & Goldberg, I. (2010). Privacy-Preserving Queries over Relational Databases. *Privacy Enhancing Technologies*.

166. OMA. (2017). Open Mobile Alliance. Retrieved September 10, 2017, from http://openmobilealliance.org/

167. OMA TS DM Protocol. (2016). *OMA Device Management Protocol* (V2_0-20160209-A) (pp. 1–105). Open Mobile Alliance.

168. OMA TS LightweightM2M. (2016). *Lightweight Machine to Machine Technical Specification* (V1_0-20160407-C) (pp. 1–127). Open Mobile Alliance.

169. OMA TS NGSI Context Management. (2012). NGSI Context Management. *openmobilealliance.org* (1st ed.).

170. oneM2M TS-0021. (2016). *oneM2M and AllJoyn Interworking*, (V2.0.0) (pp. 1–62).

171. oneM2M TR-0007. (2015). *Study of Abstraction and Semantics Enablements* (V2.5.1) (pp. 1–127). oneM2M.

172. oneM2M TS-0001. (2015). *Functional Architecture* (V1.6.1) (pp. 1–321). oneM2M.

173. oneM2M TS-0003. (2015). *Security Solutions* (V1.0.1) (pp. 1–91). oneM2M.

174. oneM2M TS-0004. (2015). *Service Layer Core Protocol Specification* (V1.0.1) (pp. 1–217). oneM2M.

175. oneM2M TS-0005. (2016). *Management Enablement (OMA)* (V1.4.1) (pp. 1–60). oneM2M.

176. oneM2M TS-0008. (2015). *CoAP Protocol Binding* (V1.0.1) (pp. 1–14). oneM2M.

177. oneM2M TS-0009. (2015). *HTTP Protocol Binding* (V1.0.1) (pp. 1–13). oneM2M.

178. oneM2M TS-0010. (2015). *MQTT Protocol Binding* (V1.0.1) (pp. 1–27). oneM2M.

179. Oxford Dictionary Online. (2012). Concept - definition of concept. Retrieved February 20, 2012, from https://en.oxforddictionaries.com/definition/concept

180. Oyman, O., Foerster, J. R., Tcha, Y., & Lee, S.-C. (2010). Toward enhanced mobile video services over WiMAX and LTE. *IEEE Communications Magazine*, *48*(8), 68–76. http://doi.org/10.1109/MCOM.2010.5534589

181. Papageorgiou, A., Schmidt, M., Song, J., & Kami, N. (2013). Smart M2M Data Filtering Using Domain-Specific Thresholds in Domain-Agnostic Platforms (pp. 286–293). Presented at the 2013 IEEE International Congress on Big Data (BigData Congress), IEEE. http://doi.org/10.1109/BigData.Congress.2013.45

182. Parkvall, S., Dahlman, E., Furuskar, A., Jading, Y., Olsson, M., Wanstedt, S., & Zangi, K. (2008). LTE-Advanced - Evolving LTE towards IMT-Advanced (pp. 1–5). Presented at the 2008 IEEE 68th Vehicular Technology Conference, IEEE. http://doi.org/10.1109/VETECF.2008.313

183. Parnas, D. L. (1972). On the Criteria To Be Used in Decomposing Systems into Modules. *Communications of the ACM*, *15*(12), 1053–1058. http://doi.org/10.1145/361598.361623

184. Pautasso, C., Zimmermann, O., & Leymann, F. (2008). Restful web services vs. “big"' web services: making the right architectural decision. Presented at the 17th international conference on World Wide Web, ACM. http://doi.org/10.1145/1367497.1367606

185. Pereira, C., & Aguiar, A. (2014). Towards Efficient Mobile M2M Communications: Survey and Open Challenges. *Sensors*, *14*(10), 19582–19608. http://doi.org/10.3390/s141019582

186. Picone, M., Busanelli, S., Amoretti, M., Zanichelli, F., & Ferrari, G. (2015). Advanced Technologies for Intelligent Transportation Systems. *Springer 2015*, *139*. http://doi.org/10.1007/978-3-319-10668-7

