Challenges of adopting DevOps in automotive software development

University of Oulu
Department of Information Processing Science
Bachelor's Thesis
Teemu Risikko
Instructor: Teemu Karvonen
06.06.2020

Abstract

DevOps is a practice of that brings developers and operations together, and it is enabled by rigorous automation, improvement and monitoring. It is beneficial for a company, because it speeds up the delivery of new software updates to customers. However, automotive software field has not yet taken the practices in use.

In this study, several different challenges of adopting DevOps-practices in automotive software development projects were identified. Previous research about both DevOps and automotive software, existed on the time of writing, but studies combining these two topics were rare.

The study concluded that since automotive software development is traditionally highly complex and distributed, most of the benefits of DevOps are not seen in the field. The study focused on both social and technical challenges seen in adopting DevOps for automotive project. Collaboration and automation were identified as the main problems that need addressing before an automotive company can shift to a DevOps-style development mode.

Collaboration problems were explained to arise from the way how automotive software projects are distributed. Traditionally, automotive manufacturing projects are split to multiple sites to allow efficient production of different parts, but it became evident that that future automotive projects are much more software intensive. This in turn requires more system thinking, which does not match with the distributed development model.

Coordination of different suppliers becomes hard when experts are rare, and time is divided between multiple projects. Little time is allocated to self-development, which makes the problem more severe. Instead of cross-functional development teams, automotive software projects were noticed to be split into so called silos, where knowledge is not spread between the silos easily.

Automation was confirmed to be a major problem for the field, mostly because low abstraction level in automotive software. Most of the code is directly written to be hardware specific and reusing old solutions for new problems was noted to be impractical. Testing the implementation is costly, because often proper testing needs to be executed on special hardware, that might not be available for the organizations that provide the software.

Monitoring of automotive software was not seen as important by the manufacturers, which makes adopting DevOps style continuous delivery useless, since no new knowledge can be easily derived from software updates that are not monitored afterwards.

Significant changes to the development methodologies are needed before any DevOps adoption can occur in automotive software projects.

Keywords

Automotive, DevOps, Continuous Integration, Continuous Delivery

Contents

1. Introduction

Automotive industry is ongoing a transformation where e-mobility, connected cars and autonomous driving are changing the landscape of the automotive software development. Traditional silos and long supply changes need breaking up, and this is where the new standards of software development can help car makers to gain an advantage.

DevOps, or developers and operations, is a phenomenon that aims to speed up the delivery of software from companies to customers. This is done mainly by fostering effective collaboration, automation, measurement and monitoring (Lucy Ellen Lwakatare, Kuvaja, & Oivo, 2015). Development and operations organizations are brought closer to each other with agile methodologies, lean processes, extensive automation and effective support structures.

DevOps-practices can help automotive industry to speed up their delivery and increase software quality, where OEMs (Original Equipment Manufacturers) are expecting shorter development lifecycles and more agile way of working. However, the domain is bound to have its own special challenges and quirks when adopting DevOps-practices.

In this study, these challenges will be investigated, and the purpose is to identify from academic literature the specific features that make automotive DevOps challenging for the companies trying to practice it. Prior research has been done on both the automotive software development and DevOps in general, but there is a lack of a comprehensive study on automotive grade DevOps. This motivation for this study is to shed more light on why DevOps-methodologies are not often associated with automotive software development.

1.1 Research problem and research methods

The research problem that this study will address is as follows:

What challenges are characteristic for the automotive domain when adopting DevOps-principles?

The research method is literature review, so the study is based on prior scientific research. The purpose is to identify the specific challenges that are especially predominant for automotive field compared to other software development.

Reference papers are selected from peer reviewed scientific papers from online catalogues suggested by University of Oulu, based by their relevance to the topic. Articles containing the keyword “automotive” combined to DevOps-related keywords - like “agile”, “automation”, “Continuous Integration” and “Continuous Delivery” - are selected for the review. Some online sources and books are mentioned to shortly explain some of the key topics of state-of-art automotive and DevOps, but these are linked to actual research done on them.

DevOps itself is a trending topic in research. At the time of writing, there are over 500 results in Scopus with the keyword “DevOps”. However, only a few exist that mention DevOps as a part of automotive software development. The topic is important for research, because automotive field is currently ongoing a transformation where software development plays a critical role to success. This study brings together different sources

to explain why DevOps in automotive is rarely mentioned, acting as a basis for future research on how these challenges could be mitigated.

