A Generic System for Automotive Software Over The Air (SOTA) Updates Allowing Efficient Variant and Release Management

Houssem Guissouma[1], Axel Diewald[1] and Eric Sax[1]

Karlsruhe Institute of Technology, Engesserstr. 5, 76131 Karlsruhe, Germany
{houssem.guissouma, axel.diewald, eric.sax}@kit.edu

Abstract. The introduction of Software Over The Air (SOTA) Updates in the automotive industry offers both the Original Equipment Manufacturer and the driver many advantages such as cost savings through inexpensive over the air bug fixes. Furthermore, it enables enhancing the capabilities of future vehicles throughout their life-cycle. However, before making SOTA a reality for safety-critical automotive functions, major challenges must be deeply studied and resolved: namely the related security risks and the required high system safety. The security concerns are primarily related to the attack and manipulation threats of wireless connected and update-capable cars. The functional safety requirements must be fulfilled despite the agility needed by some software updates and the typically high variants numbers.

We studied the state of the art and developed a generic SOTA updates system based on a Server-Client architecture and covering main security and safety aspects including a rollback capability. The proposed system offers release and variant management, which is the main novelty of this work. The proof of concept implementation with a server running on a host PC and an exemplary Electric/Electronic network showed the feasibility and the benefits of SOTA updates.

Keywords: Connected Vehicles, SOTA Updates, Variant Management, Security, Safety, Release Management, Electronic Control Unit

1 Introduction

Electric/Electronic (E/E) architectures include nowadays up to 150 ECUs with various safety and real-time demands and over 100 million lines of code [1]. The increasing integration of electronics and software in modern vehicles in form of embedded systems raises the error probability of ECU's code. These errors cause the program to perform in a way that produces an unintended outcome [2], which can lead to system failures needing to be fixed by an adequate software update. A lot of efforts are spent to detect these errors before the final production [3]. But more and more often errors occur during use. In this case, updates are urgent and the Original Equipment Manufacturer (OEM) must develop a bug fix, which is usually updated during global recall campaigns.

In 2020, 75 % of cars shipped globally are expected to have wireless connectivity [4]. One key benefit of the rising vehicle communication is the deployment of software or firmware updates over the air. In this work, we describe both kinds (software and firmware) as SOTA considering firmware as a special case of software. The Over The Air (OTA) approach would minimize the customer inconvenience and allow faster updates, since the software could be downloaded, as soon as the release is ready [5]. Otherwise, it will save important costs compared to the traditional update process in the workshop. According to IHS Technology, the total worldwide OEM cost savings from SOTA updates are forecasted to grow to over $35 billion in 2022 [6]. For these reasons, many institutions in the automotive industry are currently working on the introduction of SOTA updates as core feature of their vehicles and agree that this kind of updates will gain more importance in the upcoming years [7].

However, before licensing SOTA updates for ECUs and including them in the fleet's life-cycle management, multiple challenges must be further studied. In addition to security risks, such as manipulating the update's content by an unauthorized third party, the existing enormous number of system variants needs new processes, methods and tools to achieve efficient and safe SOTA updates [7]. This variants abundance, rising from the multitude of customer's wishes and the numerous configuration's possibilities, makes the validation of software releases for whole product lines a difficult task.

This paper is structured as follows: in Sect. 2, we studied the state of the art of software updates in research and industry. Then, in Sect. 3, we gave a brief overview of some related works. Thereupon, the architecture and main parts of the implemented generic system for SOTA Updates is explained in Sect. 4. Thereafter, a system for variant and release management based on a dedicated database is introduced in Sect. 5. Sect. 6 includes a conclusion and a description of future work.

2 State of the Art

2.1 Traditional Software Updates for Vehicles

A system software release is usually defined for each vehicle and flashed at the end of the production line. New system releases during after sales phase are implemented depending on the product nature with different frequencies, for example once per six months. One important use case for such releases is recall campaigns conducted by OEMs after discovering a severe bug, which could influence the safety of the passengers or lead to a problem with respect to a national or an international licensing norm. In order to install these releases on the concerned target ECUs, updates are deployed traditionally in dealer workshops over cables. A sequence diagram describing the typical steps of such an update process is illustrated in **Fig. 1**. In this lengthy and costly process, the OEM must not only manage the complexity of multiple software versions, vehicle variants and configurations, but also manage the distribution to dealerships. The update is usually done by connecting through the On-Board Diagnostics

(OBD) port [9], and can take about 15 to 90 minutes [10]. This can block other garage activities during the update process, and requires special equipment, which is not always available in the workshop. For the consumers, who must visit the dealer and wait for the update to be completed, this presents an inconvenience factor, which reduces their satisfaction [10].

