Lean DevOps for automotive OEMs

Master's thesis in Quality and operations management

Antonio Juric

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2021
www.chalmers.se

MASTER’S THESIS IN QUALITY AND OPERATIONS MANAGEMENT

Lean DevOps for automotive OEMs

Antonio Juric

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2021

Lean DevOps for automotive OEMs
Antonio Juric

Supervisor and examiner: Ola Benderius

Master's Thesis 2021:63
Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Department of Mechanics and Maritime Sciences
Göteborg, Sweden 2021-06-09

Lean DevOps for automotive OEMs
Master’s thesis in Quality and operations management
Antonio Juric
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

The accelerated software development has forced researchers and industries to reorginaze in terms of the software development process. One such recent philosophy in software development is DevOps. DevOps focuses on collaboration, continuous integration and continuous deployment amongst other things, with the intention to increase the cycle speed between feedback from end-user to an updated software. The intention with this research has been to investigate how a DevOps implementation can be achieved at an automotive OEM and what potential challenges and consequences there are of such implementation. The current software development process, from requirement to development was mapped with the help of three interviews with people that have a long background within the automotive industry. Based on the interviews, a approximated model was created to illustrate the challenges with the current centralized approach in the industry. The results show how such an approach is slow with little collaboration and the consequences are costly adjustments, slower speed to market and less customer satisfaction. With DevOps and OTA updates however, this process changes into being more responsive whilst additional functions can be added after the release of the car. Many OEMs in the automotive industry are dependent on their suppliers and to achieve a DevOps implementation, the OEMs must somehow integrate together with the suppliers in a CI/CD pipeline where the developers are located at different suppliers, and the OEM has an operations team that monitors the different car fleets and decides when a new release is going to happen.

The challenges that come with an integration of DevOps is how this re-organization can happen practically since the automotive industry is highly centralized and someone with enough power must be convinced that this is the right decision. Furhermore, with this integration and closer collaboration between suppliers and OEM, a new culture will emerge. Having a cross-border culture regarding DevOps and having another internal culture within the own borders can be challenging. Finally, there must also be a way to protect the different IP that belong to the suppliers. The IP is usually different algorithms in the software that is compiled and delivered to the OEM. If this protection is not achieved in the CI/CD pipeline, it will be difficult to persuade the suppliers to agree to this kind of integration. There are certainly many challenges in terms of organization but with software being the future of automotive, there should definitely be a push for increasing the software development speed. That is however only one part of the whole infrastructure and supply chain since there are other challenges outside of this research concerning more technical detail regarding the CI/CD pipeline and what is required from the servers and cars for this to be released OTA.

Key words: DevOps, automotive, software development, continuous integration, continuous deployment

Acknowledgements

I would like to thank everyone that has contributed to this research by finding interviewee candidates, participating in interviews or by sharing knowledge as well as experience. Through this work in this relatively new field, the research contributes to a better holistic view of what it takes for automotive OEMs to adapt DevOps and what to expect in the near future.

I would also like to thank my supervisor and examiner, Ola Benderius, who has been guiding me in the right direction and contributed with his long experience and expertise in this field.

Göteborg June 2021-06-09

Antonio Juric

Contents

1 Introduction

Ever since the Ford Model T was launched, automotive manufacturers have been striving to make mechanical and hardware improvements to their newly released cars. Henry Ford himself stated that "Any customer can have a car painted any colour that he wants so long as it is black" [1], showing no intent of changing anything since it was already a successful and highly standardized concept. Ever since then, the market has showed off a vast variety of color on cars, as well as extensive hardware improvements. Furthermore, the highly competitive environment has led to shrunk life-cycles of cars while variety has increased [2]. This implies a fast-paced reality where consumers are becoming more selective, changing correspondingly to the market meaning that the successful concept of yesterday may not be successful today. This is ongoing in the current market where there have been several shifts lately that original equipment manufacturers (OEMs) are challenged with. Different regulations urge OEMs to decrease the CO_2 emissions [3], which is to a high extent being done by increasing the amount of electrical vehicles produced while reducing new production of ICE cars [4]. Another shift along with the electrification of vehicles is towards a higher software focus. The car industry is currently approaching an environment where autonomous vehicles are possible to some extent [5] and the software in cars is an essential part of it. As autonomous vehicles become more advanced, the amount of software increases as well as the level of complexity. A majority of the OEMs are not ready for this transition now when the software development is being accelerated and cars become more software dependent.

The first car with a software was launched in 1977 by General Motos (GM) in one of the Oldsmobile models [6]. With the increase of software in cars since then, OEMs have been outsourcing the software engineering process to companies with more expertise in this area, while focusing on processes like manufacturing and selling cars. It is now becoming more and more important that OEMs understand the software better, since it is now possible to work with over-the-air updates (OTA) on cars, to make improvements or solve deficiencies. In the end, the need for a better understanding of the complexity of the software's interaction with the car is only increasing. This means that there is an increased need of understanding the complexity and how the software interacts with the car, therefore to rely solely on a code to execute a specific task is not enough. To illustrate the importance, a report by Counterpoint estimates that 75% of all cars shipped worldwide 2020-2025 are going to have the capacity to connect to the internet [7]. Therefore, it is easy to see that the industry, which has been focusing on physical and mechanical engineering for the past 100 years, need to be updated.

As mentioned earlier, a lot of the software that has been used in cars throughout the years has been outsourced to suppliers, where the OEMs are to a large extent kept in the dark of what is happening in the development process. Now, as the importance of software is growing and the complexity is increasing, many OEMs cannot cope with it since there is no apparent clear strategy. With the advent of electrification, autonomous and shared vehicles, this will need to change. The number of heterogeneous systems in cars will only grow, and software development will need to speed up. Indeed, to build any sort of autonomous system, an OEM must at the very least be able to do frequent software updates over-the-air for large numbers of cars and gather data on how these vehicles behave in real-world situations all over the world. In software engineering terms, a DevOps chain can be used to make this effective. A DevOps chain which is a way to to break down the walls and narrow the gap between the development and operations departments, with the goal to accelerate the software development.

1.1 A brief history of software development approaches

Before describing DevOps further, there is a need to take a step back and briefly review how the software development processes have developed over the years. During the 1970s, software was treated as any other product development project, which meant that the the stage-gate model (waterfall model) was applied. This means that the project goes through each stage getting different requirement before being approved for the next stage. This was due to the assumptions that the requirements were not going to change over time (e.g. like physically designed parts). Different approaches similar to the stage-gate model were used up until the 21st century when the so called agile development was introduced through the agile manifesto [8]. The agile manifesto was a set of principles with the goal to accelereate the software development and get away from the stage-gate approach [9]. Here, the perhaps most important principle was "responding to change over following a plan", since software is very dynamic and in constant change, meaning that it is difficult to plan for a release far in the future.

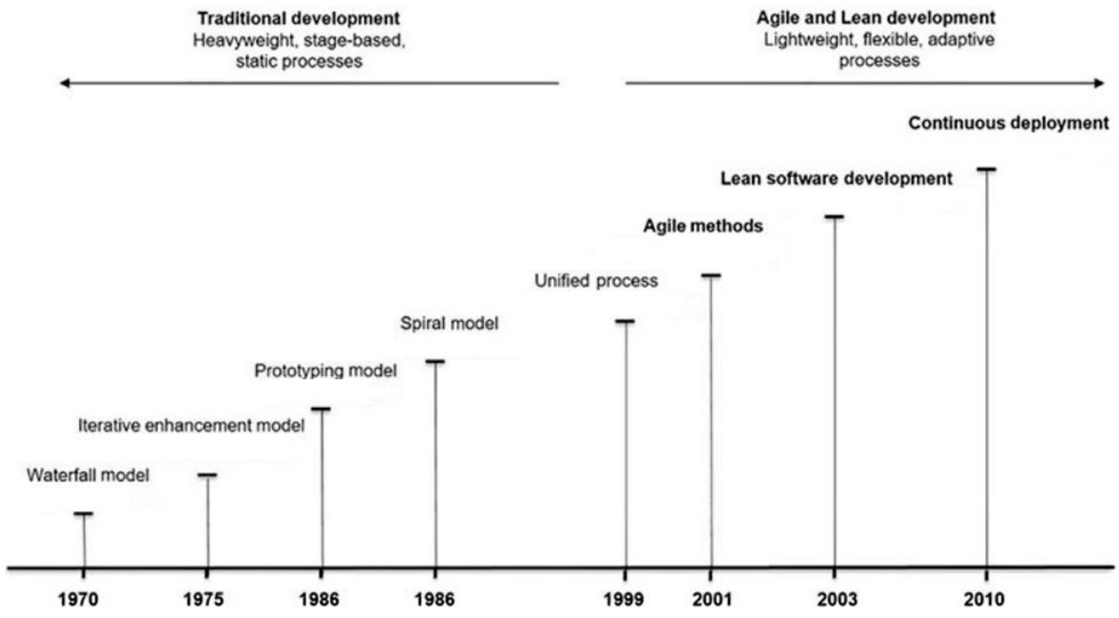

Traditional development
Heavyweight, stage-based, static processes
Agile and Lean development
Lightweight, flexible, adaptive processes
Continuous deployment
Lean software development
Agile methods
Unified process
Spiral model
Prototyping model
Iterative enhancement model
Waterfall model
1970
1975
1986
1986
1999
2001
2003
2010

Figure 1: Different project management approaches towards software development throughout the years [8].

