A Context Map as the Basis for a Microservice Architecture for the Connected Car Domain

Sebastian Abeck[1], Michael Schneider[2], Jan-Philip Quirmbach[3], Heiko Klarl[4], Christof Urbaczek[5] and Shkodran Zogaj

Abstract: In the near future cars will have two properties: They will be electrically powered and they will be connected to the Internet. Such cars will provide a huge amount of sensor data which can be accessed via web APIs in order to develop innovative connected car applications, such as traffic control, hazard warning, assisted or even autonomous driving. However, current software solutions in this field are mainly monoliths solving single problems in an isolated way. Therefore, we propose a systematic approach by which each single connected car application becomes part of a microservice architecture. This approach requires a sound and well-elaborated domain model from which the microservices' APIs and implementation of the applications can be systematically derived. The main contribution of this paper is a context map for the connected car domain. We demonstrate a structured software development approach with the example of a mobile application, the Electric Car Charger, by showing how this application is integrated into the context map and, thus, into a connected car microservice architecture.

Keywords: ***Connected car, microservice architecture, domain modeling, context map, bounded context, API***

1 Introduction

Connected cars are in the center of innovative and complex mobility concepts for our society [Co+16]. Such mobility solutions, in which cars are only one means of transportation besides bus, train, bikes etc., requires the exchange of data between all involved transportation means (vehicle-to-vehicle) and the transportation infrastructure (vehicle-to-infrastructure) via the Internet. Therefore, the Internet of Things (IoT) aspect plays an important role in the field of integrated mobility solutions [DK+18]. Connected cars are one of these "things" of the Internet for which such new mobility services are offered. They can be seen as the drivers of IoT-based mobility solutions resulting from the economic power of the automotive industry. The necessary movement towards e-cars and their integration into an overall Internet-based mobility infrastructure lead to disruptive changes in this industrial domain. Besides the traditional automobile manufacturers offering cars as a product to their customers, new companies from the IT domain appear on stage. They perceive the cars as things of the Internet and provide

[1] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, sebastian.abeck@kit.edu
[2] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, michael.schneider@kit.edu
[3] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, jan-philip.quirmbach@student.kit.edu
[4] xdi360, Munich, Germany, heiko.klarl@xdi360.com
[5] xdi360, Munich, Germany, christof.urbaczek@xdi360.com

connected car services. Examples of such services are shown in Fig. 1.

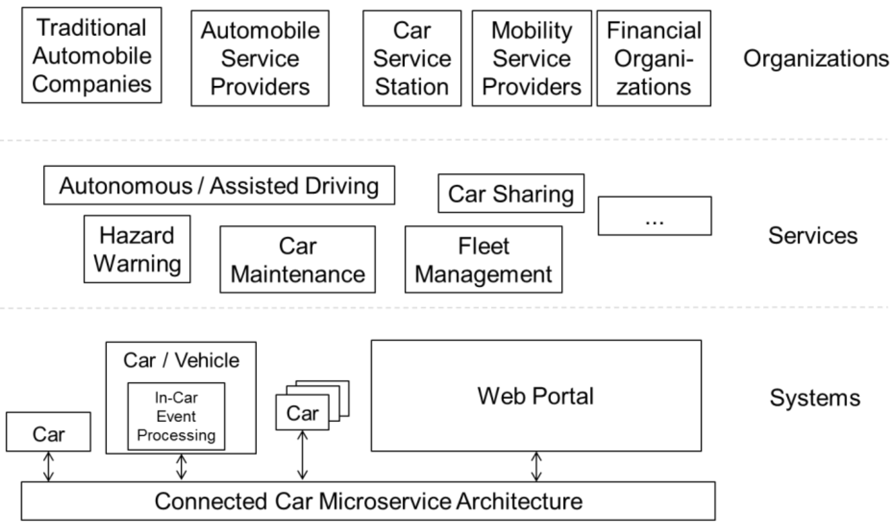


Figure 1: Examples of Connected Car Services

To be able to develop flexible and maintainable connected car solutions, a software architecture is needed which can be easily extended by new functional services. These services are provided by, and offered to, different organizations. We believe that a microservice architecture [Ne15] is an adequate concept to build a connected car system. This system consists of loosely coupled connected car services using other services via web APIs specified in a standardized language (e.g. OpenAPI). The microservice architecture is based on a domain model [Ev03] which prescribes the functional structure of the connected car domain.