2. Research background

This chapter introduces previous research about both DevOps and automotive. The aim is to get familiar with different interpretations of what DevOps is, and to generalize what is characteristic for automotive software development.

2.1 DevOps in a nutshell

DevOps in general is not a well-established term in academic research (Lucy Ellen Lwakatare et al., 2015). Michael Hüttermann explains the origins of DevOps in his book "DevOps for Developers". According to him, DevOps is a way to expand the ideas of agile development to operations and even further, to the delivery process as whole. The DevOps-movement has emerged from the conflict between developers and operations. In short, traditional operation teams want the environments to be as stable and unchanging as possible, while development teams want to make changes as often as possible. To allow breaking the conflicts between these "silos", a different approach is needed. (Hüttermann, 2012).

Ebert et al. (2016) state that DevOps is "about fast, flexible development and provisioning business processes". According to them, DevOps brings together development and operations via automation, which enables cross-functional teams to develop, deploy and monitor small chunks of software ("microservices") independently (Ebert, Gallardo, Hernantes, & Serrano, 2016).

Another possible way to define the term is in the form of a conceptual framework that describes the different elements that are characteristic for it: collaboration, automation, measurement and monitoring. (Lucy Ellen Lwakatare et al., 2015).

A similar approach is the CALMS-model. As explained by Caprarelli, Di Nitto and Tamburri (2020), this model consists of the following principles:

- Culture of collaboration between development and operations
- Automation, added to most parts of development chain
- Lean – continuous improvement of work
- Measurement for assisting decision making
- Sharing between people

They also add one additional point: Quality Assurance.

According to the study, DevOps is largely based on the methodologies of Continuous Integration and Continuous Delivery (CI/CD). These practices are about integrating and delivering code changes as often as possible. CI/CD is possible through extensive automation of building and quality assurance. (Caprarelli, Di Nitto, & Tamburri, 2020).

A modification of the CALMS-model is CALMR-model of Scaled Agile Framework (SAFe). This framework, used for multi-team agile software development, defines another aspect: recovery. SAFe emphasizes the need for low-risk deployments, made possible by having a strategy for fast recovery. The environment should support an easy rollback in case of a failure, and the teams should prepare for applying these strategies. ("DevOps - Scaled Agile Framework," 2018).

2.2 Automotive software development

Pernstål, Magazinius and Gorschek (2012) study the interface between software product development and manufacturing in automotive systems. According to them, the development usually follows so called "V-model", which splits the design into several different abstraction layers, each of them having a corresponding layer of software development processes. However, this model still resembles very closely a waterfall model, due to several gates during the software delivery process.

2.2.1 Automotive value chains

Broy, Krüger, Pretschner, & Salzmann (2007) explain the basics of automotive development process in their paper *"Engineering Automotive Software"*. They explain that automotive industry has over the years become very modular. Different suppliers do most of the engineering, while the original OEM focuses on integrating the various parts together under a brand. Thus, the whole value chain can be distributed both geographically and organizationally.

Traditionally the OEM has been mainly assembling the parts delivered by the supply chain. However, the complexity of automotive software makes each part less independent, as communication is required between different systems, forcing the OEM to take more responsibility on the actual integration. (Broy et al., 2007).

In an article about named *"Future Automotive Architecture and the Impact of IT Trends"*, the authors talk about several upcoming paradigm shifts in the automotive business, dubbing new cars as "distributed IT systems". (Traub, Maier, & Barbehon, 2017).

2.2.2 Characteristics of automotive software

According to Broy (2006), the requirements for automotive-grade software are complex. An on-board software needs to be real time critical and consider different usage and maintenance situations. Also, a wide range of functions is expected, from safety critical to infotainment. (Broy, 2006).

Ebert & Favaro (2017) identify automotive software as "one of the largest and most complex in all industries", caused by the number of different microcontrollers (ECUs, electronic control units), which ranges from 50 to 120. These ECU:s are connected within the car, and often also to external cloud services. (Ebert & Favaro, 2017).

Amount of code in cars ranges from 10 million lines (Broy, 2006) to 100 million, and is increasing still (Ebert & Favaro, 2017), touching thousands of different functions in a car. However, due to hardware specific requirements, only 10% of code can be reused between car-generations. The software is often highly optimized for a specific platform, making reuse cumbersome. (Broy, 2006).