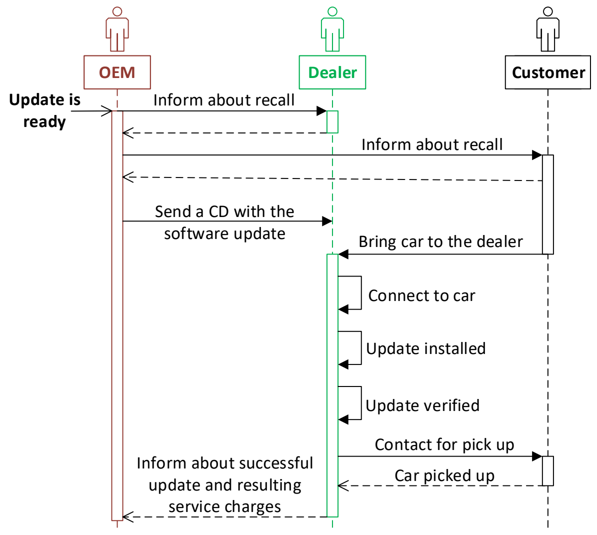

Fig. 1. Sequence diagram of a typical update's process in a dealer's workshop (according to [8])

Another possible update alternative is the customer implemented update, which is already offered by some OEMs for updating the radio/infotainment system. In this case, the customer receives a CD with the software installation files or has to manually download the software on a USB drive [8]. This approach is also lengthy and inconvenient for the customer, and induces a high risk by putting much responsibility on the customer's side. From a security point of view, it increases the possibility of reverse engineering and uploading a hacked version of the installation file by having the software in the hands of many customers [8].

2.2 Updates for Mobile Devices and Computers

Software updates are today an important part of the lifecycle management of computer systems and mobile devices such as smartphones or tablets. Diagnostic, version and configuration management, updates and other life-cycle activities need to be integrated in an efficient framework to keep an overview of all devices and their evolution. Seen the rapid spread of wireless connected devices and the success they are having in our modern society, this framework usually also supports an OTA implementation.

One of the most important mobile devices management solutions is the Open Mobile Alliance (OMA) Device Management (DM), which is practically supported by all smartphone manufacturers [11]. It defines the management information (management objects) for mobile devices in a so called DM tree. The management objects are the entities for software setting that may exist as environment variables or firmware images [12]. The management is done remotely, for example over a WLAN connection, through the interaction between the OMA DM server and the management agents using the OMA DM protocol aimed at providing remote synchronization of mobile devices.

Although many requirements for automotive SOTA updates, such as higher safety levels, distribution of functions through dozens of connected ECUs and the wider multidisciplinarity including mechanical aspects, are not included in the OMA DM standard, many aspects can be transferred and used in the automotive field.

2.3 Current Situation of Automotive SOTA Updates

SOTA updates are nowadays not only a topic of research, but also already implemented by different OEMs, like BMW, VW and Tesla for navigation maps. Total vehicles that have map OTA updates are projected to grow from approximately 1.2 million units in 2015 to nearly 32 million units by 2022 [13]. SOTA updates for Infotainment services like email or social media have also been realized.

At ECU's firmware level such as for chassis or power-train modules, SOTA is still at the very beginning of the road. These updates are very critical to the safety of the passengers and need to be secure and safe enough to avoid each unexpected ECU behavior. Only the automaker Tesla is known for offering SOTA capability for almost all domains, including updates for the Autopilot, which is the autonomous driver assistance system of the car [14]. However, the number of vehicle variants in this case is still relatively low, which makes the management of the fleet easier.

3 Related Works

Considering the significance of SOTA updates for the future of the automotive industry, different research works have been conducted in the recent years.

In [15], a computationally efficient protocol with low memory overhead for secure firmware updates over the air is presented.

Mansour et. al. introduced a system in [16] called AiroDiag for diagnosing embedded systems and updating ECU's software of vehicles through the Internet using a client-server structure.