Starting from the related work (Section 2), this paper elaborates a context map and its included bounded contexts of the connected car domain model (Section 3). The usage of the elaborated domain model artifacts is shown with the example of an Electric Car Charger (ECC) service (Section 4). The main advantage of our approach is the integration of the connected car service, in our case the ECC, into the overall connected car domain. The domain model and the derived microservice architecture provide the basis for all connected car services leading to a non-monolithic, loosely coupled connected car system (Section 5).

2 Related Work

Intensive work on digital technology and software engineering in the automotive sector started about the turn of the millennium [Br03]. The main competence of car companies traditionally lies in the field of mechanical and electrical engineering. In order to cope with the high complexity of automotive software, frameworks specific for the automotive domain were developed, such as the Volvo Cars Architecture Framework [PK+16] or the Automotive Architecture Framework [BG+09]. A characteristic of this work is the focus on the architecture of the software that is needed in a car. In [PK+16] the aspect of connected cars is covered in two so-called viewpoints, namely "connected cars and safety" and "security and privacy of connected cars". Although such automotive frameworks cover certain aspects of the automotive domain, they do not provide a domain model which is one of the main goals of this paper.

The Domain-Driven Design (DDD) [Ev03, Ve13] provides the conceptual foundation of our approach. As shown by [SH+18, HG+17], DDD can be applied in a structured software development process in order to derive a sound and comprehensible microservice architecture. A central part of the domain model is the so-called context map which is the result of DDD's strategic modeling. A context map is used to decompose the domain into subject-specific (especially not technically-driven) parts which are called bounded contexts. Since each bounded context is a candidate for a microservice [Ne15], the context map can be seen as a blueprint of the microservice architecture for the modeled domain. In [TH+18] a systematic approach to derive the bounded contexts in order to identify microservices is presented. The functional decomposition is carried out based on the requirements on the software system. A characteristic of this approach is given by the fact that a concrete software system, and not the domain, is in the focus. Therefore, a context map of the domain is not part of this approach.

Existing white papers from different companies (e.g. [KA+16, VA+14, DK12]) provide a fine-grained decomposition of a connected car's application landscape into different categories, such as navigation, vehicle management, or safety. This related work describes the domain in a more or less informal way. Nevertheless, for our work they provide a valuable practical input for the formal connected car domain model which we develop in the next Chapter 3 and apply to build a microservice-based application in Chapter 4.

3 Connected Car Categories

In the related work, different categories for the connected car domain are proposed which are illustrated in Figure 2. Vehicle management and driving management are directly related to the core functionality of a car. The category vehicle management is divided into the sub-categories remote control, diagnosis and maintenance; sub-categories of driving management are driver assistance, parking, and refueling [KA+16].

Further, there exists a cross-cutting functionality safety and security. Safety and security need to be concerned by all other categories, most important for vehicle management and driving management, because critical functionalities need to be secure. For example, it should not be possible that one can remotely control a car without permission. Safety and security can be divided into further sub-categories, such as emergency and theft protection.

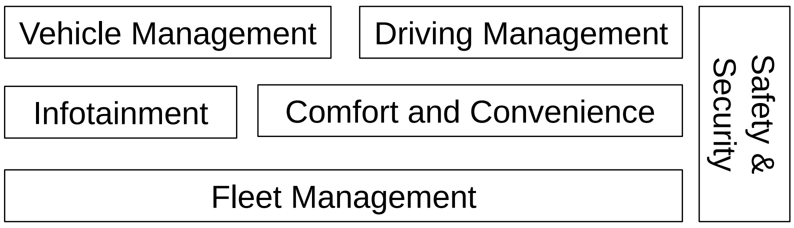


Figure 2: Existing Connected Car Categories

Infotainment as well as comfort and convenience provide less critical, but relevant functionalities. The infotainment category implies entertainment, information and smartphone integration and deals with streaming music and videos, interacting with social networks and providing news and weather information. Hand-free calls are an example of smartphone integration. Furthermore, information about the current traffic and navigation are also attributed to this category. Comfort and convenience are divided into the sub-categories well-being, interaction, and payment. Comfort and convenience include personalizing the vehicle, for example by pre-setting seats, temperature or ambience lighting.

In addition, issues referring to the governance of services across many cars deal with the category fleet management which consists of sub-categories like policies and optimization [DK12].

4 Formalization Based on UML and Domain-Driven Design

We use the existing proposals of a decomposition of the connected car domain to develop a formalized domain model based on the Domain-Driven Design (DDD, [Ev03]). This domain model serves as a design artifact from which we derive the microservice architecture for all connected car applications we are developing. The approach is not specific to the connected car domain since we use it also for other complex domains.