187. Pretschner, A., Broy, M., Kruger, I. H., & Stauner, T. (2007). Software Engineering for Automotive Systems: A Roadmap. *FOSE '07: 2007 Future of Software Engineering*, 55–71. http://doi.org/10.1109/FOSE.2007.22

188. Protzmann, R., Massow, K., & Radusch, I. (2014). An Evaluation Environment and Methodology for Automotive Media Streaming Applications (pp. 297–304). Presented at the 2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), IEEE. http://doi.org/10.1109/IMIS.2014.38

189. Quintero, R., Llamazares, A., Llorca, D. F., Sotelo, M. A., Bellot, L. E., Marcos, O., et al. (2011). Extended Floating Car Data system - experimental study. *Intelligent Vehicles Symposium (IV), 2011 IEEE*, 631–636. http://doi.org/10.1109/IVS.2011.5940444

190. Raja, B. S., Iqbal, M. A., & Ihsan, I. (2005). Moving From Problem Space to Solution Space. *World Academy of Science, Engineering and Technology*.

191. Ramesh, V., Glass, R. L., & Vessey, I. (2004). Research in computer science: an empirical study. *Journal of Systems and Software*, *70*(1-2), 165–176. http://doi.org/10.1016/S0164-1212(03)00015-3

192. Richardson, L., & Ruby, S. (2007). RESTful Web Services. O'Reilly Media, Inc.

193. Richardson, L., Amundsen, M., & Ruby, S. (2013). RESTful Web APIs. O'Reilly Media, Inc.

194. Rost, P., Banchs, A., Berberana, I., Breitbach, M., Doll, M., Droste, H., et al. (2016). Mobile network architecture evolution toward 5G. *IEEE Communications Magazine*, *54*(5), 84–91. http://doi.org/10.1109/MCOM.2016.7470940&orderBeanReset=true&startPage=84&endPage=91&volumeNum=54&issueNum=5","displayPublicationTitle“:”IEEE

195. Sagstetter, F. R. (2016). *Schedule Synthesis for Time-Triggered Automotive Architectures.* Technische Universität München.

196. Saleh, O. (2013). Complex Event Processing in Wireless Sensor Networks. *Grundlagen Von Datenbanken*.

197. Saleh, O., & Sattler, K.-U. (2013). Distributed Complex Event Processing in Sensor Networks (pp. 23–26). Presented at the 14th IEEE International Conference on Mobile Data Management (MDM), IEEE. http://doi.org/10.1109/MDM.2013.60

198. Salminen, A., & Tompa, F. (2011). Communicating with XML. Boston, MA: Springer Science & Business Media. http://doi.org/10.1007/978-1-4614-0992-2

199. Sandonis, V., Soto, I., Calderón, M., & Urueña, M. (2016). Vehicle to Internet communications using the ETSI ITS GeoNetworking protocol. *Transactions on Emerging Telecommunications Technologies*, 1–16. http://doi.org/10.1002/ett

200. Sangiovanni-Vincentelli, A. (2002). Platform-based Design, 1–20.

201. Sangiovanni-Vincentelli, A., & Di Natale, M. (2007). Embedded System Design for Automotive Applications. *IEEE Computer*, *40*(10), 42–51. http://doi.org/10.1109/MC.2007.344

202. Sangiovanni-Vincentelli, A., & Martin, G. (2001). Platform-Based Design and Software Design Methodology for Embedded Systems. *IEEE Design & Test*, *18*(6), 23–33. http://doi.org/10.1109/54.970421

203. SAP. (2017). Internet of Things | SAP Solutions. Retrieved February 24, 2018, from https://www.sap.com/uk/products/supply-chain-iot/iot.html

204. Sauter, M. (2015). Grundkurs Mobile Kommunikationssysteme. Wiesbaden: Springer-Verlag. http://doi.org/10.1007/978-3-658-08342-7

205. Schäuffele, J., & Zurawka, T. (2013). Automotive Software Engineering. Wiesbaden: Springer-Verlag. http://doi.org/10.1007/978-3-322-91194-0

206. Scheider, T., & Böhm, M. (2010). Extended Floating Car Data in Co-operative Traffic Management. In *Traffic Data Collection and its Standardization* (Vol. 144, pp. 161–170). New York, NY: Springer New York. http://doi.org/10.1007/978-1-4419-6070-2_11

207. Schneider, J., & Nett, T. (2014). Safety Issues of Integrating IVI and ADAS functionality via running Linux and AUTOSAR in parallel on a Dual-Core-System. *Automotive - Safety & Security*, 55–68.