Reuse of software is also identified as one of the key topics in concern, based on the number of academic literature available. This challenge stems from the market need for high level of customization and tailoring, which generates complexity to the development process. This is accentuated by poor strategies for managing variability. (Haghighatkhah, Oivo, Banijamali, & Kuvaja, 2017).

Knauss et al. explain that the problem of variance derives from the fact that the same platform is usually used for several models, with a range of different variants for between

regions. The study emphasizes the importance of architectural work and the tools and processes used for it. (Knauss et al., 2016).

2.2.3 Standardization

ASPICE (**A**utomotive **S**oftware **P**rocess **I**mprovement and **C**apability d**E**termination) is a standard for developing embedded automotive systems (VDA QMC Working Group 13 / Automotive SIG, 2017). ASPICE is created to tailor previous process improvement models to match the characteristics for automotive software development. A further goal is to define a common standard between European car makers to avoid overhead for the suppliers from following several different standards required by different OEMs. (Fabbrini, Fusani, Lami, Sivera, & Sivera, 2007).

According to Charles Murphy, ASPICE focuses on software engineering practices from a process management point of view. It has multiple maturity levels, and automotive OEMs may require the specification to be met on a certain maturity level. (Murphy, 2019).

A more technical point of view is defined for example AUTOSAR (**Aut**omotive **O**pen **S**ystem **A**rchitecture). AUTOSAR with its different variants focuses on defining common interfaces for automotive systems, allowing reusable development of functions within a vehicle network. It allows suppliers to focus on function-based design instead of focusing on ECUs. ("AUTOSAR - What is the goal of AUTOSAR," 2020).

Stefan Voget (2010) explains that AUTOSAR includes all the functions of a car, but it can be implemented also partially. It provides an abstraction layer on top of hardware, called Virtual Functional Bus (VFB), that connects the software and the hardware. This abstraction layer between software and electronic components makes a software unit independent from hardware, which helps on configuring the same software for different variants significantly. (Voget, 2010).

2.2.4 Quality assurance and tooling

Knauss et. al criticize the number of different tools used for similar purposes, claiming that most of the tools are at the same time very complex, but still cannot provide all the needed functionality for the whole architectural process. This leads to an inefficient usage of multiple complex tools, making the data transfer between them often a cumbersome tasks, limiting especially continuous integration activities. (Knauss et al., 2016).

According to Vöst and Wagner, the integration testing stage for automotive software is performed with ECUs, that test the interfaces introduced in the code with actual hardware components. If these tests are passed, the pipeline continues to a test farm, where several test targets are connected to the CI-system to allow the execution of functional integration tests in parallel. Due to real-time constraints, the execution of these tests is time-consuming. (Vöst & Wagner, 2016).

Design and configuration tools exist for developing AUTOSAR applications. Eclipse-based ARTOP (**A**UTOSAR **TO**ol **P**latform) is designed to provide initial functionality to implement AUTOSAR systems, allowing for a seamless development experience without needing to export data between different tools. The area in general provides room for new innovation and software. (Voget, 2010).

2.2.5 Working context

Katumba and Knauss identify heavy workloads for people working in automotive software development. Several different duties are assigned for a single person, which generates a lot of switching between contexts and unbalanced workloads. There is often no time for learning and improvement. Moreover, because of this multitasking, the full context of the work done is not known, which leads to separations of concern and lack of a holistic view of the system under development. (Katumba & Knauss, 2014).

Traub et al. describe the current situation in automotive software development as “heterogeneous”. Different silos use a wide range of tools and processes for both design and development. They expect the situation to change in the future with more alignment in the methods and tools used, to create a seamless workflow for the whole development process. Service oriented architecture, Continuous Integration, agile processes and collaboration are the key principles to achieve this. (Traub et al., 2017).

3. Findings

This section will compare automotive software development characteristics to general challenges of DevOps-adoption. The findings related to the research question are categorized to social and technical challenges, the same way as for example Claps et al. (2015) has done. Each sub-chapter adds findings about challenges when adopting automotive DevOps and identifies the factors that make these challenges especially relevant for automotive software development.

3.1 Social challenges

Hamunen (2016) notes that DevOps itself might not be known by development teams and management, and the motivation behind it requires practical training and convincing for both management and executives to enable efficient communication and sharing between departments. Each party affected by the transformation needs to be aware of the changes needed, or the adoption has a high chance of failing.