The context map displays the strategical relationships of a domain [HS+19, HG+17]. A context map consists of subdomains, bounded contexts and relationships between the bounded contexts. Following DDD, the bounded contexts are assigned to domain-specific subdomains, which further improve the overview of the domain. According to our approach, subdomains are modeled as a UML package which is extended by the stereotype <<subdomain>>.

A bounded context represents a candidate for a microservice which can be developed by an independent team [Ne15]. We formalize a bounded context as a packaging component which is annotated with the stereotype <<bounded context>>. Each bounded context contains tactical models like the relation view which describes the inner structure of this bounded context [SH+18].

Relationships between bounded contexts are formalized using UML associations extended by stereotypes corresponding to the context map relation. Depending on the type of relationship, the team communication between the bounded contexts is defined. [Ev14] provides several communication patterns for the relationships between bounded contexts. For example, the pattern <<conformist>> is a directed association between two bounded contexts. The consuming service has no influence on the offering service. Foreign bounded contexts are encapsulated by an <<anti-corruption layer>> (ACL). The ACL is formalized as a package which is part of the bounded context that uses the foreign bounded context.

5 Context Map for the Connected Car Domain

A decomposition of the connected car domain into subdomains and bounded contexts based on the formalization is derived. The context map, as shown in Figure 3 displays the result of the formalized connected car domain and suggests a separation of the different software services. For an easy overview and a better understanding, we put the main subdomains and bounded contexts in the center. Cross-section bounded contexts are placed on the right side of the context map diagram. Domain-enhancing bounded contexts that have a stronger user interaction are placed above the central area, and, finally, domain-supplementing bounded contexts, which express a more technical content, are located below the central area. Subdomains and bounded contexts that are close together are modeled in close proximity.

The context map is a design artifact of a structured software development process for microservice-based applications. Typically, CamelCase and PascalCase are used as a naming convention in such software development artifacts (e.g. VehicleManagement instead of vehicle management or Vehicle Management).

The category vehicle management offers a good starting point for the derivation of the subdomain VehicleManagement. We see these sub-categories as services for the vehicle management and therefore, bounded contexts for vehicles, sensor processing, remote control, diagnostics, and maintenance are established for this subdomain. The bounded context SensorProcessing processes the raw sensor data and provides semantically