As the agile methods were intended for small businesses and enterprises, the so called lean software development came into the picture. The lean methodology emerged within manufacturing but has also been successfully applied in several other fields. The lean software development helped to scale the agile methods so they could be applied in larger organizations. However, nowadays lean software development is not only used to scale the agile methods, but also as it has been used traditionally. This means utilizing waste elimination and and improving processes through e.g. value stream mapping. Quality is increased by testing as soon as possible, fast deliveries are done by small batches, which also reduces the work in progress (doing small task at a time) and learning is created through fast feedback [8].

1.2 What is DevOps?

The word “DevOps” is a combination of *development* and *operations*. The goal with DevOps is to accelerate the software development process and reduce the time to market by breaking down ‘silos’ within organizations and utilizing cross-functional teams [10]. The concept was introduced 2009, making it relatively new, however many of the challenges it deals with have already been identified [11]. DevOps is applied through several stages and a so called “DevOps” chain, where each phase has several tools to apply to reach the goals for that specific phase. The phases in Figure 3 illustrate how development and operations are combined into cross-functionality.

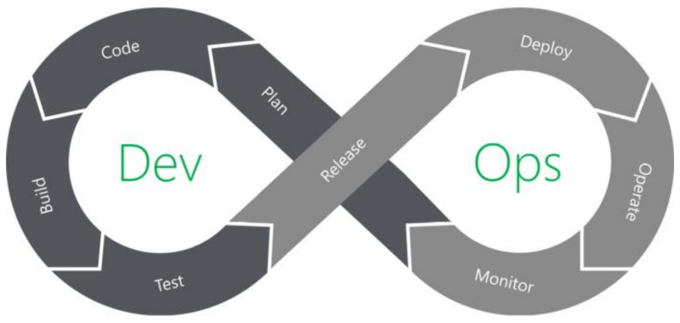
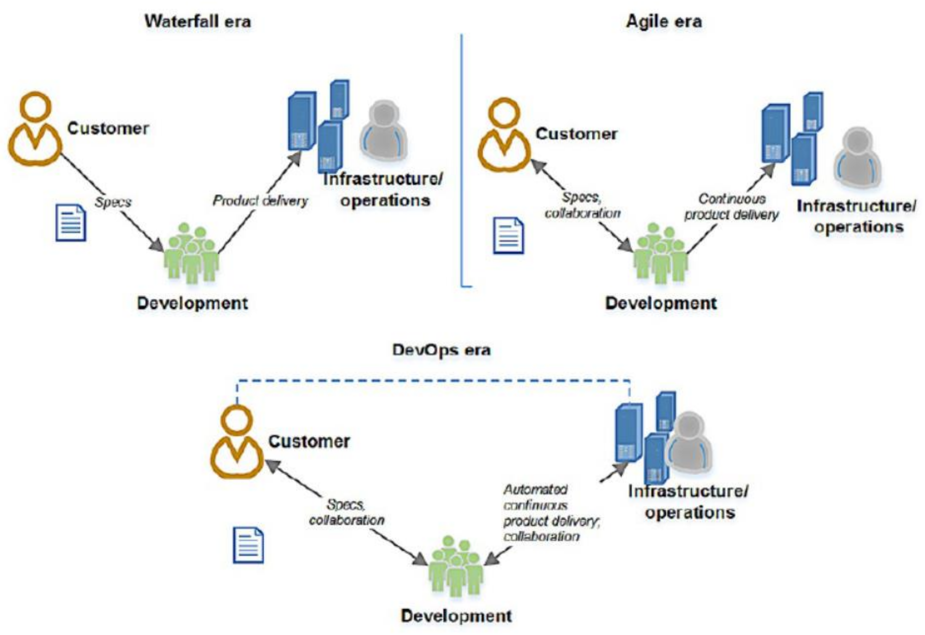


Figure 2: DevOps toolchain with all phases during software development [29].

DevOps is used to eliminate the gap between development and operations. Even though the purpose is clear, it has been rather difficult ot thoroughly define it [12]. DevOps is a relatively new concept, based on agile and lean principles, but there is still a difference between them. The lean principles are about eliminating waste and shortening the delivery time of new software. Eliminating waste could also be to shift to the left in the toolchain, meaning that tests are done earlier in the development process to keep the costs of change down. However, it is also encouraged to use value stream mapping to get an overview of the development process and in order to eliminate waste [13]. The agile principles on the other hand puts the development team closer to the customer, while DevOps looks for cross-functionality between operations and development. DevOps also brings a systematic and holistic overview over the whole development process, while agile software development might only be working with a few parts of the DevOps chain. Despite this, both agile and DevOps have the goal to quickly develop software, gather information in order to make frequent releases. Agile principles are shown to be required for a successful adoption of DevOps [10]. Prior the agile era, the development and the deployment was scheduled at specific dates and operations knew when the software would be ready for testing and could plan for that. What the agile software development has contributed with is an increased speed in software development. However, the speed of testing has remained the same, creating a bottleneck in the system due to sub-optimization. Continuous integration has been achieved and daily sent to the operations team, but they could not cope with the pace since the continuous deployment was missing and the testing was not automated [13].



Waterfall era
Customer
Specs
Product delivery
Infrastructure/ operations
Development
Agile era
Customer
Specs, collaboration
Continuous product delivery
Infrastructure/ operations
Development
DevOps era
Customer
Specs, collaboration
Automated continuous product delivery, collaboration
Infrastructure/ operations
Development

Figure 3: History of software development and illustration of how the collaborations has extended throughout the years, reaching the operations engineers [8].

Continuous integration and delivery is only a part of the DevOps philosophy. According to Meera R (2018), CI/CD are often mixed up with DevOps, where organizations are doing some CI and CD but believe that they are doing DevOps since everyone else claim that they are doing it. Allegedly, the development and operations teams are not even communicating with each other [14]. This would be equal to a company claiming they are utilizing lean production while they are only using 5S, which is only a small portion of the lean production philosophy. Instead, a literature study studying 30 articles suggests several conceptual elements that DevOps consists of. The research was conducted by studying the frequency of how often the elements were mentioned among the studied articles. The following concepts are reported (along with their frequency):

- Communication and collaboration (23%)
- Continuous delivery (21%)
- Automated pipeline (15%)
- Quality assurance (13%)
- Continuous feedback (10%)
- Continuous deployment (8%)
- Continuous planning (5%)
- Roll back code (5%)

Communication and collaboration along with automation and CD are the most frequent mentioned concepts of DevOps. As argued, DevOps is a philosophy of how to work with software development rather than only using some of these concepts and thinking that DevOps has been adopted [15]. Furthermore, these concepts are only a hint of what DevOps involves and the concepts that seem to be the most important part, but they could also include more, as DevOps expands. It is important to note that the concepts presented are not further explained as how to implement the practices, which means that each DevOps adoption might find a different way of implementing quality assurance.

1.3 Research questions and problem formulation

As presented, software has become substantial within the automotive industry and the importance will probably increase with autonomous vehicles. With the emergence of DevOps, there is a new alternative way to increase the speed of software development and releases. There is a need to study the DevOps concept in a setting at an automotive OEM to see if there is a possibility to implement it, as well as what kind of advantages and challenges it brings to the industry. Therefore, this project was based on the following questions.

Research question 1: What are the minimum requirements on a vehicle OEM to support a full DevOps process, including OTA and process monitoring?

Research question 2: What are the main problems for traditional OEMs to fully adapt to a full DevOps process, and are there any potential conflicts related to technical challenges or the automotive oligopolistic structures?

1.4 Limitations

The limitations of the research is that many of the used sources are secondary sources which means that the results relies and depends on other researchs and articles. Furthermore, DevOps is a relatively new concept that has only recently started and the research within the field is limited. This means several things. The field is dynamic and developing fast where new information and facts are successively agreed upon, thus some of the presented facts in this report migh change. It also means that there are not many books nor chapter written where DevOps and the dynamics are explained in-depth. The available sources for gathering different kind of information might be scarce, or written by the same researchers, risking similar approaches and results. Also, how companies apply DevOps might not be fully mature which might have resulted in limitations in other researches due to them not covering e.g. all benefits and drawbacks. Another limitation will be the results of the research since it is impossible to tell whether the OEMs plan on a transition to DevOps or how far they are in this transition. Therefore, the results will be generalized without taking any specific OEM into account. The research considers only OEMs as organizations, technical details of how an exact CI/CD pipeline might look like, or how the heterogenous systems in the car will look like are not included.