208. SEBoK authors. (2015). Guide to the Systems Engineering Body of Knowledge (SEBoK), 1–1005.

209. Segal, J. (2003). The Nature of Evidence in Empirical Software Engineering (pp. 40–47). Presented at the Eleventh Annual International Workshop on Software Technology and Engineering Practice, IEEE. http://doi.org/10.1109/STEP.2003.33

210. Seo, H., Lee, K.-D., Yasukawa, S., Peng, Y., & Sartori, P. (2016). LTE evolution for vehicle-to-everything services. *IEEE Communications Magazine*, *54*(6), 22–28.

211. Shelby, Z., Hartke, K., & Bormann, C. (2013). Constrained Application Protocol (CoAP). CoRE Working Group, IETF.

212. Shimizu, N. (2004). Analysis of Automotive Telematics Industry in Japan.

213. Siebenpfeiffer, W. (2014). Vernetztes Automobil. (W. Siebenpfeiffer, Ed.). Wiesbaden: Springer-Verlag. http://doi.org/10.1007/978-3-658-04019-2

214. Simoens, P., De Turck, F., Dhoedt, B., & Demeester, P. (2011). Remote Display Solutions for Mobile Cloud Computing. *IEEE Computer*, *44*(8), 46–53. http://doi.org/10.1109/MC.2011.70

215. simTD-Consortium. (2013). simTD: Safe and intelligent mobility - test field Germany. Retrieved April 8, 2013, from http://www.simtd.de

216. simTD-Consortium. (2009, October 6). Deliverable D11.4. Retrieved April 16, 2013, from http://www.simtd.de/index.dhtml/15516d4d95665060970r/object.media/enEN/6486/CS/-/backup_publications/Projektergebnisse/simTD-Deliverable-D11.4_Funktionsanforderungen_Architektur.pdf

217. Smethurst, G. (2010). Changing the In-Vehicle Infotainment Landscape. *GENIVI Allliance*. Retrieved from http://www.genivi.org/sites/default/files/GENIVI%20White%20Paper%20-%20Changing%20the%20IVI%20Landscale.pdf

218. Sommerville, I. (2010). Software Engineering. Addison-Wesley.

219. Song, J. (2014). Global IoT/M2M Service Framework Standardizations – oneM2M perspective (pp. 1–49). Presented at the IoT - Beyond Connectivity Workshop.

220. Strasser, A., Cool, B., Gernert, C., Knieke, C., Körner, M., Niebuhr, D., et al. (2014). Mastering Erosion of Software Architecture in Automotive Software Product Lines. In *SOFSEM 2014: Theory and Practice of Computer Science* (Vol. 8327, pp. 491–502). Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-04298-5_43

221. Swetina, J., Lu, G., Jacobs, P., Ennesser, F., & Song, J. (2014). Toward a standardized common M2M service layer platform: Introduction to oneM2M. *IEEE Wireless Communications*, *21*(3), 20–26. http://doi.org/10.1109/MWC.2014.6845045

222. Takei, Y., & Furukawa, Y. (2005). Estimate of Driver's Fatigue Through Steering Motion (Vol. 2, pp. 1765–1770). Presented at the 2005 IEEE International Conference on Systems, Man and Cybernetics, IEEE. http://doi.org/10.1109/ICSMC.2005.1571404

223. Talbot, S. C., & Ren, S. (2009). Comparision of Fieldbus Systems CAN, TTCAN, Flexray and LIN in Passenger Vehicles. Presented at the 29th IEEE International Conference on Distributed Computing Systems.

224. Tehrani, M. N., Uysal, M., & Yanikomeroglu, H. (2014). Device-to-device communication in 5G cellular networks - challenges, solutions, and future directions. *IEEE Communications Magazine*, *52*(5), 86–92.