Resistance to change is also a popular theme in the sources listed by Hamunen. This includes managerial resistance in situations where managers are not aware of the problems that DevOps is set to solve, and developer resistance to changing their way of working and responsibilities, especially related to software testing. (Hamunen, 2016).

Caprarelli et al. identify different fallacies when trying to adopt DevOps in an organization. The common themes in these fallacies are bottlenecks and human errors. The case-study uses an example of a "Delivery and Operations -team", where people spend most of their time deploying new services, instead of focusing on the quality of deliveries and the system. Human errors are also often introduced because some deployments are done manually. (Caprarelli et al., 2020).

3.1.1 Working in silos

Lwakatare et al. (2016) study challenges that are prevalent for the embedded software domain, which also includes automotive software. In embedded domain, team structure is focused around specialized low-level module teams. For a successful DevOps adoption, cross-functional feature teams are a key factor. Focusing development around module teams leads to people working in silos, which is counterintuitive for the culture of continuous improvement. This is also confirmed by Caprarelli et al. (2020), who suggest that these silos need to be broken down in order to do DevOps successfully.

For a company with a multitude of departments interacting with each other with complex processes, adoption DevOps principles to reduce lead times is hard. In large and complex organizations, DevOps transformation can fall flat if a right strategy is not applied. Especially DevOps as a Service -type of strategy can become an antipattern that can maintain the organization silos. (Díaz, Perez, Yague, Villegas, & de Antona, 2019).

The chaotic situation with heterogenous methods and tools described by Traub et al. (2017) is counterintuitive to the collaboration required from DevOps-style development. As Díaz et al. (2019) mention, this is problem in DevOps adoption in general. However, as described by Broy et al. (2007), the distributed manner of automotive software development and the length of the supply chain multiply this problem, making it a characteristic challenge for the domain.

Van der Valk et al. (2018) state in their pilot project study that a holistic overview of a software project is missing, even though all the information would be available. According to Broy, this is typical for automotive software industry, since in the past cars have been manufactured in a very modular fashion, but complex software systems in cars require coordination between all different functions, emphasizing the need for systems engineering (Broy, 2006).

The lack of cross-functional teams is characteristic challenge for automotive DevOps.

3.1.2 Customer processes

Even if DevOps-style deployments would be technically possible, the customer's way of working may prevent applying it. As noticed by Lwakatare et al. (2016), working embedded systems are often critical to their daily work, and continuous updates for them are risky. Usually updates are done only for improving availability, as an update usually requires a full shutdown of a system. This is typical for the embedded domain of software development.

According to Broy et. al (2007), OEMs have a strategy of marketing cars as a complete "package", neglecting the need for regular software maintenance. This leads to a situation where no new software is ever delivered to the car, which makes adoption of the whole DevOps methodologies somewhat irrelevant, since they do not bring value in maintenance phase.

Using the V-model to execute automotive software projects is close to a traditional waterfall model, as mentioned by Pernstål et al. (2012). L E Lwakatare et al. (2016) mention that adopting DevOps methodologies helps to improve communication and collaboration between development teams that have previously used waterfall model, but this does not scale to customer communication due to long deployment cycles. The same methodologies need to be adopted by all in the development chain to make it effective.

3.1.3 Sharing knowledge between people and organizations

According to the CALMS-model, sharing knowledge and processes between people is an important aspect for successful DevOps-practices (Caprarelli et al., 2020). In automotive software development, this is especially important, since a software designer needs to have understanding of not only software development processes, but also car manufacturing processes (Pernstål et al., 2012).

Ebert & Favaro (2017) predict that the growth of automotive software development is increasing, and due to the nature of automotive projects, a wide understanding of both IT-systems and embedded development is needed. This requires companies to train their developers more, since formal education about automotive development is rare. (Ebert & Favaro, 2017).

However, as Pernstål et al. mention, the spread of competence varies significantly between different parties. In the study they suggest that manufacturing engineers should be much more involved in the software development process. But as Katumba & Knauss (2014) notice, there often isn't time for self-improvement, and especially more experienced developers don't have time to share their knowledge due to a high workload and splitting between projects.

Even transparency between different automotive suppliers is low. The longer the supply chain, the less transparency is seen in communication. Coordination mechanisms between OEMs and software development teams are not up-to-date, leading to unclear requirements for the software to be delivered. (Marner, Theobald, & Wagner, 2019).