1.5 Report structure

The report starts with an introduction into the software and automotive field. The background and problem description is presented together with the research questions. The next chapter presents the approach the research had and how each research question will be answered. After the the results will be presented and followed up with a discussion. Finally, the report ends with a chapter of conclusions and recommendations for further research.

2 Background

2.1 How is DevOps used?

Some of the challenges and results with using DevOps are presented in *DevOps in practice: A multiple case study of five companies*, where five different case studies have been conducted. The insight of these case studies is used as an example to give a more detailed picture of how the implementation might look like. The case study largely shows how the development team and operations team have connected and integrated tasks which have allowed for elimination of the old bureaucracy that delayed tasks for a very long time. Also, it seems to be common that there are conflicting and competing metrics in terms of key performance indicators (KPIs) that exist beforehand between development and operations. During a DevOps adoption, those KPIs aligned so both teams could together focus on a common goal. In majority of the cases, the transition to DevOps was a bottom-up approach, where initiatives were taken and implemented by developers that many times had experience with DevOps since before. This naturally resulted in a change of culture towards a more supportive and knowledge-sharing one since the DevOps concept itself is relatively new. The change in culture has also shown to increase transparency where the everyone gain a higher understanding of what is happening with the developed software and how their tasks contribute to the development. The process itself can usually be like the following explained. Different tickes of small tasks are picked from a Kanban board and completed to a feature branch in GitHub which are later merged to a master branch. In this specific case, the development team acted both as a development team and as an operational one, which means that tasks could be both of development and operational nature. A continuous integration (CI) server track the changes to the master branch and builds a docker image base. These changes are later tested and reviewed in an development server by another developer. Finally, the product owner does his own tests of the implemented tickets and manually releases the code. In some cases, there are beta releases to only serve a several amount of customers to see whether there are some major issues before releasing it to everyone. This type of procedure is slightly different from case to case (Figure 7), however they all emphasize the importance of a collaboration between the development- and operations team to increase the responsiveness and decrease the time to market [11].

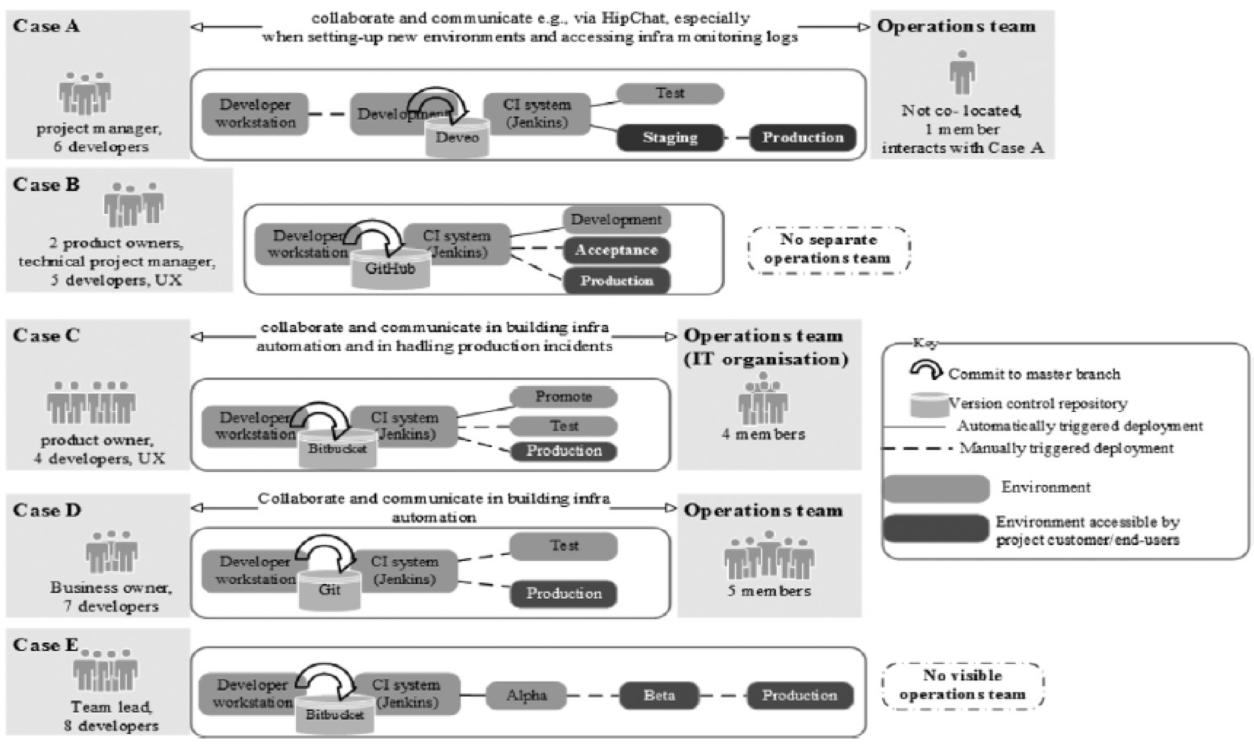

Case A
project manager, 6 developers
collaborate and communicate e.g., via HipChat, especially when setting-up new environments and accessing infra monitoring logs
Operations team
Not co-located, 1 member interacts with Case A
Developer workstation
Development
Deveo
CI system (Jenkins)
Test
Staging
Production
Case B
2 product owners, technical project manager, 5 developers, UX
Developer workstation
GitHub
CI system (Jenkins)
Development
Acceptance
Production
No separate operations team
Case C
product owner, 4 developers, UX
collaborate and communicate in building infra automation and in hadling production incidents
Operations team (IT organisation)
4 members
Developer workstation
Bitbucket
CI system (Jenkins)
Promote
Test
Production
Key
Commit to master branch
Version control repository
Automatically triggered deployment
Manually triggered deployment
Environment
Environment accessible by project customer/end-users
Case D
Business owner, 7 developers
Collaborate and communicate in building infra automation
Operations team
5 members
Developer workstation
Git
CI system (Jenkins)
Test
Production
Case E
Team lead, 8 developers
Developer workstation
Bitbucket
CI system (Jenkins)
Alpha
Beta
Production
No visible operations team

Figure 4: The DevOps process in five different cases [11].

2.1.1 Benefits from DevOps adoption

As for the case studies, the most perceived benefits of DevOps was a significantly higher speed of delivery. In some cases, the speed went from several months down to a few days. With shorter release cycles, the reaction speed to the customer needs is shortened. The pace of the whole iterative process of develop, test, release and get feedback is increased which usually means a higher customer satisfaction. When the speed of this iteration is increased, the quality of the software might get questioned. Due to the tasks being so incremental and small each time, the quality issues are lower than they were before the use of DevOps since it is easier to minimize the quality risk with smaller updates to the software. Also, the testing is automatized to a high degree and problems are solved on the fly, which means a higher quality assurance. Some developers and product owners still stated that there could be a trade-off between speed and quality, and that it was difficult to find the perfect balance between how well made the software is going to be upon release and how fast it is going to be released [11].

Another benefit of implementing DevOps is the breakdown of silos. The enhanced collaboration between the two departments software development and operations is improved due to increased communication. This eventually means that the teams exchange experience and knowledge which boosts productivity and maximize competences between the two teams. With such a collaboration and constant releases, practitioners also experience that their well-being is increased since the stress before/during/after a big release is gone. Instead they are focused on smaller day-to-day activities [16]. As collaboration and breaking down silos is one of the core elements of DevOps, it is important that this is seen as a benefit and contributes with an improvement of knowledge-sharing, communication, productivity, but also of well-being.

2.1.2 Challenges with DevOps adoption

There are more general challenges an organization needs to deal with when implementing DevOps. One of the main challenges with a DevOps adoption is the cultural shift that comes with it. Practiocioners must take on hybrid roles and different responsibilities compared to what their initial role is, e.g. a developer might need to have a more hybrid job description involving working with operations as well [16]. A change in an organization is often difficult to undergo, and making changes to break the silos and achieve DevOps might be met with opposition. Furthermore, creating such hybrid roles can lead to an initial confusion of what kind of responsibilities someone has. This can make the DevOps team assume that someone else has completed a task, leaving ut uncompleted until it is too late [17]. This might only be something that happens in the beginning of a transition before everyone are used to their new roles, but it is important to mention that this can cause trouble initially and delay realeases.

Another challenge is convincing senior management that DevOps is something the organization should transition to. In many traditional industries, a top-down control is utilized meaning that most of the decisions come from the top which involves a lot of bureaucracy. The top of the company, where the decision-making is happening, is many times separated from the bottom that is executing the decisions, therefore a gap exists between those who take the decisions and those who execute them. If there is no understanding from the senior management to why a change to DevOps is necessary, or what capabilities it could bring, they will not be willing to change since they perceive that it is good enough as it is [18].