225. Terroso-Sáenz, F., Valdés-Vela, M., Campuzano, F., Botia, J. A., & Skarmeta-Gómez, A. F. (2015). A complex event processing approach to perceive the vehicular context. *Information Fusion*, *21*, 187–209. http://doi.org/10.1016/j.inffus.2012.08.008

226. Tesla Motors. (2016). Model S Autopilot Press Kit | Tesla Motors. Retrieved January 18, 2016, from https://www.teslamotors.com/presskit/autopilot

227. The GridWise Architecture Council. (2008). *GridWise Interoperability Context-Setting Framework*. Smart Grids Interoperability.

228. TIA. (2017). Telecommunications Industry Association. Retrieved September 10, 2017, from http://www.tiaonline.org/

229. Tiako, P. F. (2008). Designing Software-Intensive Systems. (P. F. Tiako, Ed.). IGI Global. http://doi.org/10.4018/978-1-59904-699-0

230. Tolk, A., & Jain, L. C. (2011). Intelligent-Based Systems Engineering. Springer Science & Business Media. http://doi.org/10.1007/978-3-642-17931-0

231. Tolk, A., & Muguira, J. A. (2003). The levels of conceptual interoperability model. Presented at the 2003 Fall Simulation Interoperability Workshop, Orlando, Florida.

232. Tolk, A., Diallo, S. Y., & Turnitsa, C. D. (2007). Applying the levels of conceptual interoperability model in support of integratability, interoperability and composability for system-of-systems engineering. *Journal of Systemics*.

233. TSDSI. (2017). TSDSI | Welcome to TSDSI. Retrieved September 10, 2017, from http://www.tsdsi.org/

234. TTA. (2017). Welcome to TTA - Telecommunications Technology Association of Korea. Retrieved September 10, 2017, from http://www.tta.or.kr/English/

235. TTC. (2017). The Telecommunication Technology Committee. Retrieved September 10, 2017, from http://www.ttc.or.jp/e/

236. Uckelmann, D., Harrison, M., & Michahelles, F. (2011). Architecting the Internet of Things. (D. Uckelmann, M. Harrison, & F. Michahelles, Eds.). Berlin, Heidelberg: Springer Science & Business Media. http://doi.org/10.1007/978-3-642-19157-2

237. Venkatesh Prasad, K., Broy, M., & Krueger, I. (2010). Scanning Advances in Aerospace & Automobile Software Technology. *Proceedings of the IEEE*, *98*(4), 510–514. http://doi.org/10.1109/JPROC.2010.2041835

238. Vergata, S., Knirsch, A., & Wietzke, J. (2012). Integration zukünftiger In-Car-Multimediasysteme unter Verwendung von Virtualisierung und Multi-Core-Plattformen. In *Herausforderungen durch Echtzeitbetrieb - Echtzeit 2011, Fachtagung des gemeinsamen Fachausschusses Echtzeitsysteme von Gesellschaft für Informatik e.V. (GI), VDI/VDE-Gesellschaft für Mess- und Automatisierungstechnik (GMA) und Informationstechnischer Gesellschaft im VDE (ITG), Boppard, 3. und 4. November 2011* (pp. 21–28). Springer.

239. Vergata, S., Wietzke, J., Schütte, A., & Dowland, P. (2010). System Design for Automotive Applications. *Proceedings of the Sixth Collaborative Research Symposium on Security, E-Learning, Internet and Networking (SEIN2010)*, 53–60.

240. Vinoski, S. (2008). Serendipitous Reuse. *IEEE Internet Computing*, *12*(1), 84–87. http://doi.org/10.1109/MIC.2008.20

241. Vogel, O., Arnold, I., Chughtai, A., Ihler, E., Kehrer, T., Mehlig, U., & Zdun, U. (2009). Software-Architektur. Springer Science & Business Media.

242. Völker, L. (2013). SOME/IP–Die Middleware für Ethernet-basierte Kommunikation. *HANSER Automotive Networks Special*.