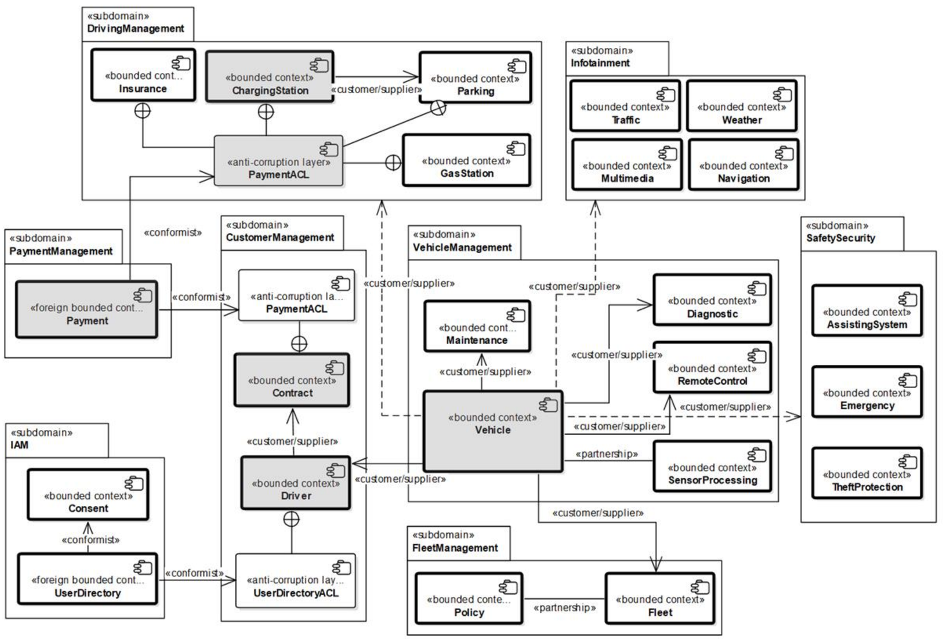

«subdomain» DrivingManagement
«bounded cont... Insurance
«bounded context» ChargingStation
«customer/supplier»
«bounded context» Parking
«anti-corruption layer» PaymentACL
«bounded context» GasStation
«subdomain» Infotainment
«bounded context» Traffic
«bounded context» Weather
«bounded context» Multimedia
«bounded context» Navigation
«conformist»
«subdomain» PaymentManagement
«foreign bounded cont... Payment
«conformist»
«subdomain» CustomerManagement
«anti-corruption la... PaymentACL
«bounded context» Contract
«customer/supplier»
«bounded context» Driver
«anti-corruption lay... UserDirectoryACL
«subdomain» IAM
«bounded context» Consent
«conformist»
«foreign bounded cont... UserDirectory
«conformist»
«customer/supplier»
«subdomain» VehicleManagement
«customer/supplier»
«bounded cont... Maintenance
«bounded context» Diagnostic
«customer/supplier»
«bounded context» RemoteControl
«customer/supplier»
«bounded context» Vehicle
«customer/supplier»
«customer/supplier»
«partnership»
«bounded context» SensorProcessing
«customer/supplier»
«customer/supplier»
«subdomain» SafetySecurity
«bounded context» AssistingSystem
«bounded context» Emergency
«bounded context» TheftProtection
«subdomain» FleetManagement
«bounded conte... Policy
«partnership»
«bounded context» Fleet

Figure 3: Proposed Context Map of the Connected Car Domain

enriched IoT data via an API. An example of how IoT platforms manage their sensor data with an IoT gateway and offer their sensor data by providing an API is given in [MK+17]. Functionalities for one remote controlling the vehicle are offered by the bounded context RemoteControl. The bounded context Diagnostic includes aspects like driving behavior analysis and telemetry data transmission. The bounded context Maintenance uses diagnostic data to perform predictive maintenance. If necessary, remote maintenance is handled by this bounded context. In addition, information from the bounded context Vehicle can be used to determine the functions supported by the vehicle. The bounded context Vehicle is one of the most important ones, because it offers the data base for many other bounded contexts.

In addition to the vehicle management, a customer management is required. This subdomain is needed in a connected car domain and the derived microservice architecture, even though no such category exists. Therefore, we added the subdomain CustomerManagement. This subdomain manages the data of the customer. For example, only the owner (or privileged users) of the car should be allowed to use the remote-control service for locking and unlocking the vehicle. The user-specific information is handled by the bounded context Driver. Since the customer is bound to contracts, we added a bounded context Contract. The customer data could be provided by a foreign identity and access management (IAM) system. The bounded context Driver uses the foreign bounded context UserDirectory. The UserDirectoryACL provides an additional layer and handles the transformation of the external and internal data for the bounded context Driver. Thus, the bounded context Driver can use its own data representation.

In our context map, the category driving management results in a subdomain DrivingManagement. An assisting system for parking helps the driver to simplify the parking process, whereas a parking system supports the driver to find free parking lots. In addition, services offering information about gas stations are part of this subdomain. Thus, we derived the bounded contexts Parking and GasStation which provide the necessary information. A connection to an external payment service could simplify or fully automate the payment process.

Further, there is the infotainment category which implies entertainment, information and smartphone integration. These services are outsourced into independent bounded contexts as Traffic, Multimedia, Weather, and Navigation, in order to be able to adequately handle the underlying domain logic. This is necessary to guarantee the understanding and uniform representation of the information. For example, multimedia can also be separated into a bounded context, which takes over the connection to third parties and ensures uniform formats for video, image, and audio.

One of the most relevant subdomains in the context map of the connected car domain is SafetySecurity. We established a bounded context for each of the subcategories, since each of these can be encapsulated as a separate service: The bounded context TheftProtection may offer an alarm (locally and on the smartphone), as well as the tracking of the vehicle on the smartphone and automatic associated damage reports. In case of a technical defect or an accident, the bounded context Emergency can process the data. Through the connection to the bounded context Vehicle, relevant vehicle-specific data can be automatically retrieved and transmitted for the intervention teams.

The classification of the category comfort into the microservice architecture is not straightforward because the subdomain Comfort does not fit the connected car domain. However, several subdomains can be derived from this category. A payment provider is required to process the payment. Therefore, a new bounded context Payment is derived and placed into the subdomain PaymentManagement. Further, the category comfort contains the automatically pre-setting of the seat position for the driver. For this reason, the bounded context Driver manages the driver together with their data and personalized vehicle settings which could be used for a car sharing application.