In [17], a generic SOTA system covering safety and security aspects for an Electronic Brake Control (EBC) system has been presented. The safety and security of the vehicle as well as the availability of the car for the passengers are introduced as the main criteria for the acceptance of SOTA by legal authorities and customers.

Most of the existing works focused clearly on the security aspect of SOTA updates. The variant management challenge and its inclusion in a generic SOTA system allowing the validation of software updates for whole fleets throughout their life-cycles hasn't been studied deep enough yet. And one of the main goals of this work is to cover this gap and propose convenient methods to manage this complexity.

4 A SAFE AND SECURE SOTA SYSTEM

The developed generic system is realized using a classic Server-Client approach composed of the OEM's SOTA Server and the vehicle as the SOTA Client. The SOTA Server acts as the administration unit of the system and therefore manages the update process. The responsibilities of the SOTA Client include the reception, validation and distribution of update files to the respective ECUs within the vehicle. The features introduced in this section are based on other works and merged into a generic system.

4.1 System Overview

The SOTA Server's three main objectives are:

- update process control
- data management
- administrator / client access.

In order to perform organized SOTA updates, the system needs to keep track of software images, i.e. the binary files to be programmed on the Flash memory of ECUs, the status of updates and the amount and type of ECUs in the different vehicles. Furthermore, vehicles are grouped into fleets, so the software updates can be applied to only a specific set of cars. A fleet in this sense can contain all the vehicle variants of an OEM or only a well-defined group of variants managed by a separate department depending on the release development strategy.

Software management is key to introduce proper software version control and to prevent compatibility mismatches. Therefore, software images are characterized by their name, version and compatible ECU. An update management component is responsible for the administration of software releases. Software updates are issued for a specific ECU type and therefore create a unique relation between a software image and its matching ECU type. These updates are then combined into so-called Deployment Packages which contain all information and software images needed to update a certain type of vehicles of the selected fleet. All necessary information and data, as current software versions of cars in the field, pending software updates and validated software images, are stored in a local database within the server.

The client's and administrator's access to the SOTA server are granted through RESTful Application Programming Interfaces (APIs). These allow system administrators to manage the distributed ECUs and their software details in cars and also their affiliation to fleets, moreover to issue software updates and initiate them. All interactions with the SOTA client such as software image downloads and status reports by the vehicle are tunneled through the client API.

The SOTA client is divided into a Telematics Unit, a Central Gateway including a local database and the various ECUs. This architecture is shown in **Fig. 2**. The Telematics Unit represents the vehicle's access point and provides a mobile communication link via e.g. WiFi or cellular networks. It is also in charge of the encryption and decryption of any outgoing or incoming transmissions. In order to orchestrate SOTA updates within the vehicle, the Central Gateway's tasks include periodically checking for new updates within the server, downloading and unzipping pending Deployment Packages and the distribution to the ECUs. After a successful installation, the Central Gateway refreshes its database regarding the affected ECUs and reports it back to the SOTA Server.

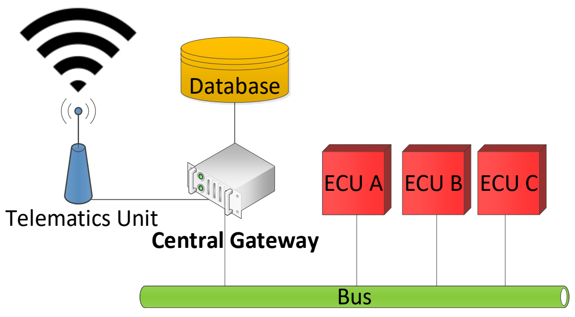

Fig. 2. SOTA Client architecture

For the implementation, a Node.js [18] server is used as the SOTA server, a Raspberry Pi 3 represents the Central Gateway as well as the Telematics Unit, and the various ECUs are realized by Texas Instruments' Tiva C LaunchPads all connected to the Central Gateway via a common Controller Area Network (CAN) bus.

4.2 Security

Due to the flexible data connectivity of a vehicle to another vehicle or an infrastructure, new risks may occur during the data transmission. As a risk can be defined as a likelihood of an attack caused by a threat, risks of unauthorized access of vehicle's software via the interconnectivity has to be predicted, and tackled by providing some additional features in the data transmissions.