243. W3C. (2013). Web Services Glossary. Retrieved April 12, 2013, from http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice

244. Wang, D. (2013). *Extending Complex Event Processing for Advanced Applications*. Worcester Polytechnic Institute.

245. Wang, W. (2009). The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S, 1–9.

246. Wannstrom, J. (2013, June). LTE-Advanced. Retrieved September 9, 2017, from http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced

247. Wietzke, J. (2012). Embedded Technologies. Springer-Verlag Berlin Heidelberg. http://doi.org/10.1007/978-3-642-23996-0

248. Wietzke, J., & Tran, M. T. (2005). Automotive Embedded Systeme (Xpert.press). Berlin/Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/3-540-28305-6

249. Winner, H., Hakuli, S., & Wolf, G. (2012). Handbuch Fahrerassistenzsysteme. Aufl. Vieweg+ Teubner.

250. Winner, H., Hakuli, S., Lotz, F., & Singer, C. (2015). Handbuch Fahrerassistenzsysteme. (H. Winner, S. Hakuli, F. Lotz, & C. Singer, Eds.). Wiesbaden: Springer-Verlag. http://doi.org/10.1007/978-3-658-05734-3

251. Wise-IoT. (2017). Home. Retrieved September 7, 2017, from http://wise-iot.eu/en/home/

252. Wu, G., Talwar, S., Johnsson, K., Himayat, N., & Johnson, K. D. (2011). M2M: From Mobile to Embedded Internet. *IEEE Communications Magazine*, *49*(4), 36–43. http://doi.org/10.1109/MCOM.2011.5741144

253. Zave, P. (1993). Feature interactions and formal specifications in telecommunications. *IEEE Computer*, *26*(8), 20–28. http://doi.org/10.1109/2.223539

254. Zhang, Q. (2008). Visual Software Architecture Description Based on Design Space (pp. 366–375). Presented at the 8th International Conference on Quality Software (QSIC), IEEE. http://doi.org/10.1109/QSIC.2008.59

255. Zhu, L., Babar, M. A., & Jeffery, D. R. (2004). Mining Patterns to Support Software Architecture Evaluation. (pp. 25–36). Presented at the Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture, IEEE Comput. Soc. http://doi.org/10.1109/WICSA.2004.1310687

256. Zimmermann, W., & Schmidgall, R. (2014). Bussysteme in der Fahrzeugtechnik. Springer-Verlag. http://doi.org/10.1007/978-3-658-02419-2

257. Zorzi, M., Gluhak, A., Lange, S., & Bassi, A. (2010). From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view. *IEEE Wireless Communications*, *17*(6), 44–51. http://doi.org/10.1109/MWC.2010.5675777

Abbreviations

3GPP	3rd Generation Partnership Project
ABS	Antilock Brake System
ACC	Adaptive Cruise Control
AD	Autonomous Driving
ADAS	Advanced Driver Assistance System
ADASIS	Advanced Driver Assistance Systems Interface Specification
ADN	Application Dedicated Node
AE	Application Entity
AMP	Asymmetric Multiprocessing
API	Application Programming Interface
ASDP	Automotive Service Delivery Platform
ASIL	Automotive Safety Integrity Level
ASM	Application and Service Layer Management
ASN	Application Service Node
ASR	Architecturally Significant Requirements
ATAM	Architectural Trade-off Analysis Method
AUTOSAR	AUTomotive Open System ARchitecture
BSD	Blind Spot Detection
BSW	Basic Software
CAN	Controller Area Network
Car2Car	Car-to-Car
Car2Infrastructure	Car-to-Infrastructure
Car2X	Car-to-X

CD	Compact Disc
CE	Consumer Electronic
CEP	Complex Event Processing
CMDH	Communication Management, Delivery Handling CSF
CoAP	Constraint Application Protocol
CPS	Cyber-Physical System
CPU	Central Processing Unit
CRUD	Create, Retrieve, Update, Delete
CRUD+N	Create, Retrieve, Update, Delete, and Notify
CSE	Common Services Entity
CSF	Common Services Functions
D2D	Device-to-Device
DA	Device Application
DAB	Digital Audio Broadcasting
DIS	Discovery
DM	Device Management
DMG	Device Management CSF
DMT	Data Management and Repository CSF
DSCL	Device Service Capability Layer
DSRC	Dedicated Short Range Communication
E2E	End-to-End
eCall	Automatic Emergency Call
ECU	Electronic Control Unit
EDA	Event-Driven Architectures