DevOps is a new concept and to a high extent still consiedered a buzzword within software development. Companies still lack clarity on how to to fully implement the whole DevOps concept. This is because the definition is not fully clear which creates ambiguity. Different people will have different understanding of the concept, different goals, and therefore different approaches of reaching these goals [12]. Without a clear vision of how the end-state is supposed to look, companies might find themselves in positions where they are struggling to understand the whole DevOps concept, but they are still trying to achieve it since they know what kind of benefits this gives. It is important to make such a transition as painless as possible. If there is lot of frustration during the transition and still no results, the developers will get tired of change initiatives and will in the future resist such initiatives.

Finally, DevOps involves many tools and a company might find themselves in a position where the developers and operations-team will need to learn how to use these tools efficiently to fully utilize the DevOps. Usually, there is at least one tool to be used for each step in the CI/CD pipeline. This means that they might have a steeper learning curve, but also that the company needs to buy new licenses for the new tools required. If the company already has license contracts for other software, this can get expensive. Furthermore, not all companies have migrated to microservices and some of them still have monolithic artchitectures where many dependencies are involved and testing time is long. Microservices are a prerequisite to enable people work efficiently and in parallel to make new releases [19]. This migration to microservices might bring another challenge along with all the tools that are to be implemented, which might be overwhelming for a business that does not have the knowledge on how to conduct such a change.

2.2 What is currently happening in the automotive industry?

The oligopolistic automotive industry has under a long time had a high entrance barrier where only niched automakers has had some success by targeting a specific part of the market. Current cars with ICE engines have reached a maturity where innovation comes at a slower pace and is often minimal. The transition towards software and electrical focus has allowed the performance of cars to increase as well as made new features available. This has disrupted the automotive market and allowed for new automotive manufacturers like Tesla to enter the market by offering the disruptive technology in their cars from the beginning. The technology is an enabler and gives cars new capabilities. A recent article by McKinsey shows that OEMs struggle with long software development times while the complexity of software is expected to increase exponentially, resulting in an increasing gap between complexity and productivity. This means that not only will the OEMs struggle, but also their first tier suppliers that are involved in development will have issues keeping up. Furthermore, the article explains the current way of how the OEMs are developing their software; a part of the software is made in-house by the OEM itself, while the rest of it is outsourced to different suppliers. These suppliers do their part and send it back. All supplied parts are eventually stitched together to become a complete software and achieve the objective [20].

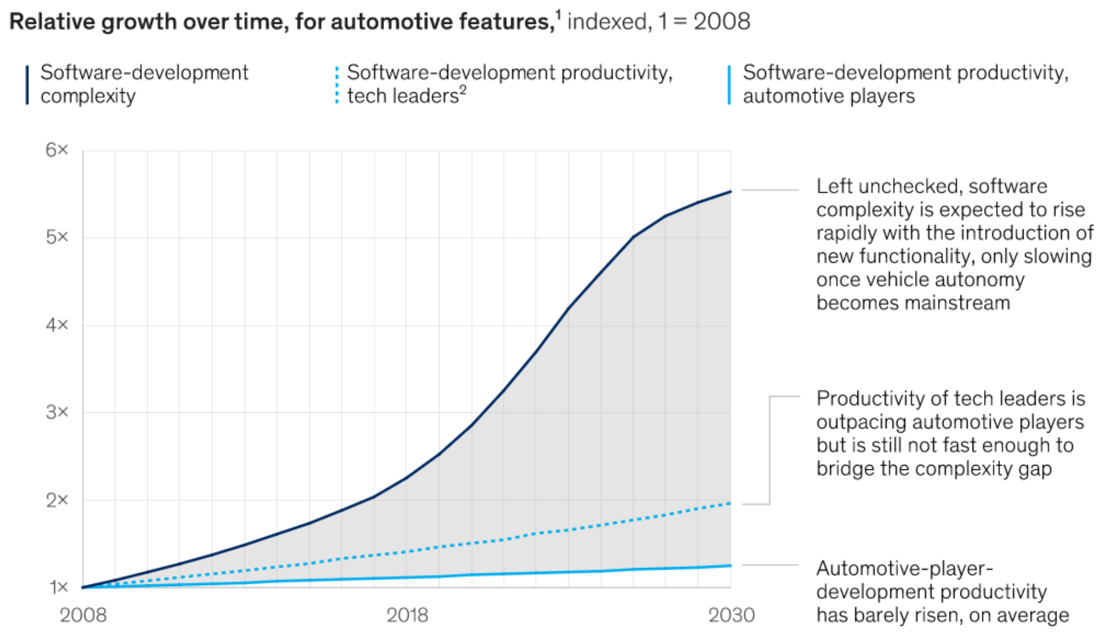


Figure 5: Evolvement of software complexity compared to software productivity [20].

The approach to software development by the OEMs is slow and complicated, with a lot of bureaucracy, which can result in delays or faulty software. One such car is the Volkswagen ID.3 that had a late launch with a 'stripped down' car with several parts of the software missing, but also further software struggles after the launch [21] [22]. Furthermore, only recently have major OEMs started to be able to update the cars over the air (OTA), while Tesla has had this possibility since 2012. This is another major sign of the slow transition and implementation of software. The OTA function makes it possible for owners to schedule software updates that are downloaded by the the Wi-Fi or 4G/5G network. This would have saved significant amount of money for the OEMs since many of cars have been recalled due to software issues, one of the latest ones being the Golf 8 [23]. This means that the owner must visit a dealership to complete the update, which can be unconvenient for the owner as well.

The only car-manufacturer that has managed well in the software field is Tesla, which is primarily a tech-company and then a car-manufacturer. They have managed to disrupt the market by having a modern and software-oriented approach, prioritizing software development and innovation over traditional car-manufacturing with focus on product quality and long development processes. This approach where speed and flexibility is more important has resulted in many times a poor quality of the vehicles, but it also seems that these quality problems are taken seriously and dealt with very quickly [24]. Tesla has also decided to make a gigafactory in Germany, and one of the reasons for that is the proximity to german suppliers which are a big part of a Tesla car [25]. Even if the build quality is still not in the top, the software quality and functions are, which is the other way around compared with other OEMs. The software is a big part of Tesla and updating/upgrading it means that the whole car product is constantly updated/upgraded, even after the customer has received it. Tesla does a massive data collection when their fleets are active (driving) which they later use to both do long-term developments of different AI systems, but also to quickly upgrade features short-term, like driving assistance and safety. Another thing that is quite unique is the ‘shadow mode’, where algorithms are run in the background while the car is driving. This allows them to see what kind of decisions an algorithm would take if it would have been implemented [26]. Speed is the capability that Tesla is utilizing by having their software development in-house and probably a modern approach to software development. The trade-off is the build quality, which they are trying to resolve quickly.

3 Method

3.1 Model

To address the second research question, a model was created of how a traditional car development process at an OEM might look like. A large part of the car development process contains software, which is assumed. This model was created with the help of three persons with long experiences and backgrounds from different OEMs. Among their experiences is different positions at Volvo Cars, Lotus and McLaren. Currently, they are all working as researchers at Chalmers University. The interviewees were interviewed and based on their answers to the questions an approximated model could be created.

3.2 Idealistic model

To address the first research question, an idealistic model of an OEM was created which illustrates how DevOps can be organized at OEMs in an idealistic scenario. It will answer the questions of what the minimal requirement for an OEM is to utilize DevOps. It will also answer the questions of how the communication will flow and what kind of communication is necessary for it to succeed. The idealistic model is based on the regular model which is created to address the second research question. To be able to set and support the dynamics within and between the teams in the model created, like division of responsibilities, the background regarding DevOps will be used.

3.3 Interviews

There were total three interviews scheduled during a two week period. The duration of the interviews were between 30–45 minutes. Two of the interviews were conducted in the following way; The research and intention was presented for the intervieewes before the questions started. There was a total of eight questions that were asked one by one and the same set of questions and order was used for each interview. The interviews were recorded and later used to analyze the answers and create the model. The third interview was slightly delayed and the interviewee asked to get the questions in a document to give preliminary answers and that we could discuss them later. This proved to be a good opportunity to give follow-up questions if there was something that needed follow up questions from the first two interviews. The reason why several interviews with different people were held is because one person might not be able to answer all of the questions. Another reason is to confirm what has been said in the other interviews and create a more accurate model. The questions that were asked were often general and not necessarily connected to each other. This since there was initially not a clear picture of how the model was going to look like or what it was going to exactly contain.

3.4 Generalization

The research could be generalized to the automotive OEMs who are outsourcing a large part of their software development. This because there is a possibility of improving speed and deployment of the software to the car fleets, especially at OEMs that has not come very far in this process. However, the challenges that the different

OEMs would face might slightly differ since all OEMs have different capabilities which might affect the type of challenges they specifically are facing.

3.5 Validity and reliability

3.5.1 Validity

Validity indicates to which extent something is precisely measured [27]. For the second research question, this means how well has the challenges with implementing DevOps at an automotive OEM been identified. The validity of research question two is considered to be high since the data collection through the interviews has led to a illustrative model of how approximately a software development process can look like at an OEM. By the help of it, another model where DevOps is applied has been developed and it has been fairly easy to pinpoint the challenges by comparing both models.