The possible attacks to a SOTA system might include identity theft and manipulation or repetition of transmitted messages which may result in gaining the vehicle's control by an unauthorized third party. This encompasses intellectual property theft of software images or sensitive vehicle data. An attacker in possession of OEM server's identity could infiltrate the electronic system and install fraudulent software on ECUs of thousands of vehicles. The interception of messages that contain clear text using a man-in-the-middle attack could unveil confidential information to unauthorized users through unauthorized access.

In order to reduce the risk of a possible identity theft, a strong authentication process has to be implemented. The client's identity is verified using an open standard protocol called OAuth2 [19]. For this matter, the client retrieves an Access Token from an authorization server providing a unique and confidential secret to proof its identity. It then uses this Access Token to make an API call to the SOTA server which verifies the received token with the authorization server and sends back the requested information. The process of the server's authentication is realized using a self-signed certificate which gets transmitted to the client during the establishment of a secure communication channel using the Transport Layer Security (TLS) handshake. The client then verifies the integrity of that certificate with a Certificate Authority.

The confidentiality of the data exchanged between the SOTA server and client is secured using reliable symmetric encryption methods. The greatest risk that endangers the integrity of any data that is being sent draws from the disclosure of the keys used for the de- and encryption procedures. Therefore, these keys need to be stored securely all the time and may only be transmitted via a secured connection established by a TLS handshake. For this, the client asks the server about which TLS version is intended to be used and about its known cipher algorithms. The server answers providing the chosen cipher algorithm from the selection, its public key and certificate. The client then verifies the certificate with the Certificate Authority. Furthermore, it generates the session key which will later be used for the symmetric encryption, encrypts it with the server's public key and sends it back. Being the owner of the respective private key, the server then decrypts the session key and acknowledges its reception. From then on, the session key is used for both encryption and decryption of any messages transmitted. An overview of methods and issues related to encryption key management can be found for example in [20].

Besides, it is important to insure the freshness of data in order to prevent unauthorized third parties from interfering with the communication by intercepting messages and sending them again at a later point in time. This risk has been prevented by appending a consecutive package number to every message.

Since security was not the main focus of this work, further research dealing with other security topics such as public keys management and certificate revoking should be carried out. As a general framework for prototyping and developing Cybersecurity in connected vehicles, the standard SAE J3061, published in January 2016 by SAE International and described for example in [21], could be used as a reference.

4.3 In-Vehicle Update Strategy

The software architecture of ECUs is designed to separate the bootloader from the actual application according to the AUTOSAR standard. This allows for the maintainability of the application while the bootloader is left intact and operational. As it comes to update strategies, there are three decisions to be made which each has an impact on the cost, design, time consumption and feature capabilities of SOTA updates. **Fig. 3** shows the resulting main realization alternatives.

SOTA updates for ECUs can either be performed using Full Binary or Delta Updates. For Full Binary Updates, the application is being replaced entirely, whereas for Delta Updates only the application parts that differ from the new version are being edited. The implemented system uses for simplification the Full Binary approach.

Another conceptual choice that needs to be made is whether an incoming software update should replace its predecessor in its memory location (Replace Approach) or should be saved in a separate memory block to which the program control is shifted, thus being called A/B Swap. Despite doubling the memory space and therefore increasing the hardware cost of the ECU, the A/B Swap enables the functionality of rollbacks, i.e. shifting back to the old software version in case of an update's failure.

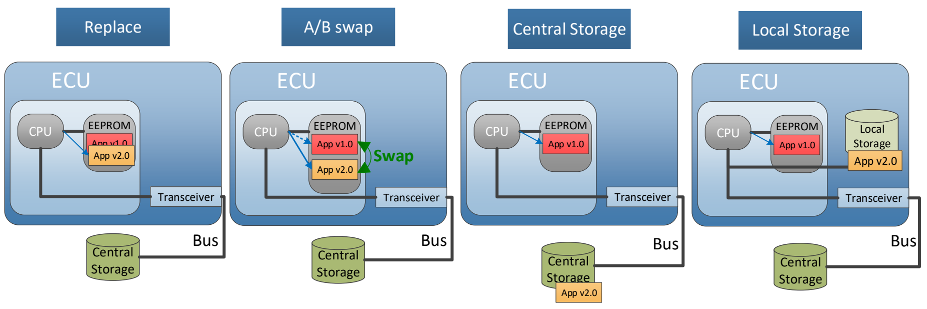

Fig. 3. Update Strategies (according to [17])

In addition, the placement of the storage device, from which the new software image is retrieved, must be negotiated. Using only a central storage comes with the advantage that no changes have to be applied to the ECU's design, but will also result in long vehicle downtimes in the case of multiple software updates coming in at once. Software images can be retrieved much quicker from a local storage, which also enables the system to perform multiple updates in parallel as the required files can be transmitted while the car is in operation.