The fleet management is particularly relevant for car sharing or for company fleets. A distinction between analysis, optimization, and guidelines is important for this area. The owner of the vehicle is a company, while the driver is an employee. Based on this, the functionalities and authorizations differ in the context of fleet management; the company is given access to data such as locations (logistics/just-in-time) and fleet consumption. These issues are captured in the subdomain FleetManagement. The bounded context Fleet is responsible for cross-fleet analysis, while the bounded context Policy can be used to define certain rules that must be observed by the vehicles in the fleet.

6 Case Study: Electric Car Charger Application

The connected car context map we have developed in the last chapter builds the fundament for all applications of this domain. One such application is the Electric Car Charger (ECC) which we informally describe in Section 6.1. In the following Section 6.2, we show how ECC fits into the connected car context map and we illustrate the development process with the example of the ECC application. We summarize the benefits of the context map's use for software development in Section 6.3.

6.1 ECC Domain Objects and Relationships

The ECC application implements a software solution for charging stations for electric cars. This application allows the user to search for charging stations displayed on a map view. Furthermore, it is possible to filter these stations based on several attributes, e.g. by plug type. The ECC also enables a monitoring function during the charging process to obtain further information during the charging.

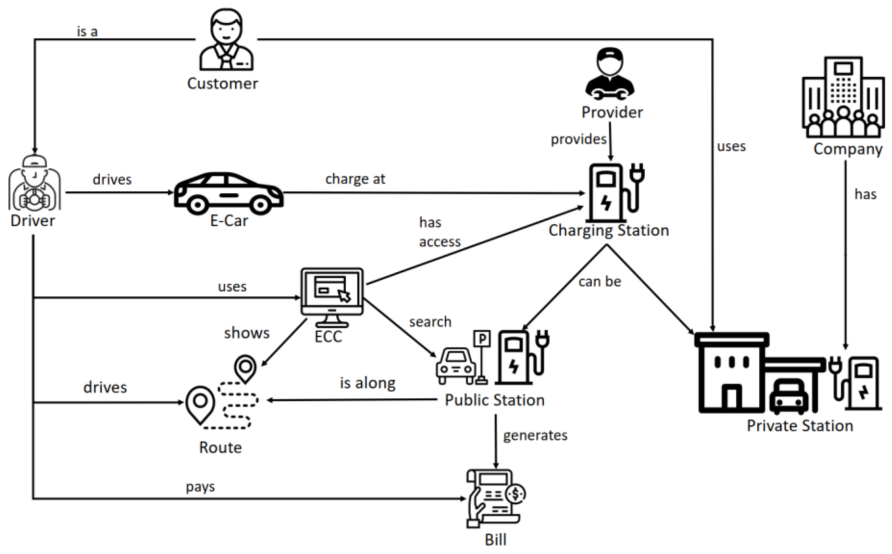


Figure 4: Domain Sketch of the ECC Application

Figure 4 shows a sketch of all relevant subjects and objects and their relations. The sketch provides an overview of the application-related part of the domain. The central element of the sketch is the charging station. A charging station can be a public station which could be installed, for example, at a parking lot or a private station which can be installed from a company at its private property. A customer is also the driver of the e-car. A driver can use the ECC application to (i) get information about the charging station, (ii) monitor the charging process, (iii) search a public station, and (iv) show all public stations along a certain route the driver wants to take. An e-car can charge its battery at a charging station which a provider provides. When an e-car is charged at a public station the station generates a bill.

6.2 Use of the Context Map

The ECC context map was developed with the help of the overall context map of the introduced connected car domain. The proposed context map in Figure 3 shows the placement of the ECC by dyeing the relevant ECC objects in grey. The ECC-relevant parts of the context map were identified as follows: The main part for the ECC is the bounded context ChargingStation in which the ECC application was developed. The bounded context Vehicle is required to access the information concerning the battery of the vehicle. Due to the fact that the vehicle is related to the bounded context Driver, it is possible to get information about the driver. The driver also provides the contracts of the driver through the bounded context Contract. The contract for the usage of a private charging station is stored in this bounded context. The relationship to the bounded context Payment is required for the payment of the charging process.

The development process that is used during the development of the ECC application is based on Behavior-Driven Development (BDD) [SM15] and Domain-Driven Design (DDD) [Ev03]. Figure 5 shows how the context map is related to software development artifacts. During the analysis phase, the required functionalities are written in Gherkin features which are the central BDD artifact. The advantage of Gherkin is that the features can not only be written in a human-readable way, but also be executed and tested. Each Gherkin feature belongs to one bounded context, which is also a candidate for a microservice. A bounded context usually consists of several features.