The requirements also change often, leading to a situation where the developers are not up to date with the latest requirements from the OEM. This is due to the geographical distance between suppliers, that makes coordination between organizations even more cumbersome, as identified by Broy et al. (2007).

The geographical and organizational distance between manufacturing and software development makes sharing of the required knowledge challenging. To adopt DevOps with the current state of automotive software development, making this information available to the developers is essential to ensure a smooth flow of software delivery to the actual cars.

3.2 Technical challenges

Claps, Berntsson Svensson and Aurum (2015) mention DevOps in their study about technical and social challenges when adopting Continuous Delivery. They identify a successful strategy for CD a critical part of DevOps transformation.

Claps et al. list multiple technical challenges that need mitigation when implementing a Continuous Deployment strategy: the infrastructure, version control and configuration management need to be in order to successfully mitigate the risks. For example, a larger amount of hardware resources is needed for a Continuous Integration system, to allow development in small patches and thorough testing. (Claps, Berntsson Svensson, & Aurum, 2015).

3.2.1 Complexity of testing tools and environments

Hamunen (2016) finds automating especially testing and tooling complicated. DevOps reduces the need for manual testing and in turn requires more automated testing. Methods and frameworks for test automation might not be known by the development team, as testing is traditionally done by a QA-department.

Finding and configuring the correct tools for automation is seen as a burden. There are multitude of tools and choosing the correct one for the job requires analysis. Some tools might not support the needed functionalities and others may become irrelevant too soon. The choice between using open-source solutions and proprietary software, and the choice between cloud-based and locally hosted software are also key decisions in the DevOps-adoption process. (Hamunen, 2016).

The problem of test automation is seen in automotive software, as described by Lwakatare et al. (2016): especially acceptance stage testing for embedded devices is often executed after the actual development. Configuring these test environments for continuous delivery is complex, and often not possible. This causes bugs to stay undiscovered until a manual testing stage is executed. This problem is named as “testing latency” by Caprarelli et al., meaning that there is too much delay between the development and testing to be able to work in DevOps mode.

These testing problems are prevalent in automotive: actual ECU hardware or target machines are often needed for performing the required testing, as described by Vöst and

Wagner (2016). Because of this, mitigating the infrastructural risks by providing more hardware for a Continuous Integration setup, as suggested by Claps et al., might become costly or not feasible.

In addition, the variability challenge mentioned by Knauss et al. (2016) makes reuse of hardware for testing purposes challenging. Since code cannot be easily reused between variants, specialized hardware is required for testing each different configuration, which makes executing Continuous Integration pipelines even harder.

The scattering of actual hardware between different suppliers complicates also quality assurance. As Broy et al. (2007) describe, automotive components are tied together with software that collaborates with multiple different components. Getting the required hardware and executing the complete target testing for a functionality is needed from the OEM side, pushing testing back to a very late stage in the development.

Haghighatkhah et al. (2017) propose that this issue should be mitigated by using virtualized platforms for integration and testing. For this, abstraction layers over the actual ECUs are needed. However, there platforms have their own complexities that make integration difficult.

3.2.2 Complexity of standards

Standards like AUTOSAR are a necessary abstraction layer to enable the level of testing and automation for automotive software development that DevOps requires. However, AUTOSAR itself generates a lot of complexity. AUTOSAR is not compliant with parallel development and modelling (Voget, 2010), which is against the principle of flexible development defined by Ebert et al. (2016).

AUTOSAR also requires complex toolchains, and in the modelling process, data needs to be exchanged several times between different tools. Even tools like ARTOP, which provides several useful functionalities for AUTOSAR development, cannot enable a smooth pipeline from requirements to executable and beyond. (Voget, 2010). This problem with toolchains and data transfers is also confirmed by Knauss et al. (2016).

The complex toolchain problem is seen especially in the requirements phase of an automotive software project. A clear and reusable toolchain does not exist, leading to poor automation and redundant tools. At the same time, traceability from requirements to the actual software is needed. To combat this issue, a more consistent process is needed. (Haghighatkhah et al., 2017).

When a clear and reliable toolchain from requirements to delivery does not exist, Continuous Integration and Continuous Delivery pipelines associated with DevOps mentioned by Claps et al. (2015) are bound to become complex to develop, maintain and automate. Furthermore, if tools need to be changed between different software projects, reusing previous solutions in other projects cannot be done. The same effort for automation is required at the start of each automotive software project.