The resulting solutions are compared in **Table 1** considering: vehicle downtime, extra cost, rollback time and necessary ECU design changes.

Table 1. COMPARISON OF THE THREE UPDATE APPROACHES

	Down Time	**Extra Cost**	**Rollback Time**	**ECU Design Changes**
Replace & Central Storage	long	minor	long	none
A/B & Central Storage	long	medium	very short	minor
Replace & Local Storage	short	medium	medium	medium

The A/B Swap with a central storage causes additional hardware costs because the size of internal flash memory of the microcontroller has to be doubled and also results in a long downtime of the vehicle during the update process due to the fact that software images can only be transmitted to the ECUs one at a time. Nevertheless, the downtime can be reduced by using a faster bus system, e.g. Ethernet, or running the ECU applications from an external flash memory. This creates the possibility to perform simultaneous updates of multiple ECUs. The big advantages this approach has over the Replace Approach with a local storage, are the significantly shorter rollback time and the fact that only minor design changes have to be applied to the ECUs. Because of these reasons, it has been adopted in the system of this paper.

4.4 Rollback Function

Establishing automotive SOTA updates will increase the frequency of software upgrades being rolled out to vehicles and therefore elevate the error probability during the update process. The option to recover from an ECU deadlock due to a failed update is in this case a very important safety feature. It can be realized through a Rollback function.

One of two possibilities to perform a Rollback is to flash a previous version of the application running on the ECU back to its program memory from either a central or local storage device. Depending on the location of the storage device, this procedure may take several seconds or even minutes to finish for one ECU.

A second option is to reverse an already executed A/B Swap and shift the program control back to block A.

5 Variant and Release Management

5.1 Variant Management in the Automotive Field

Through the offered freedom to configure vehicles according to the customer needs and wishes, a huge variant space arises comprising millions of unique products with differences at various granularity levels. This leads to a high degree of complexity inside automotive product lines. Within each vehicle model, the customer has normally the choice between different equipment variants. For each equipment variant, there is the possibility to choose different kinds of motors, gearboxes and bodies. We named these first variability parameters Variants Features, as shown in **Fig. 4**. Then the numerous ECU functions such as airbag, radio or driver assistance systems can be defined as further configuration features, which increase the variants number considerably.

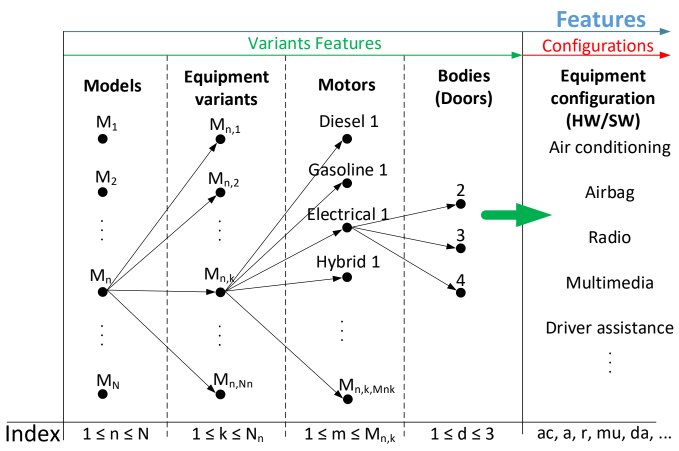

Fig. 4. Features determining the vehicle variant

Conducting an evaluation of the number of variants of e.g. Volkswagen cars using the online model configurator [22] and considering only the variants features of **Fig. 4**, we found a sum of 1164 variants. By including the ECU configuration features, this number will increase exponentially. By introducing optional SOTA updates, another variability dimension in time is added to the already existing variant space. This means that the same software modules can exist for the same product with different versions depending on whether or not optional updates have been installed.

One of the most used notations to model variability is the feature-based representation, similar to the model in **Fig. 4** [23]. It documents the features of a product line and their relationships and specifies the set of valid products.