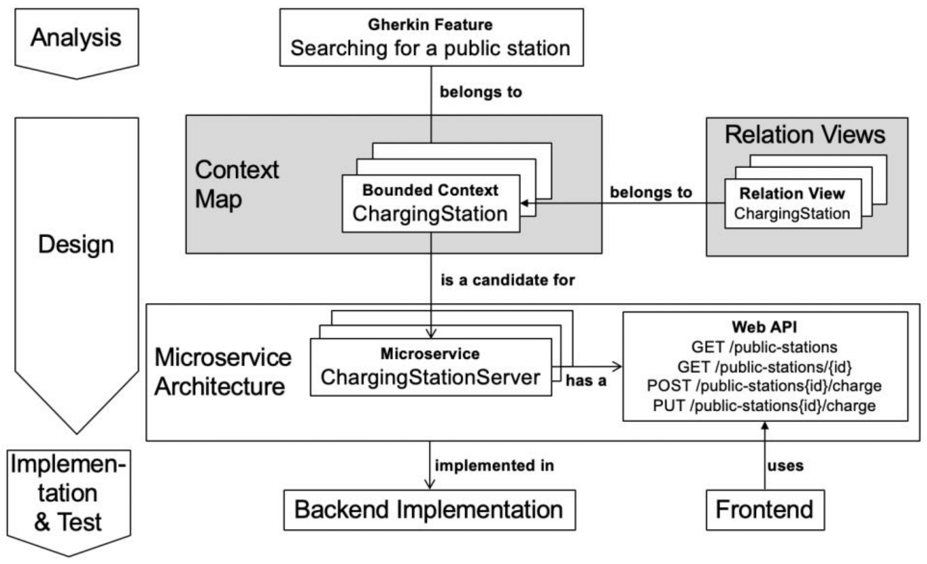
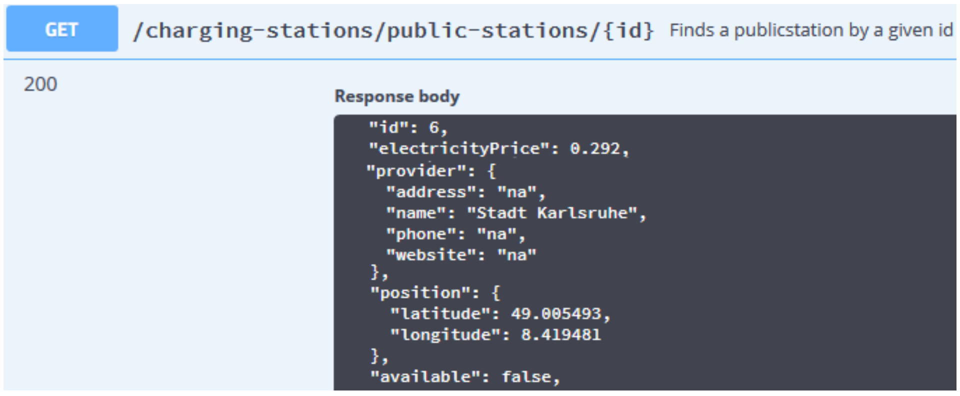


Figure 5: Context Map and related Software Development Artifacts

One feature of the ECC is searching for a public station (see Figure 5). This feature should display all public stations on a map view. During the design phase, the context map for the ECC application was designed. The ECC application was realized in the bounded context ChargingStation. For the technical interface of the resulting microservice, a web API based on the architectural style REST (REpresentational State Transfer, [Fi00]) was designed which offers the required functionality of the ECC. Figure 6 shows an excerpt of the Swagger UI for the request GET/public-stations/{id}.

The implementation of the backend and frontend was split and developed by two teams that could work almost independently of each other. The frontend team implemented the graphical user interface for the ECC, whereas the backend team implemented the required functionality in the backend and exposed it via the web API. The data of the public stations can be accessed via one of the web API methods, in particular the GET /public-stations method.


GET
/charging-stations/public-stations/{id}
Finds a publicstation by a given id
200
Response body

```
  "id": 6,
  "electricityPrice": 0.292,
  "provider": {
    "address": "na",
    "name": "Stadt Karlsruhe",
    "phone": "na",
    "website": "na"
  },
  "position": {
    "latitude": 49.005493,
    "longitude": 8.419481
  },
  "available": false,
```

Figure 6: GET Request for a Specific Charging Station

Figure 7 illustrates the map view as the central frontend element of ECC. Charging stations that are in close proximity are clustered, as shown by the numbered black dots on the map.