For the first research question, the validity is also considered high but it has a slightly different approach. An idealistic model of an OEM using DevOps was created based on the information gathered during the interviews, the first model and literature used in the background chapter. The reason for the different approach is simply that there must be enough understanding of the DevOps concept to be able to change the original software development approach into a DevOps one.

3.5.2 Reliability

The reliability measures the accuracy of the method. This means that if the same research would be repeated with the same approach, how many of the times would yield the same result [27]. Both research questions have an interview as an approach with a set of question that were asked different people with a background in the automotive industry. Depending on the set of questions, how they are perceived and the type of experience the interviewee has, the answers might vary. The reliability is therefore not considered high since a different research would probably yield a slightly different type of model. That is one of the reason why it is important to state that the model created is not fully accurate, and neither is the goal to have such an accurate model, but to rather map a simplification of how processes look like and identify challenges.

4 Results

4.1 Key takeaways from interviews

When OEMs are ordering the software from suppliers, the software is always intellectual property (IP) protected which means that an OEM does not have access to the specific code in this software, except in the case when they are developing a part or the full software and act as a supplier to the supplier. An OEM is however able to tweak some of the parameters for optimizations and do security analysis, but not more than that. In case they want software updates, they need to plan ahead and inform the supplier that they want it in the next version of the component. This means that suppliers of different system are a crucial part to the development of a full vehicle. The software is rigorously tested by both supplier and OEM to avoid any unwanted bugs or problems.

The type of software an OEM is usually developing in-house is the one that is close to the customer experience and this could be cruise-control or an automatic brake safety system. There are several reasons for this. An OEM wants to be first with a specific function and therefore has to develop it in-house. This gives advantage in the end market against other OEM competitors that perhaps have not developed anything similar yet. Another reason is that some software developed in-house can be used as leverage towards suppliers. If the supplier is the only one that has access to an IP protected software that is crucial for the OEM, they can set the prices. But if there is competition from the OEM that perhaps has developed some of this software, they can always get a better offer from that supplier, or another supplier. The issue with the software developed in-house is that there are usually many consultants working at an OEM and developing a high part of it. There is no guarantee that an OEM fully understands the software they have since it has been developed by consultants from other companies working at the OEM for the time.

The initial car requirements are developed at a high level in the organization, and the requirements are usually set to full-car requirements. Parts of the requirements are taken from the database from older project but as R&D develops new technology, those are added. The requirements are then broken down into systems and handed further down the organization, which turns the requirements into goals and milestones to work towards. The problem with this approach is that there is no one that is knowledgeable enough to set the full car requirements, and the approach is slowly changing to bottom up, where departments set the system requirements first which eventually ends up in a full-car requirement.

Updating existing cars with new software is very expensive and only happens in rare cases. Software is nowadays treated as something dynamic and that is constantly changing. Giving existing cars new software updates is something that happens only in cases where the software is faulty to such an extent that it is not safe enough to drive that car. The cars would need to be recalled to certified services and uploaded with new software, which is costly. The uploaded software is either created by the OEM or a supplier that has been developing that specific system.

4.2 Model of the current software development approach

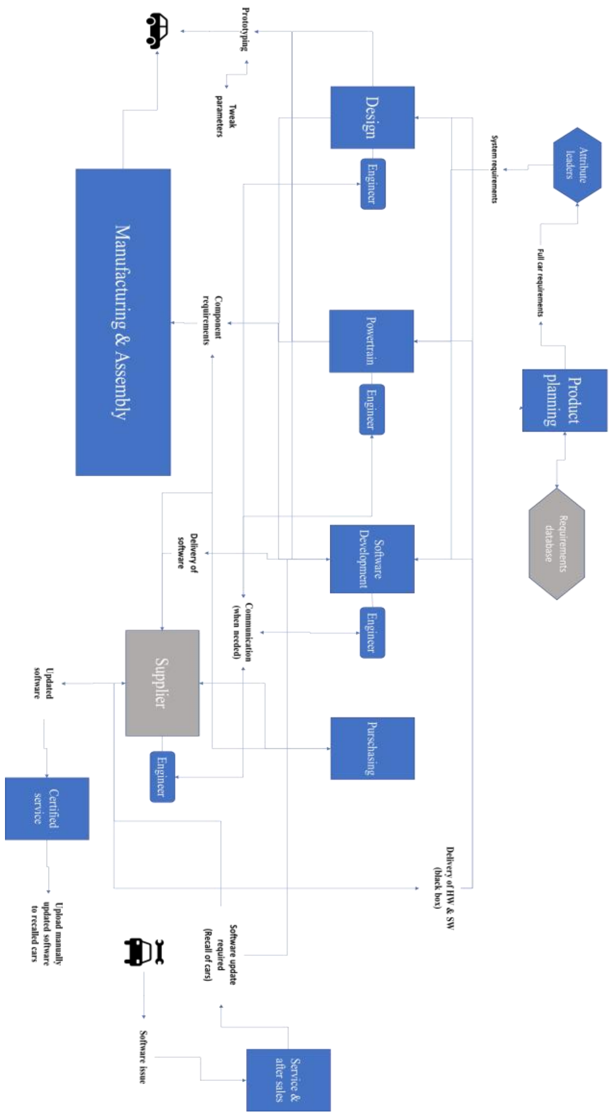


Figure 6: Model of the software development at an automotive OEM. Based on the input from the interviews.

The development of a car starts with product planning higher up in the organization. This is where the requirements for the whole car are set, such as features, software, design, etc. Some of the requirements from old projects are taken from the requirement database and re-used, but new requirements are also added as R&D and other departments continuously develop new solutions. The full car requirements are then handed over to attribute leaders where their new task is to break down these requirements into system requirements. They take decisions such as what department will be responsible for each system and where to draw the line between the different systems. The system requirements are then handed further down to each department.

A system contains many part that need to interact with each other, but at the same time the system must be robust and work as intended. Based on the system requirements, each department works with breaking down these into smaller tasks and goals, as well as setting up new KPIs to work towards to reach these goals. Decisions such as what to outsource and what to produce in-house are taken. The system is also broken down further to hardware and software components. The requirements for these components is then sent either in-house, for production, or to suppliers. When a supplier is supposed to produce a part of the system, each department discuss the component requirement together with the supplier and the purschasing department. When agreement is reached on the expectations, deliveries, etc. the part/sub-system is then ordered from the purschasing department. In some cases when there is software involved, the OEM acts as an supplier to the supplier. This means that an OEM will deliver some of its own software that the supplier uses in the system/component and delivers it back to the OEM.

When there are on-going projects, the communication between a department and a supplier happens in case of need. It is usually engineer to engineer communication where some uncertainties are cleared. When the product is developed and rigorously tested by the supplier, it is delivered back to the OEM. The software part of the system is a so called black-box, meaning that the file is compiled and no source code is visible. The only thing the OEM is able to do is to tweak some of the parameters of how the software is going to behave. This is due to IP protection of algorithms or other things existing in the source code that the supplier does not want to share. There are also contracts in place for who is responsible in case of system/component malfunction and it involves specifications such as maximum car weight allowed for the brake/coil system or whatever is delivered.

All these systems and components that are delivered internally and from suppliers are then used to make prototype cars, testing, tweaking, re-doing things and iterating until the OEM is satisfied with the result. The prototypes are made quite late in the process since there are simulation tools that are considered good enough to simulate real environments, those are used earlier in the product development process in each department. When testing is done and the results are satisfying, the car is then officially ready for production.

When the cars are on the market and in use, problems might still occur and software problems are not unusual. In case that happens, it is reported to the service and after-sales. In some cases the issue is so severe that a recall of car must be done. This happens when there is a risk of accidents, such as the auto-brake system malfunctions during some conditions that the OEM did not test. Depending on who is responsible of the software with issues, that supplier/department is contacted to make an updated version of the new software system. This is then once again tested to make sure that the problem is resolved, and compiled as a black box. The software is sent out to certified services and the people there are educated on how to update the cars. Whenever a customer arrives with a faulty car, they install the updated software and the issue is considered solved.

5 Discussion

5.1 The current approach of software development for automotive OEM’s

The current approach to the software development within automotive OEMs is highly hierarchical and top-down. The decisions for how the requirements for a car are going to look like are set high up in the organization, broken down and handed down one step. This is done until decisions are made for all components for the cars. The approach has several issues and many of them are regarding the knowledge of the people setting the requirements. They do not know everything regarding the requirements, but they use the database as help which has probably been updated by themselves during prior projects. It is possible to use this approach but there is a high probability that not the most suitable requirements are set and when it is pushed down the organization, the departments will not produce the best cars due to a knowledge gap. A lot of this is not illustrated in the model, nor was it mentioned in the interviews, but setting the requirements is probably an iterative process that is sent between different departments and people, then checked, updated and iterated again. When everything is pushed down to the departments, they start setting goals to work accordingly for their own system. The problem that can arise is that the different departments end up with sub-optimization where each department works towards optimizing their system and making it as good as possible. The systems must later be integrated with each other in a car and work together as well. One such case is when algorithms are developed but must then be translated to be able to run in an ECU in the car. Another point that was made during the interviews is that some of these people are from other consultancy firms but work for the OEM. They can produce software but the OEM might not have the knowledge to actually understand what kind of software it is and how to fully utilize it, so when a consultant leaves that knowledge is lost since no one made sure it transfers into the organization.