5.2 Realization of Variant and Release Management

At the back-end (server) side of the system, each vehicle is defined by a unique ID differentiating it from all other vehicles. All its relevant management information is saved according to the feature-based variant modeling in a structured database, which has been implemented using the SQL language. This data structure serves as the reference for deploying the updates into the fleet and keeping diagnostic information up-to-date. It should also allow for the management of the development and validation of SOTA updates under consideration of the existing variant space. This can be achieved by traceability implementation to the static and dynamic system models at different abstraction levels.

At the beginning of a new SOTA release, an administrator selects a fleet and the software updates to be distributed. In the next step, the management system creates a Deployment Package for every unique vehicle configuration of the fleet in question. The Deployment Package contains all necessary software images for a particular vehicle configuration. The selection of the relevant software images is made based on a comparison of two subsets of ECU types and their resulting cut set. One subset contains the ECU Types of the vehicle configuration present in the fleet in question. The other subset contains the ECU Types to which the selected software updates are compatible to. To track the progress of the installation of the Deployment Packages on every vehicle of the fleet, the management system creates a Deployment entity for every car that receives a Deployment Package and an Update entity for every ECU that is being supplied with a new software image. This management system based on the implemented database is represented is **Fig. 5**. With this procedure, the OEM can manage the evolution of its fleet and the distribution of the Deployment Packages derived from the main software ECU updates.

6 CONCLUSION

After studying the state of current automotive updates, such as during global recall actions, the main benefits as well as the challenges of SOTA could be identified. The state of the art of common SOTA updates for mobile devices and computers as well as the already existing first SOTA systems in the automotive industry have also been described. Besides, some of the related works have been mentioned and the boundary to this work has been explained.

A generic SOTA system has been designed and implemented in a proof of concept framework. Security mechanisms, different update strategies combined with various rollback function realizations have been introduced as main characteristics of the system. The needed resources and structures for efficient variant and release management have been included as principal part of the generic system.

Due to the generic aspect of this work and the diversity of the involved parts in the system, different subjects will be further investigated in future works. Experimental work to validate the release management procedure for realistic update use cases will be carried out. In addition, the variant management of software updates for distributed functions should be deeper investigated and validated by considering the dependencies between the involved ECUs. Consistency checks, simulation and virtualization techniques need to get involved in a concrete validation scenario, where dependencies require different update versions between various variant models.

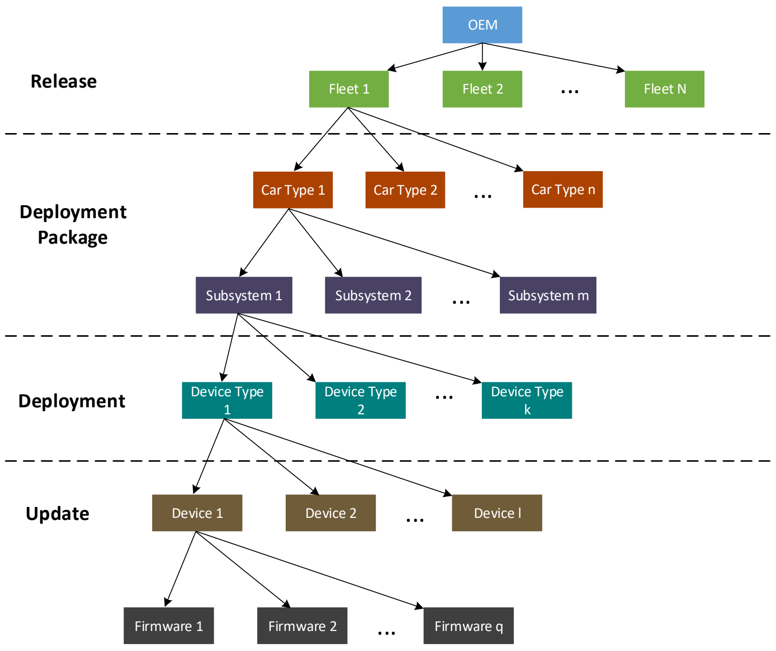

Fig. 5. Data Structure for variant and release management

References

1. M. Staron, Automotive Software Architectures: An Introduction, 1st ed. Springer Publishing Company, Incorporated, 2017.
2. C. Hobbs, Embedded Software Development for Safety-Critical Systems, Boston, MA, USA: Auerbach Publications, 2015.
3. E. Sax, „Automatisiertes Testen Eingebetteter Systeme in der Automobilindustrie“, Hanser-Verlag (ISBN 978-3-446-41635-2), 2008.
4. M. Khurram, H. Kumar, A. Chandak, V. Sarwade, N. Arora, and T. Quach, “Enhancing connected car adoption: Security and over the air update framework,” in 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), Dec 2016, pp. 194–198.