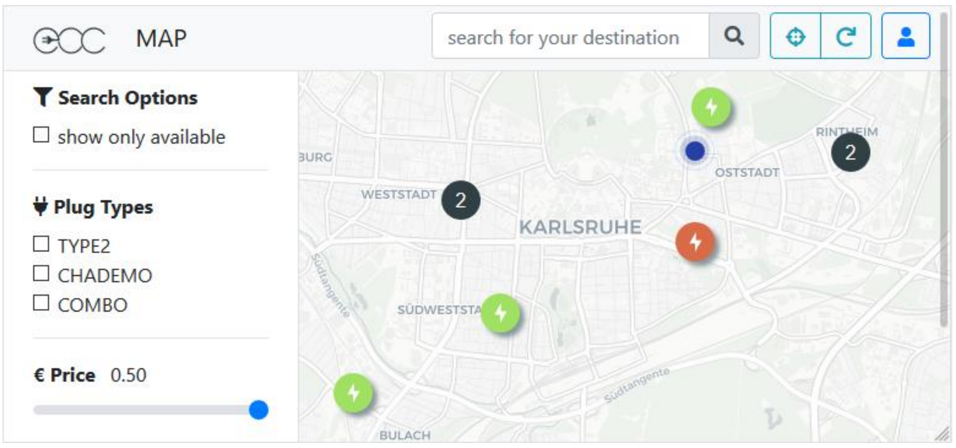


Figure 7: Map View of the ECC Application

Depending on the zoom level, the points are clustered together. When the zoom level is increased the clusters are resolved. Single charging stations are indicated by the bolt icon. An orange bolt icon means that the charging station is currently not available, whereas a green icon indicates a currently available charging station. The blue dot represents the current position of the user. The filter options described in Section 6.1 can be applied in the left menu. Further and detailed information about a charging station is available when one clicks the icon of a charging station.

6.3 Benefits of the Context Map for the Development of Microservices

A context map introduces a structure of a domain that is elaborated by domain experts over a long period of time. The knowledge of the domain is processed in a way that the microservice architecture can be derived from the context map and the microservices from the including bounded contexts. Whenever a new application adhering to the domain should be developed the software development team benefits from the domain knowledge captured by the context map. In the example of the Electric Car Charger, the connected car context map provides a primary architectural structuring of this application (e.g. VehicleManagement including Vehicle, DrivingManagement including ChargingStation, etc., see Figure 3) in a way that the connected car application, ECC, fits into the overall domain structure, and thus into the overall connected car microservice architecture derived from this structure. This enables the re-use of microservices that were implemented during the development of former connected car applications. In our specific case, before the ECC application, a car sharing application was developed which, among others, required the implementation of the Vehicle and the Driver bounded context as microservices. Since the car sharing application and the ECC application are based on the same connected car context map, they can share parts of the map, in this specific case microservices related to the vehicle and the driver. The more applications based on the context map are developed the more microservices can be re-used by newly developed applications.

7 Conclusion

A sound understanding of the domain for which a software application should be developed is necessary. A misunderstanding of the stakeholders who should have the domain knowledge and the developer of the software is the main reason why software projects often fail [Sm15]. In the connected car domain a common understanding is constantly growing because there is a high demand on flexible and environmentally friendly mobility solutions. So far, this understanding is documented in an informal way mostly in white papers from companies. We presented an approach on how to formalize the domain knowledge that is available in the field of connected car. Our approach is based on the widely accepted software design concept of Domain-Driven Design. Since this concept provides no formalization on the level of the modeling language, we extended the (also well accepted) Unified Modeling Language to be able to specify the strategic and tactical modeling parts of the domain model by different diagrams. A central diagram which expresses the main structure of the domain is the context map. In this paper, we proposed an initial draft of a context map for the connected car domain. Certainly, the concrete subdomains and including bounded contexts are subject for further discussions. The real value of our contribution is the systematic and formally sound approach on which the discussion of the domain knowledge with experts from the domain can be started – and documented in a way that this knowledge can be directly used in a structured development process. We believe that the close connection of domain knowledge capturing (also called knowledge crunching) with the software development process is a main advantage of our approach.

We demonstrated our approach with the example of the microservice-based software system Electric Car Charger. We have shown how the context map becomes a central design artifact of the software development process. The context map expresses the main structure of the domain and makes sure that the independently developed microservices are fitting into an overall connected car service landscape. Our approach guarantees that the model and its implementation are always in sync – according to our practical experience this is one of the most important demands of Domain-Driven Design. So far, the alignment of model and implementation is mainly done manually leaving room for model-to-code and code-to-model automation.