3.2.3 Lack of monitoring

Lwakatare et al. identified that monitoring the usage of software after they deployment to target hardware is usually not executed continuously and is often done only during maintenance or failure analysis. Even if the OEM would have the data available, software companies don’t have access to it by default.

Furthermore, diagnosing software errors and recovering from software failures is lacking in cars. Software errors are often treated by replacing the whole hardware in question, because no detailed error diagnostics can be performed. Broy et al. (2007).

The unavailability of monitoring data about usage and errors situations complicates the measurement and monitoring part of DevOps, that is defined by multiple authors to be one of the key parts of DevOps. Without this data, updating and maintaining existing automotive software cannot be done easily, because the state of the current software is not explicit, and no development decision can be done based on it.

4. Discussion

Most of the challenges listed in this study seem to stem from the inflexibility of either the automotive suppliers or the architecture itself. No matter what definition of DevOps is used, each area of it is impacted by these factors. The summary of the findings is listed in table 1.

Table 1. Summary of findings

Automotive characteristic	**Impact on DevOps adoption**
Silos: distributed suppliers focusing on specialized software units.	Lack of coordination and knowledge sharing.
Long supply chains.	Lack of transparency between tiers.
Software maintenance not important for OEMs.	Hard to bring development and operations together.
Costly hardware.	QA is delayed to a late phase of development.
ECU-specific software without abstraction.	High level of QA automation is costly.
Complex standards with complex toolchains.	Automation of development tools is cumbersome or impossible.
Lack of monitoring data.	Using measurements and monitoring data to assist decision making during development is impossible.

Based on these results, the distributed supply chain described by Broy et al. (2007) is the most impactful social problem. Waterfall-like software projects with distributed supply chains cause numerous problems in development, and often completely prevent iterative and incremental development required even by agile software projects. Physical and social distance between OEMs and software suppliers increases feedback time and cuts communication channels, and development is often done with delayed testing and unclear requirements.

As seen from the studies of for example L E Lwakatare et al. (2016) and Caprarelli et al., (2020), without agile development, DevOps is impossible, since new software cannot be delivered in small enough increments that would be easy to test for regression and functional correctness.

The length of supply chains and lack of coordination also makes sharing the relevant knowledge between different parties cumbersome. As a result, fostering a culture of continuous learning, improvement and sharing is often not done.

Automation is the biggest technical problem: hardware, architecture and tools used do not support enough abstraction and automation to be flexible enough for continuous delivery. Costly and limited testing targets delay QA and make it hard to scale on the level needed for highly automated testing. Abstraction layers over ECUs are a step to the right direction, but in their current state are still not mature enough to allow concurrent development on the whole chain, mostly due to the modelling processes required. Monolithic software and redundant tools do not encourage breaking the traditional V-model development. This is confirmed by multiple authors sited in this study: Voget (2010), Knauss et. al (2016) and Vöst and Wagner (2016).

Finally measuring and monitoring the software in use is limited. The latest car models are connected to cloud, but usage of the data collected is mostly limited to OEMs for maintenance purposes, and not used for updating the current software with new features. Without this data, decisions cannot be effectively done on the development level, and the steering needs to be done on OEM level.

5. Conclusions

In this study, different challenges of adopting DevOps for automotive software development were investigated. It was revealed that the even though the field is ongoing transformation to become more software oriented, many established software development best practices are either not in use or are hard to achieve due to different limitations.

The field experiences coordination and collaboration problems, inflexible development methodologies, non-abstract and complex architectures, redundant and hard-to-automate tools, and lack of visibility and measurement. These challenges are often seen together and hinder any DevOps adoption efforts significantly.

5.1 Limitations of this study

The study was based upon prior research and literature, and no empirical study was conducted as part of this bachelor's thesis.

Reference material was limited to Finnish and English sources.

5.2 Recommendations for future research

The possible already existing or upcoming solutions for these challenges were not extensively covered. Future research could be done for example on how the OEMs are currently breaking the mentioned silos, and how connected cars and over-the-air updates can change the landscape of the development. Once the abstraction layers on top of ECUs mature, research on their impact on development and test automation is possible.

References

AUTOSAR - What is the goal of AUTOSAR. (n.d.). Retrieved February 23, 2020, from https://www.autosar.org

Broy, M. (2006). Challenges in automotive software engineering. In *Proceedings of the 28th international conference on Software engineering* (pp. 33–42).

Broy, M., Krüger, I. H., Pretschner, A., & Salzmann, C. (2007). Engineering automotive software. *Proceedings of the IEEE*, *95*(2), 356–373. https://doi.org/10.1109/JPROC.2006.888386

Caprarelli, A., Di Nitto, E., & Tamburri, D. A. (2020). Fallacies and Pitfalls on the Road to DevOps: A Longitudinal Industrial Study. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12055 LNCS*, 200–210. https://doi.org/10.1007/978-3-030-39306-9_15

Claps, G. G., Berntsson Svensson, R., & Aurum, A. (2015). On the journey to continuous deployment: Technical and social challenges along the way. *Information and Software Technology*, *57*(1), 21–31. https://doi.org/10.1016/j.infsof.2014.07.009

DevOps - Scaled Agile Framework. (n.d.). Retrieved February 18, 2020, from https://www.scaledagileframework.com/devops/

Díaz, J., Perez, J. E., Yague, A., Villegas, A., & de Antona, A. (2019). DevOps in Practice – A Preliminary Analysis of Two Multinational Companies. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11915 LNCS*, 323–330. https://doi.org/10.1007/978-3-030-35333-9_23

Ebert, C., & Favaro, J. (2017). Automotive software. *IEEE Software*, (3), 33–39.

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *Ieee Software*, *33*(3), 94–100.

Fabbrini, F., Fusani, M., Lami, G., Sivera, E., & Sivera, E. (2007). A SPICE-based software supplier qualification mechanism in automotive industry. *Software Process: Improvement and Practice*, *12*(6), 523–528.

Haghighatkhah, A., Oivo, M., Banijamali, A., & Kuvaja, P. (2017). Improving the State of Automotive Software Engineering. *IEEE Software*, *34*(5), 82–86. https://doi.org/10.1109/MS.2017.3571571

Hamunen, J. (2016). Challenges in Adopting a Devops Approach to Software Development and Operations, 1–69. Retrieved from https://aaltodoc.aalto.fi/bitstream/handle/123456789/20766/hse_ethesis_14441.pdf?sequence=1&isAllowed=y

Hüttermann, M. (2012). *DevOps for developers*. Apress.

Katumba, B., & Knauss, E. (2014). Agile development in automotive software development: Challenges and opportunities. In *International Conference on*

Product-Focused Software Process Improvement (pp. 33–47).

Knauss, E., Pelliccione, P., Heldal, R., undefinedgren, M., Hellman, S., & Maniette, D. (2016). Continuous Integration Beyond the Team: A Tooling Perspective on Challenges in the Automotive Industry. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2961111.2962639

Lwakatare, L E, Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H. H., Bosch, J., & Oivo, M. (2016). Towards DevOps in the Embedded Systems Domain: Why is It So Hard? In *2016 49th Hawaii International Conference on System Sciences (HICSS)* (pp. 5437–5446). https://doi.org/10.1109/HICSS.2016.671

Lwakatare, Lucy Ellen, Kuvaja, P., & Oivo, M. (2015). Dimensions of devops. In *International conference on agile software development* (pp. 212–217).

Marner, K., Theobald, S., & Wagner, S. (2019). Real-life challenges in automotive release planning. *Proceedings of the 2019 Federated Conference on Computer Science and Information Systems, FedCSIS 2019*, *18*, 831–839. https://doi.org/10.15439/2019F326

Murphy, C. (2019). Automotive SPICE: 0-60 in No Time Flat. *IEEE Engineering Management Review*, *47*(2), 26–28. https://doi.org/10.1109/EMR.2019.2915217

Pernstål, J., Magazinius, A., & Gorschek, T. (2012). Study investigating Challenges in the Interface Between Product Development and Manufacturing in the Development of Software-Intensive Automotive Systems.

Traub, M., Maier, A., & Barbehon, K. L. (2017). Future Automotive Architecture and the Impact of IT Trends. *IEEE Software*, *34*(3), 27–32. https://doi.org/10.1109/MS.2017.69

VDA QMC Working Group 13 / Automotive SIG. (2017). Automotive SPICE Process Assessment / Reference Model.

Voget, S. (2010). AUTOSAR and the automotive tool chain. In *Proceedings of the Conference on Design, Automation and Test in Europe* (pp. 259–262).

Vöst, S., & Wagner, S. (2016). Towards Continuous Integration and Continuous Delivery in the Automotive Industry. *CoRR*, *abs/1612.0*. Retrieved from http://arxiv.org/abs/1612.04139