5. D. K. Nilsson, L. Sun, and T. Nakajima, “A framework for selfverification of firmware updates over the air in vehicle ecus,” in 2008 IEEE Globecom Workshops, Nov 2008, pp. 1–5.
6. E. Bird, Colin; Juliussen, “Improving software, reliability & innovation - executive summary,” IHS Technology, Tech. Rep., 2015.
7. E. Sax, R. Reussner, H. Guissouma, and H. Klare, “A survey on the state and future of automotive software release and configuration management,” KIT, Tech. Rep., November 2017.
8. H. Dakroub and R. Cadena, “Analysis of software update in connected vehicles,” SAE International Journal of Passenger Cars – Electronic and Electrical Systems, vol. 7, no. 2, pp. 411–417, 2014. [Online]. Available: https://doi.org/10.4271/2014-01-0256
9. E. Els, “The hackers holy grail - the obd has manufacturers worried,” Automotive Diagnostic Systems, June 2017.
10. H. A. Odat and S. Ganesan, “Firmware over the air for automotive, fotamotive,” in IEEE International Conference on Electro/Information Technology, June 2014, pp. 130–139.
11. L. Liu, R. Moulic, and D. Shea, “Cloud service portal for mobile device management,” in 2010 IEEE 7th International Conference on E-Business Engineering, Nov 2010, pp. 474–478.
12. J. Shin, Y. Chung, K. S. Ko, and Y. I. Eom, “Design and implementation of the management agent for mobile devices based on oma dm,” in Proceedings of the 2Nd International Conference on Ubiquitous Information Management and Communication, ser. ICUIMC ’08. ACM, 2008, pp. 575–579.
13. M. Culver, “Over-the-air software updates to create boon for automotive market, ihs says,” IHS Automotive, September 2015.
14. Tesla, “Software updates,” 2017. [Online]. Available: https://www.tesla.com/software
15. D. K. Nilsson and U. E. Larson, “Secure firmware updates over the air in intelligent vehicles,” in ICC Workshops - 2008 IEEE International Conference on Communications Workshops, May 2008, pp. 380–384.
16. K. Mansour, W. Farag, and M. ElHelw, “Airodiag: A sophisticated tool that diagnoses and updates vehicles software over air,” in 2012 IEEE International Electric Vehicle Conference, March 2012, pp. 1–7.
17. A. Freiwald and G. Hwang, “Safe and secure software updates over the air for electronic brake control systems,” SAE International Journal of Passenger Cars - Electronic and Electrical Systems, vol. 10, no. 1, pp. 71–82, sep 2016.
18. S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," in *IEEE Internet Computing*, vol. 14, no. 6, pp. 80-83, Nov.-Dec. 2010. doi: 10.1109/MIC.2010.145
19. Internet Engineering Task Force (IETF), “OAuth 2.0 Authorization Framework” May 2018. [Online]. Available: https://tools.ietf.org/html/rfc6749
20. D. W. K. Tse, D. Chen, Q. Liu, F. Wang, and Z. Wei, “Emerging issues in cloud storage security: Encryption, key management, data redundancy, trust mechanism,” in Multidisciplinary Social Networks Research, L. S.- L. Wang, J. J. June, C.-H. Lee, K. Okuhara, and H.-C. Yang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.
21. C. Schmittner, Z. Ma, C. Reyes, O. Dillinger, P. Puschner, „Using SAE J3061 for Automotive Security Requirement Engineering“
22. Volkswagen, “Online configuratro,” February 2016. [Online]. Available: https://www.volkswagen.de/app/konfigurator/vw-de/de
23. T. Berger, R. Rublack, D. Nair, J. M. Atlee, M. Becker, K. Czarnecki,and A. Wasowski, “A survey of variability modeling in industrial practice,” in Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems, 2013.