References

[Br03] Manfred Broy: Automotive Software Engineering. 25th International Conference on Software Engineering, 2003.

[BG+09] Manfred Broy, Mario Gleirscher, Stefano Merenda, Doris Wild, Peter Kluge, Wolfgang Krenzer: Automotive Architecture Framework: Towards a Holistic and Standardised Sys-tem Architecture Description, Technical Report of the of the Technische Universität München and White Paper of the IBM Cooperation, June 2009.

[Co+16] Riccardo Coppola, Maurizio Morisio: Connected Car: Technologies, Issues, Future Trend. ACM Computing Surveys, Vol. 49, No. 3, Article 46, Publication date: October 2016.

[DK12] Vivek Diwanji; Nilesh Karamarkar: Exploring the Connected Car, Whitepaper, cognizant. 2012, URL: https://www.cognizant.com/InsightsWhitepapers/Exploring-the-Connected-Car.pdf, [retrieved: 2019.04.02].

[DK+18] Soumya Kanti Datta, Mohammad Irfan Khan, Lara Codeca, B. Denis, Jerome Haerri, Christian Bonnet: IoT and Microservices Based Testbed for Connected Car Services, IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), p. 14 – 19, 2018.

[Ev03] Eric Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley Professional, 2003.

[Fi00] Roy T. Fielding.: Architectural Styles and the Design of Network-based Software Architectures, University of California, Irvine, Dissertation, 2000. https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation_2up.pdf, [retrieved: 2019.04.02].

[HG+17] Benjamin Hippchen, Pascal Giessler, Roland H. Steinegger, Michael Schneider, Sebastian Abeck: Designing Microservice-Based Applications by Using a Domain-Driven Design Approach, International Journal of Advances in Software, ISSN 1942-2628, vol. 10, no. 3&4, pages 432 - 445, 2017

[HS+19] Benjamin Hippchen, Michael Schneider, Iris Landerer, Pascal Giessler, Sebastian

Abeck: Methodology for Splitting Business Capabilities into a Microservice Architecture: Design and Maintenance Using a Domain-Driven Approach, Conference on Advances and Trends in Software Engineering (SOFTENG 2019), Valencia, 2019.

[KA+16] Per-Henrik Karlsson, Hong K. Ahn; Byeongmin Choi: Connected Car – A New Eco-system, Ipsos Business Consulting. 2016, URL: https://www.ipsos.com/sites/default/files/2016-06/022.1-connected-car-a-new-ecosystem.pdf, [retrieved: 2019.04.02].

[MK+17] Arthur de M. Del Esposte, Fabio Kon, Fabio M. Costa, Nelson Lago: InterSCity- A Scalable Microservice-based Open Source Platform for SmartCities, In Proceedings of the 6th International Conference on Smart Cities and Green ICT Systems (SMARTGREENS), pages 35-46, 2017.

[Ne15] Sam Newman: Building Microservices, O'Reilly Media, Inc., 2015.

[PK+16] Patrizio Pelliccione, Eric Knauss, Rogardt Heldal, Magnus Agren, Piergiuseppe Mal-lozzi Anders Alminger, Daniel Borgentun: A proposal for an Automotive Architecture Frame-work for Volvo Cars, IEEE Workshop on Automotive Systems/Software Architectures, 2016.

[SH+18] Michael Schneider, Benjamin Hippchen, Sebastian Abeck, Michael Jacoby, Reinhard Herzog: Enabling IoT Platform Interoperability Using a Systematic Development Approach by Example, Global Internet of Things Summit (GIoTS). IEEE, 2018. pages 1 – 6, 2018.

[Sm15] John Ferguson Smart: BDD in Action – Behavior-Driven Development for the whole software lifecycle. Manning Publications, 2015.

[TH+18] Shmuel Tyszberowicz, Robert Heinrich, Bo Liu, and Zhiming Liu: Identifying Micro-services Using Functional Decomposition. International Symposium on Dependable Software Engineering: Theories, Tools, and Applications. Springer, Cham, 2018.

[VA+14] Richard Viereckl, Jörg Assmann; Christian Radüge: In the fast lane – The bright future of connected cars, strategy&., 2014, URL: https://de.scribd.com/document/379805993/Strategyand-In-the-Fast-Lane-pdf, [retrieved: 2019-04-02].

[Ve13] Vaughn Vernon: Implementing Domain-Driven Design. 1st. Addison-Wesley Professional, 2013.