The step when the suppliers are involved is usually when decisions regarding a system is taken. This means that the supplier along with the purschasing department are involved to make a requirement of what the supplier is supposed to deliver. Since the requirement specification for the whole car comes from the top, there must be several compromises made here since the probability of the suppliers being able to meet all those initial requirements is not high. When there is something that is ordered, the occasional communication between egineers from OEM and suppliers is supposed to clear things up when there are additional questions. The case is probably many times the opposite due to missunderstandings, unclear communication, and where not all engineers are fully aware of the parts that are being developed by suppliers. When the hardware and software is later delivered by a supplier, the software is delivered as a black-box due to IP protection. This means that the software delivered by the supplier is already flashed onto the hardware, preventing any changed by the OEM except for parameter tweaking or security analysis. This becomes another issue if there is a new version of software available since the only way to get access to the new software is by ordering hardware with the new software version. There is no other way to implement new versions and improve functionalities. The hassle of this method is that it takes time to make the changes since they have to go through all types of bureaucracy before all specifications are changed to the new part. If there is an option to update such things OTA, there would only be an update during the next release.

Similar to the prior example, a software issue that requires a recall of cars will also be treated in a similar way. Either a supplier or the OEM needs to update the software containing bugs or other issues. The process of updating is bureaucratic as well since software needs to be tested, sent out to certified services, people need to be educated and then there must be a time when every customer is supposed to get their update. The response between reported problem and updated software is probably quite long- In many cases, customers need to adapt to when there is an available spot to get the specific update for their car. This can as well be seen as a burden or something negative.

5.2 What advantages does DevOps bring to an automotive OEM?

There are several things that an implementation of DevOps can contribute with at an automotive OEM. To illustrate this, an idealistic model was made similar to the first model, but with parts of it re-organized and changed so a DevOps philosophy can take place. The model was based on the original model, the interviews and the DevOps concept that is explained in the background.

5.2.1 Model with DevOps

There are two main differences in this model when compared to the current software development approach. The first big difference is in the approach of how car requirements are set. The different departments are supposed to collaborate and set the requirements for each system as their knowledge of the different system is probably the highest. This hopefully means that the knowledge gap that probably exists between the product design team and the different departments will be narrowed. The process of setting requirements is also probably less iterative as each department consist of people that are knowledgeable in their area and can therefore set ‘better’ requirements initially.

The second large change to the initial model is how the software development approach is done. The DevOps model contains a CI/CD pipeline as well as constant communication between the operations team at the OEM and all developers involved in different projects. This means both developer teams from suppliers and from the in-house software department. The idea with this approach is to accelerate software releases to cars that improve functions and fix existing issue through OTA updates, cutting out the prior process involving bureaucracy. There will be constant committing of software through the pipeline and then testing when it reaches the operations team. The operations team takes decisions as when to release it and how. This probably means that how the car and different hardware parts in the car handles software must be redesigned.

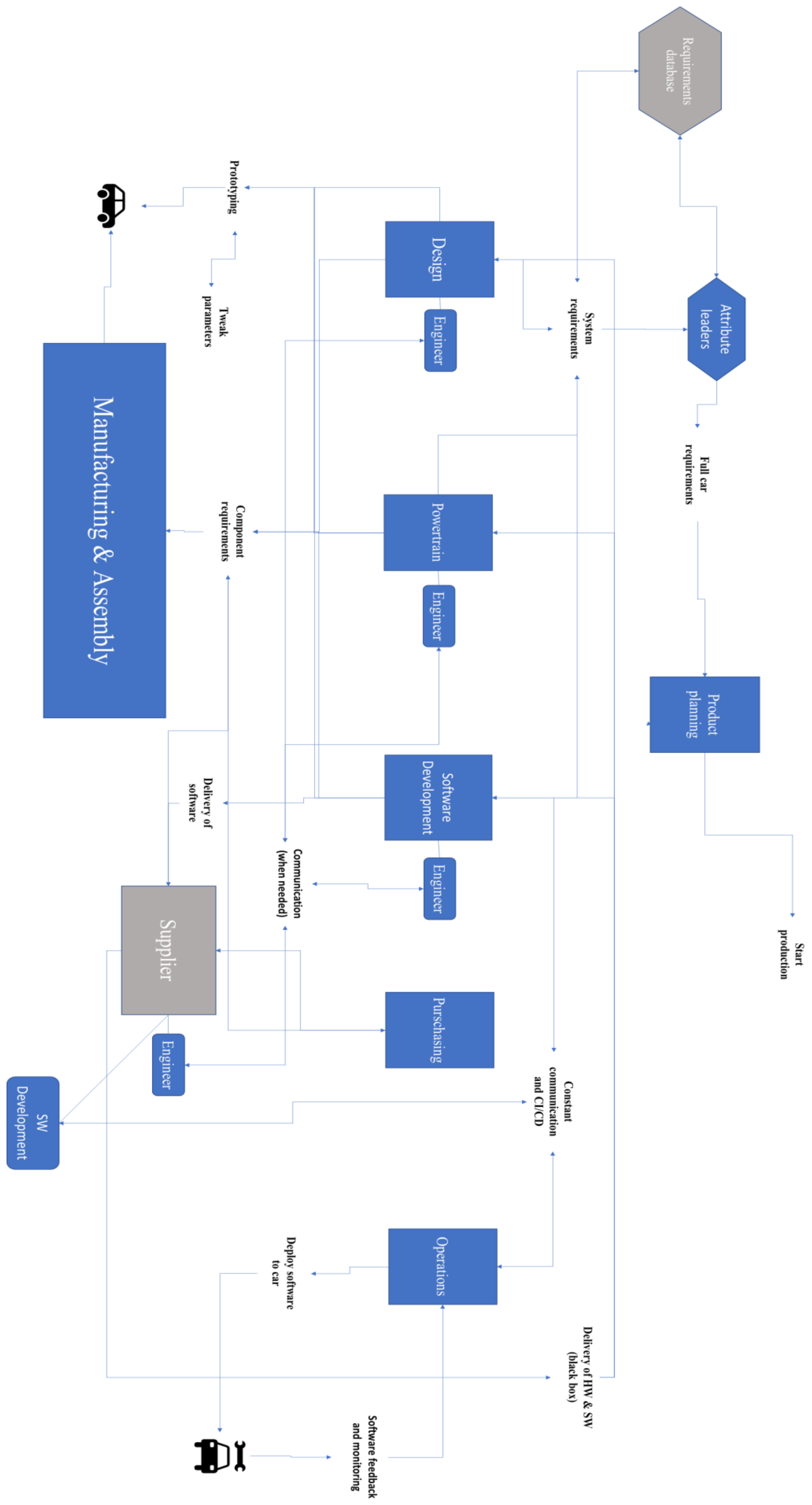


Figure 7: Idealistic model of minimal requirements of how DevOps can be implemented at an OEM. Based on the input from the interviews as well as the DevOps concept.

5.2.2 Improved user experience and well-being of workers

A car that is sold to a customer is rarely perfect in terms of software. Engineers always find new solutions/functions to add to the already existing ones. Generally, due to this kind of over-engineering, sometimes products get delayed. With DevOps in place, engineers will no longer be stressed to implement all kinds of functions before release, but only the necessary ones. The customer will get used to using the functions that come with the car, which will gradually improve. Apart from those functions, new functions will also be new added. This means that the customer experience will probably improve if the car is made more comfortable, safer or easier to use. As an example, one of the first OTA updates from Tesla added functions such as customizable seat options for each driver that is driving the car. Those could be easily changed from the computer in the car. The loop between the monitoring and software development is substantially shorter with DevOps which means that these updates will happen at a high pace. The reaction to customer demands is shorter which eventually means satisfied customers. Without the possibility to do OTA updates, the reactions to customer demands could as fastest only happen when a new model was released.

Furthermore, when there is an option to update cars over the air and add functions as explained, there will probably be an improvement in workers well-being since they do not have to work towards strict deadlines risking a delay in release or long over-time hours. Naturally, there will be less stress and tensions within teams working with software implementation.

5.2.3 No expensive recalls anymore

To recall cars to certified services, make new software and implement it with a manual approach is very expensive. In the last 10 years, the average recall of a car has had a cost of $500, including both software and hardware updates [28]. With larger fleets being recalled, it can soon get very expensive for the OEM. But the cost of a recall is many times more than just replacement of parts and software updates, the speed and quality might be equally important for customer satisfaction which is difficult to measure in money. Another such variable to consider is the brand perception by current and potential customers. Recalls eventually means a worse reputation which might discourage potential buyers from buying a car from that brand.