EDEL	Enhanced Data Exchange Layer
eHorizon	Electronic Horizon
EMC	Electro-Magnetic Compatibility
ESP	Electronic Stability Program
ETSI	European Telecommunications Standards Institute
FCD	Floating Car Data
FCW	Forward Collision Warning
FM	Frequency Modulation
GA	Gateway Application
GENIVI	Geneva In-Vehicle Infotainment
GMG	Group Management CSF
GPS	Global Positioning System
GPU	Graphics Processing Unit
GSCL	Gateway Service Capability Layer
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
H2M	Human-to-Machine
HAD	Highly Automated Driving
HATEOAS	hypermedia as the engine of application state
HMI	Human-Machine Interface
HTTP	Hypertext Transfer Protocol
ICM	In-Car Multimedia
ICT	Information and Communications Technology
IEC	International Electrotechnical Commission

IEEE	Institute of Electrical and Electronics Engineers
IERC	IoT European Research Cluster
IN	Infrastructure Node
IoT	Internet of Things
IP	Internet Protocol
IPE	Inter-working Proxy Application Entity
ISO	International Organization for Standardization
ITS	Intelligent Transportation System
IVI	In-Vehicle Infotainment
KB	Kilobyte
LCIM	Levels of Conceptual Interoperability
LDW	Lane Departure Warning
LIN	Local Interconnect Network
LKA	Lane Keeping Assistance
LOC	Location CSF
LTE	Long Term Evolution
LTE-A	Long Term Evolution Advanced
M2H	Machine-to-Human
M2M	Machine-to-Machine Communication
MANET	Mobile Ad hoc NETwork
MEC	Mobile Edge Computing
MN	Middle Node
MOST	Media Oriented System Transport
MP3	MPEG-2 Audio Layer III

MQTT	Message Queue Telemetry Transport
MTC	Machine-Type-Communication
NA	Network Application
NDS	Navigation Data Standard
NFV	Network Function Virtualisation
NoDN	Non-oneM2M Device Node
NSCL	Network Service Capability Layer
NSE	Network Services Entity
NSSE	Network Service Exposure, Service Execution and Triggering CSF
OASIS	Organization for the Advancement of Structured Information Standards
OBU	On Board Unit
OEM	Original Equipment Manufacturer
OMA	Open Mobile Alliance
OS	Operating System
OSI	Open Systems Interconnection
OTA	Over The Air
PBD	Platform-Based Design
PC	Personal Computer
POI	Points-of-Interest
POSIX	Portable Operating System Interface
PSAP	Public Safety Answering Point
RDP	Remote-Desktop-Protocol
REG	Registration
REQ	Requirement

REST	REpresentational State Transfer
RFID	Radio-Frequency Identification
RSU	Road Side Unit
RTE	Runtime Environment
SCA	Service Charging and Accounting CSF
SDK	Software Development Kit
SDN	Software-Defined Networking
SDO	Standard Developments Organisation
SEC	Security CSF
SN	Subscription and Notification
SOA	Service-Oriented Architecture
SoC	System on Chip
SOME/IP	Scalable Service-Oriented Middleware over IP
SoP	Start of Production
SQL	Structured Query Language
SW-C	Software Components
TCP	Transmission Control Protocol
T-CPS	Transportation Cyber-Physical System
TCS	Traction Control System
TJA	Traffic Jam Assistant
TJC	Traffic Jam Chauffeur
TPMS	Tire Pressure Monitoring System
TR	Technical Report
TS	Technical Specification

TSR	Traffic Sign Recognition
TTCAN	Time-triggered CAN
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VANET	Vehicular Ad hoc NETwork
VCG	Vehicle Communication Gateway
VFB	Virtual Function Bus
VMM	Virtual Machine Monitor
VoIP	Voice over IP
W3C	World Wide Web Consortium
WAVE	Wireless Access in Vehicular Environments
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network
XFCD	Extended Floating Car Data
XML	Extensible Markup Language
XSD	XML Schema Definition