5.2.4 Data collection and improved software development process

As algorithms for different systems run in the car, the OEM can similarly to Tesla collect data of the decisions algorithms make when they run in the car, or in the background. This can further improve the software development process as many decisions are based on certain data, and data is easier to collect when there are many cars that are being used in a real environment where all kinds of situations exist. However, there is a large difference between how data is collected in the EU compared to US. Due to the GDPR law, citizens of EU must opt-in if they wish their data to be collected by someone. The US has an opt-out approach, meaning that data is collected by default as long as the user does not change the specific setting. This means that there will probably be less data from the EU roads to rely upon for OEMs and that they have to be careful with how they use the data and how well it is protected.

5.2.5 The integration of suppliers and OEM

A B2B supply chain integration has many advantages since no business can standalone produce everything that is required for the end-product. This means that an OEM needs to rely on its suppliers to effectively execute what is required of them. Through a digital integration between suppliers and OEMs, which is common within supply chain management, tasks can be executed automatically with no need of human interference. Similar to this, an integration between development teams at suppliers and the operations team at an OEM is required to reach a state of DevOps. The borders between two organizations need to be broken down through communication and collaboration, which might be more challenging than breaking down silos within organizations. This approach is supposed to enable a CI/CD pipeline where the different development teams are suppliers but the operations teams are at an OEM. The reason for this approach would be that the OEM is responsible for the monitoring and feedback of the car fleets, but also of the deployment of new software. The development teams that work for different suppliers are there to improve and develop new functions that are related to their specific field.

The most OEMs have during a long time relied on their suppliers to develop software for them, which has resulted in a situation where OEMs cannot develop software without suppliers (except for Tesla and similar actors). It is impossible in short-term to bring the software development in-house and start learning the necessary capabilities to develop this kind of software and algorithms, therefore the best solution is to let the suppliers do it since they have the capabilities and knowledge. There is a lot of potential in this kind of approach and collaboration past the CI/CD pipeline. Additional services that are not initially there can be developed and sold to the customers. Another alternative is to sell extra packages that the customer perhaps decide they did not need initially. One such thing could be that the driveline of a car has a certain amount of horsepowers, but if the customer decides they want a stronger car, they can buy such an upgrade through the OEM’s homepage and update the car OTA.

5.3 Challenges with DevOps adoption at an OEM

As there are advantages with a DevOps adoption at an automotive OEM, there are challenges as well.

5.3.1 Centralized decision-making

The automotive industry has a history of being conservative in the way it is managed. Many decisions are centralized, a top-down approach is used and the organization is built for efficiency. Efficiency is many times necessary when competing on price because this keeps the margins high enough to make profit. The trade-off with this kind of approach is that it does not cultivate innovation as much as less centralized approaches. Historically, many organizations with a long history has been conservative in their way of organizing without embracing change to fit the dynamic environment that has come with the modern world. Many of these organizations have gone bankrupt, been bought or lost a substantial part of the business to disruption. A report by Mckinsey shows that in the end of the 70’s, the average tenure of a company on the S&P500 list was 35 years. The average tenure for a company on the same list was down to 20 years by 2019.

There is currently an on-going transformation in the automotive industry which is necessary. The reason is that software and innovation is becoming a larger part of cars and automotive organizations built only for efficiency will likely struggle. The DevOps transformation is in many cases a bottom-up transformation and if the organization does not allow for this kind of initiatives, the likelihood of it never being implemented is high. As an example of the transformation, it was said during the interview that the requirement specification for each car project is changing from a top-down approach to bottom-up. In that case, each department creates their own requirements which are coordinated before eventually ending up in a full-car requirement. The same approach must be used in the case with DevOps, which would allow for a bottom-up change. The other option is to prove the value that DevOps creates and hope that it is persuasive enough for someone who can take the final decision. It can seem to be an easy decision to take since DevOps brings advantages but it is only when DevOps is considered exclusively without taking anything else into consideration. In an organization there are already contracts for existing software that are probably tied for several years, there are already existing software development processes that must be compared to DevOps, there is a change that must happen which costs money and finally the people must be able to adopt and perhaps get new responsibilities. There is always an alternative cost for decisions like these and to consider whether such a change will have a higher long-term positive effect than the current development process.

5.3.2 Integration of suppliers and OEM

As much as an integration of suppliers can be advantageous, they can also be disadvantageous and create unwanted friction. The kind of integration and communication must be created and based on new kind of culture that neither of the organizations is used to. The operations team at the OEM and the different development teams at the different suppliers will have two different cultures to alternate between, the culture within the organization and the culture that is based on this collaboration. There is no guarantee that this will work in all type of cases, or that the collaboration will be smooth as they wish it to be. Unfortunately, all of the DevOps studies found and used for this report only focus on DevOps as within a single organization. It is difficult to draw any conclusion of whether a cross-border DevOps would work as efficiently as DevOps within the organization.

Another issue is the way decision-making and bureaucracy exists within the automotive industry. DevOps is supposed to remove a lot of this, and it is possible when it happens in a single organization. But when using cross-border DevOps there might be higher entry barriers such as trust issues. Therefore each project given to the development team at a supplier would probably have to be confirmed by someone on a higher level, and this project would probably contain a specification of what the supplier is supposed to develop. This kind of specification for specific projects proved to be counterproductive since every time there was a new idea of how something is going to be programmed, it was necessary from the issuer of the project to confirm it by changing the specification. A solution to this problem could be to set overall goals of what functions to develop and how to improve existing software without necessarily having strict requirement lists. But then again if things are not specified enough, both parties might be disappointed by how things turn out and dissatisfied by the amount of work that is being done which would not favour any of them.

Finally, there must be a CI/CD pipeline that both the development and operations teams are using to code, test and deploy everything. There is usually a tool for each step in the pipeline and it is easy to organize this on the operations side since the whole team will belong to the OEM. Suppliers might want to work with different softwares when coding, depending on their preferences, experience, etc. This makes it more difficult to standardize the tools within the pipeline, at least the development part as well as someone needs to pay for the subscriptions of the different softwares. There will also be a need from the supplier side to protect the source code from the OEM as it is considered IP that belongs to the specific supplier. This might add further problems as practitioners of DevOps many times need to take hybrid roles with both development and operations tasks. The operations team will be testing the developed software without access to code, if that is possible. This will add an additional level of bureaucracy by sending it back to the developers at the supplier to re-do the code. It makes it probably even more difficult to find the exact issue if the operations team does not have access to the source code. This is perhaps only a trade-off with cross-border DevOps, but there might exist more advanced solutions to this problem.

5.3.3 Culture change

One of the most difficult and important transformations regarding DevOps is the culture. Without a culture where borders between development and operations are brought down and people take new responsibilites, there will be difficulties with implementation. As mentioned earlier, collaboration is one of the main essences of DevOps and this cross-border DevOps between organizations requires a new type of culture. The culture that will emerge between supplier development teams and operations team at OEM will probably be unique and difficult to maintain. They are different organization with their limits of what information they are allowed to share and how much, meaning that this new culture will contain many restriction. But they must still work together as a team to offer customer value and quickly bring new updates to software. Furthermore, depending on how the operations team is set up, and with what supplier they are collaborating with, the culture might be slightly different for each collaboration. As an example, one development team at supplier #1 might use specific tools to develop the specific software they are responsible for and then hand it over to operations team #1. This relationship will probably have a specific culture dependent on prior contact, type of collaboration and tools used. Then there will be another development at supplier #2, using different tools and processes to develop their specific software, but they will still collaborate with operations team #1 at the OEM, as each supplier will probably not be assigned an operations team.

Another thing to consider is how the internal culture will change between the development team and operations, since these team members can use DevOps’s full potential with hybrid roles. This means that they can share all information, source codes, etc. But then this kind of approach will be different to the approach that is being used with each supplier. Once again it adds more complexity while the purpose with DevOps is based on standardization, lean and agile principles.

6 Conclusion and future research

The accelerated pace of software development has introduced DevOps as a philosophy, emphasizing collaboration and CI/CD withing software development. Despite DevOps being a relatively new term, it is clear that it has its benefits as increased speed of software releases, OTA updates and quality assurance. Despite the challenges that exist within the automotive indsutry such as cross-border DevOps and a highly centralized decision making, the belief is that DevOps can still be utilized within the industry. For this to happen, all parts that are involved must work towards enabling a state of such collaboration by actively working to change the current environment and processes when software is being developed. Such changes will amongst others involve the integration of suppliers and a culture change towards a DevOps philosophy. There must be a certain way of collaboration between the borders since this does not affect only the OEM, but also the suppliers. This change is perhaps not necessary to make, nor are the all other possible changes a business can make. However, an implementation of DevOps should be given more consideration than an average change since the future of automotive clearly lies within software, and an automotive manufacturer that does not wish to exceed in this field will probably struggle to gain on the competitors in the future.

There can be made several recommendations for future research. One of them is to evaluate how an organizational implementation of DevOps is to be enabled in terms of supply chain. This practically implies to in more detail study how a cross-border DevOps can be enabled with respect to the IPs that exist at different suppliers as well as taking the tools and different pipelines into consideration. Furthermore, how will the culture change affect all involved stakeholders since the ones involved with DevOps will have one culture while maintaining the other culture within the walls of the company. Another recommendation is to evaluate the supply chain in terms of cars and technology. This means how the different cars can be redesigned to enable a CI/CD pipeline with continuous OTA releases that update the car in a specific way but also what is required from different servers. Many of the cars have currently parts that have installed software directly on them. This means that an update of that specific part is not possible and the only way is changing the hardware. This must be redesigned depending on what kind of updates the OTA releases will contain.

7 References

[1] H. Ford and S. Crowther, My life and work, Cornstalk publishing company, 1924.

[2] G. Fedorko, V. Molnár and D. Sabadka, “Shortening of life cycle and complexity impact on the automotive industry,” *TEM journal,* vol. 8, no. 4, pp. 1295-1301, 2019.

[3] Euroean comission, “CO2 emission performance standards for cars and vans (2020 onwards),” 06 11 2017. [Online]. Available: https://ec.europa.eu/clima/policies/transport/vehicles/regulation_en. [Accessed 25 02 2021].

[4] D. Carrington, “Global sales of electric cars accelerate fast in 2020 despite pandemic,” The Guardian, 20 1 2021. [Online]. Available: https://www.theguardian.com/environment/2021/jan/19/global-sales-of-electric-cars-accelerate-fast-in-2020-despite-covid-pandemic. [Accessed 25 02 2021].

[5] D. Faggella, “The self-driving car timeline - predictions from the top 11 global automakers.,” 14 03 2020. [Online]. Available: https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/. [Accessed 25 02 2021].

[6] J. Madden, “How much software is in your car? From the 1977 Toronado to the Tesla P85D,” 1 04 2015. [Online]. Available: https://www.qsm.com/blog/2015/how-much-software-your-car-1977-toronado-tesla-p85d. [Accessed 25 02 2021].

[7] A. Madhok, “Three quarters of all new cars to be connected in five years.,” Counterpoint research, 20 11 2020. [Online]. Available: https://www.counterpointresearch.com/three-quarters-new-cars-connected-five-years/. [Accessed 25 02 2021].

[8] P. Kuvaja, L. E. Lwakatare, O. Markku, M. Mäntyä, P. Rodríguez och P. Seppänen, ”Advances in using agile and lean processes for software development,” *Advances in computers,* vol. 113, p. 358, 2019.

[9] K. Beck, M. Beedle, A. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, M. Robert, S. Mellor, K. Schwaber, J. Sutherland och D. Thomas, ”Agile manifesto,” 2001. [Online]. Available: https://agilemanifesto.org/. [Använd 31 03 2021].

[10] P. Kuvaja, L. E. Lwakatare and M. Oivo, “Relationship of DevOps to agile, lean and continuous deployment.,” in *International conference on product-focused software process improvement*, 2016.

[11] L. E. Lwakatare, T. Kilamo, T. Karvonen, T. Sauvola, V. Heikkilä, J. Itkonen, P. Kuvaja, T. Mikkonen, M. Oivo and C. Lassenius, “DevOps in practice: A multiple case study of five companies,” *Information and software technology,* vol. 114, pp. 217-230, 2019.

[12] K. Nybom, I. Porres och J. Smeds, ”DevOps: a definition and perceived adoption impediments,” i *International conference on agile software development*, 2015.

[13] S. Sharma, The DevOps Adoption Playbook : A Guide to Adopting DevOps in a Multi-Speed IT Enterprise : A Guide to Adopting DevOps in a Multi-Speed IT Enterprise, John Wiley & Sons, 2017.

[14] S. Mansfield-Devine, ”DevOps: Finding room for security,” *Network security,* vol. 2018, nr 7, pp. 15-20, 2018.

[15] G. B. Ghantous och A. Gill, ”DevOps: Concepts, Practices, Tools, Benefits and Challenges,” i *Pacific Asia Conference on Information Systems*, 2017.

[16] L. E. Lwakatare, S. Mäkinen, T. Männistö, L. Riungu-Kalliosaari och J. Tiihonen, ”DevOps Adoption Benefits and Challenges in Practice: A Case Study,” *Product-Focused Software Process Improvement,* vol. 10027, pp. 590-597, 2016.

[17] J. Buchan, H. Osman och M. Senapathi, ”DevOps capabilities, practices, and challenges: insights from a case Study,” i *The 22nd international conference*, 2018.

[18] M. Kamuto och J. Langerman, ”Factors inhibiting the adoption of Devops in large organisations: South African context,” i *IEEE international conference on recente trends in electronics information & communication technology*, 2017.

[19] F. Kon, L. Leite, P. Meirelles, D. Milojicic och C. Rocha, ”A Survey of DevOps Concept and Challenges,” *ACM Computing Surveys,* p. Article 127, 2019.

[20] R. Fletcher, A. Mahindroo, N. Santhanam och A. Tschiesner, ”The case for an end-to-end automotive-software platform,” McKinsey, 16 01 2020. [Online]. Available: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-case-for-an-end-to-end-automotive-software-platform. [Använd 14 03 2021].

[21] S. O'Kane, ”VW’s first mass-market EV suffers delay thanks to software struggles,” The Verge, 11 06 2020. [Online]. Available: https://www.theverge.com/2020/6/11/21288572/volkswagen-id3-ev-delay-software-vw-herbert-diess. [Använd 14 03 2021].

[22] J. Traugott, ”Volkswagen's Major ID.3 Problem Has Been Solved,” Carbuzz, 06 12 2020. [Online]. Available: https://carbuzz.com/news/volkswagens-major-id-3-problem-has-been-solved. [Använd 14 03 2021].

[23] T. Seythal, ”Volkswagen recalls 56,000 Golf models for software update,” Reuters, 12 1 2021. [Online]. Available: https://www.reuters.com/article/uk-volkswagen-software-idUSKBN29H15Q. [Använd 14 03 2021].

[24] C. Isidore, ”Elon Musk admits Tesla has quality problems,” CNN, 03 02 2021. [Online]. Available: https://edition.cnn.com/2021/02/03/business/elon-musk-tesla-quality-problems/index.html. [Använd 07 04 2021].

[25] J. Hofer, A. Postinett och M. Wocher, ”Tesla: German Under the Hood,” Handelsblatt, 28 01 2016. [Online]. Available: https://www.handelsblatt.com/english/companies/e-car-components-tesla-german-under-the-hood/23535490.html?ticket=ST-20060-tFO7m7Uf6Se7aHwjaRvH-ap6. [Använd 07 04 2021].

[26] G. Paolini, ”Tesla, the data company,” CIO, 28 08 2019. [Online]. Available: https://www.cio.com/article/3433931/tesla-the-data-company.html. [Använd 07 04 2021].

[27] R. Heale och A. Twycross, ”Validity and relieability in quantitative studies,” *Evidence-based nursing,* 15 05 2015.

[28] C. D.-V. P. Isidore, ”CTTV,” 25 02 2021. [Online]. Available: https://www.ctvnews.ca/autos/hyundai-s-recall-of-82-000-electric-cars-is-one-of-the-most-expensive-in-history-1.5324265. [Använd 07 05 2021].

[29] J. Pennington, Artist, *DevOps chain.* [Art]. Medium, 2019.

[30] A. Lau, ”Motortrend,” 06 10 2020. [Online]. Available: https://www.motortrend.com/news/tesla-model-y-ev-safety-quality-issues-problems/. [Använd 10 03 2021].

[31] C. Isidore, ”CNN Business,” 03 02 2021. [Online]. Available: https://edition.cnn.com/2021/02/03/business/elon-musk-tesla-quality-problems/index.html. [Använd 10 03 2021].

8 Appendix

8.1 Questions that were used for the interviews

1. How does a car development project look like regarding requirement specifications and communication between the departments? Where in the organization are the requirement specifications coordinated?
2. Does every department communicate with their suppliers or does that happen on a higher level? Can a department write contracts with suppliers by themselves? Can a department decide what type of software version will be included in a component?
3. How does the process between a purchasing decision and delivery look like? How do you update the software?
4. What procedure is there when a car is recalled due to software issues?
5. What kind of responsibility does a supplier have except for delivering the product. What happens in case the software from a supplier is faulty and needs to be updated/replaced when it is already in the car?
6. What kind of IP protections are there regarding the suppliers' software? Is there any transparency so the OEM can make changes to the software? Can an OEM do a security analysis on the software?
7. How is software made? E.g., if an OEM wants to develop something like cruise-control. How big part of it does suppliers deliver?
8. Advanced software functions are developed in e.g., MatLab/Simulink presumably in a department that works with ADAS. How and by whom is this translated to a code that can be used in an ECU? How is this kind of software updated?

DEPARTMENT OF MECHANICS AND
MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY