CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Assessment and optimization of automation processes in environments used for Autonomous Driving

Master's thesis in Computer science and engineering

OSCAR FORSBERG
JOHANNA WIBERG

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Assessment and optimization of automation processes in environments used for Autonomous Driving

Oscar Forsberg, Johanna Wiberg

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Assessment and optimization of automation processes in environments used for Autonomous Driving
Oscar Forsberg, Johanna Wiberg

Supervisor: Miroslaw Staron, Department of Computer Science and Engineering
Advisor: David Kastö, Volvo Cars
Examiner: Gregory Gay, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Assessment and optimization of automation processes in environments used for Autonomous Driving
OSCAR FORSBERG
JOHANNA WIBERG
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Testing autonomous driving functionalities is a very important but complicated task. Due to the high costs of real-life testing, testing in virtual machines has become a must in the industry. At Volvo Cars, a virtual testing toolchain where traffic scenarios are generated, simulated, and analyzed is utilized. However, due to a large number of test cases, this is a very costly process. This thesis aims to optimize the execution time of the testing toolchain to cut down costs. The study started with profiling the toolchain to identify bottlenecks and areas where execution time could be shortened. Based on the profiling, a concept of a "monitor" was proposed, which analyzes scenarios in real-time rather than sequentially. The study implemented a proof of concept of a "stay-in-lane monitor" which checks whether the monitor remains within its original lane at every time step of the simulation. The results showed that the "stay-in-lane monitor" significantly reduced the toolchain's execution time. Based on these results, the "monitor" concept was deemed successful, and the team at Volvo Cars will continue to implement more monitors in the future. We conclude that the optimization approach presented will help the collaborating team achieve more efficient testing, thus reducing costs and improving the development of autonomous drive functionalities.

Keywords: automation processes, testing, tool-chain, assessment, optimization, autonomous driving, automotive, execution time.

Acknowledgements

We would like to express our sincere gratitude to our supervisor, professor Miroslaw Staron for their guidance, support, and feedback throughout the study. Their knowledge and expertise have been crucial in helping us achieve our results. In addition, this project would not have been possible without the support of our advisor David Kastö and the members of the simulation team at Volvo Cars who have helped majorly with technical challenges and validation of results.

We would also like to extend our appreciation to our examiner professor Gregory Gay for taking the time to review our work and provide insightful feedback that has helped us improve the quality of our research and report.

Thank you!

Oscar Forsberg, Johanna Wiberg, Gothenburg, June 2023

Contents

List of Figures

List of Tables

1
Introduction

For the last few years, autonomous car technology has advanced rapidly [1], leading to autonomous functionalities in cars being the most prominent drivers of value [2]. Autonomous functionalities offer several major benefits; improved safety, business opportunities, ease of use, and convenience, among other things [1]. While early regulations require drivers to take control of autonomous cars in uncertain situations, upcoming expansions of the technology will uphold a higher level of autonomous drive automation [3].

Lately, testing of autonomous cars has partially been conducted in virtual machines, "software-in-the-loop" (SIL), to minimize the time it takes to drive in a real-life car and to minimize the high cost of physical testing [4]. It has been shown that to secure the reliability of autonomous cars, a physical car has to drive from a hundred of millions of miles up to hundreds of billions of miles [5]. The method of virtual testing provides the ability to recreate complex traffic scenarios by the use of real data from the field, whilst allowing tests with a multitude of conditions, such as weather and climate. Virtual testing also provides advantages such as efficiency, since one can run tests in parallel, or run by day or night, whilst supporting the testing of human-related aspects [4]. The demand for fast and efficient testing grows as the vehicle automation industry continuously pushes its efforts to advance to the next level of automation [3]. This is where automation of testing comes into play.

The study was executed in collaboration with the car manufacturer Volvo Cars and the in-house team Autonomous Driving Functions Verification & Validation and will be referred to as AD functions V&V in the rest of this thesis. The team is currently using a toolchain to simulate scenarios and test their functionalities. However, this testing toolchain needs to be optimized to handle the expected increase in scale in the near future.

1.1 Problem Statement and Purpose of the Study

The software powering these autonomous functionalities, such as the ones used by the collaborating company, is intricate and must be precise [1], since they operate in critical life-and-death situations. Autonomous cars safety standards dictate that the utilized software must be up-to-date and quickly and accurately deployable to comply with safety requirements [6][7]. Due to the risk of accidents in the automotive industry [4], safety, reliability, and quality [1] are of high importance. Testing

is therefore highly important as well. However, this comes with a few challenges [4].

Two of these challenges are scenario complexity, and the number of test cases [1][4], Knauss et. al. explains these difficulties: "*As automated vehicles will go everywhere and have to be tested that they are safe in every environment, not only the complexity of the scenarios but also the variety of test cases to test for will have to increase*", as well as that it's "*...almost impossible to think about all possible traffic scenarios*" [4]. To scale continuous integration modules in larger projects and companies has proven to be difficult [8][9]. As software and projects get bigger and bigger, a chain reaction occurs that usually affects; the size of code, the build duration, the scope of tests, and the number of people involved, which in turn could affect the rate of changes [9]. Logically, scenario complexity and the amount of test cases would impact the rate of changes as well.

Therefore, the thesis aims to optimize the execution time of the collaboration company's AD functions V&V team's testing toolchain. Executing the toolchain is currently a costly process due to the large amount of data generated, as well as strict testing and scenario requirements. The aim is to address the challenges posed by testing a large number of traffic scenarios and to find ways to reduce the time it takes to execute the toolchain. Even if the improvement of time is minor, it can result in significant cost savings since the toolchain should be capable of handling hundred of millions of scenarios regularly in the future. Ultimately, this optimization can cut costs for the organisation and help the AD functions V&V team create a faster deployment of functionalities.

1.2 Research Questions

As previously stated, in Section 1.1, the study aims to assess and optimize Volvo Cars AD functions V&V team's toolchain. It's currently very costly to run due to the large amount of data. We aim to answer the following research questions in the context of a car manufacturing autonomous driving functions group:

RQ1: *What characterizes an optimal testing toolchain for an AD functions Verification and Validation team in the automotive industry*?

Identifying the characteristics of an optimal testing toolchain is crucial to enable efficient and effective testing. By understanding the key features and functionalities of an effective toolchain, AD functions V&V teams can improve their testing processes, reduce costs, and accelerate time-to-market.

Some characteristics of an optimal testing toolchain for AD functions V&V teams may include scalability. As touched upon previously, testing is particularly important in autonomous drive functionalities due to the strict safety requirements. As the company reaches higher degrees of automation these requirements will get stricter. To ensure safety, the testing toolchain will need to be scaled up to handle more scenarios.

RQ1.1: *How should a testing toolchain be optimized for an AD functions Verification and Validation team in the automotive industry*?

Addressing the question of how to improve a testing toolchain in the automotive industry can provide numerous benefits, such as providing teams with a concrete approach to optimizing their toolchain as well as it can provide teams with ideas on how to compose their toolchain most effectively. This could result in teams improving their toolchain, resulting in greater product quality and a competitive advantage in the market.

RQ2: *What are the challenges of testing AD functionalities in automated processes in the automotive industry*?

By finding specific challenges of testing AD functionalities in automated processes in the automotive industry, organizations can create guidelines and best practices for the testing and development of their AD functions. By working towards these guidelines, optimization of the testing and development process could run smoother, and hopefully decrease the price of future problems. All of this whilst increasing the safety and reliability of their AD vehicles.

The three research questions investigated in this thesis aim to solve the challenges and problems of testing AD functions brought up in Section 1.1. By identifying the characteristics of an optimal testing toolchain and how to optimize it for an AD functions V&V team, the thesis can offer insights into how to improve the current toolchain. In turn, this could assist in decreasing the execution time of the toolchain and resulting in a decrease in cost spending for organizations. Additionally, by identifying specific challenges of testing these AD functionalities, the thesis can create guidelines and best practices that assist the organization in optimizing the testing and development process and ensuring that they follow required laws and regulations.

1.3 Limitations

The research context is limited to the in-house toolchain used by an AD functions V&V team at the collaboration company. Therefore, this thesis is constrained by the automatic testing of autonomous features in the automotive industry, of a specific team. The study will not consider the accuracy or efficiency of the autonomous functionalities themselves as the code for these functionalities is outsourced to a separate company and is only received as a black box.

Secondly, the thesis will focus on the technical parts of the toolchain, thus eliminating any discussion in regards to the organization of the team, managing said toolchain, or e.g. how often or when the team should execute the toolchain. Even though it is of great importance, the time constraint of the thesis makes us unable to dig deeper into this discussion. Furthermore, the thesis will not look at the envi-

ronment in which the toolchain is executed, such as cloud and node configurations, as that is decided higher up within the organization thus the thesis will be unable to affect this aspect.

1.4 Significance of the Study

The study aims to offer insightful information on how to optimize expensive testing toolchains, which process large volumes of data as a result of testing a rigorous amount of critical scenarios. Autonomous vehicles are the future of transportation, thus requiring a high level of safety, reliability, and quality [1]. The risks associated with traffic accidents are clear, and the consequences could sometimes be fatal [4]. Therefore, any effort to increase the safety of AD functions is of high importance.

The thesis could provide benefits, not only to the collaborating company working with automatic AD functions testing but to the entire automotive industry. The hope is that reducing the execution time for toolchains could lead to faster deployment of new software updates, which would ultimately result in safer autonomous vehicles on public roads. It would decrease the risk of accidents and increase the confidence of various actors in the technology, such as potential customers, leading to an easier and increased adoption of autonomous vehicles.

Since AD functionalities in cars are the most prominent drivers of value [2], it is critical from a business standpoint to keep up with or be ahead of, competitors. A testing toolchain that can execute a large number of scenarios in a relatively short time may result in faster software deployment and, as a result, more advanced functionalities in the vehicles. With the increased demand for autonomous functionalities, it is critical to have a toolchain that can effectively test and validate these functionalities.

Lastly, toolchains like the collaborating companies are costly to run to this day, thus reducing the cost of running these chains in the automotive industry is of high value. Even a minor improvement in the toolchain could result in significant cost savings for companies long term. Therefore, this study's potential cost savings should not be underestimated, as it could lead to significant cost savings for the collaborating company and similar companies trying to advance in AD functions.

1.5 Structure of the Thesis

The structure of the thesis is the following: Chapter 1 will go through the introduction of the subject and its challenges in the autonomous industry, and introduce the collaborating company. Moreover, Chapter 1 will introduce the research questions, goals, significance, and limitations of the study. Chapter 2, Background, will highlight important areas connected to the thesis and testing of autonomous cars. Chapter 3, Research Design, will go through the research methodology utilized in the thesis, whilst Chapter 4, Execution and Results, will provide the execution and

results, hence the name. Chapter 5 will discuss the results, and Chapter 6 will conclude the thesis and introduce future work.

2
Background

This chapter goes through all the background information necessary to understand the thesis. Topics such as levels of autonomy, scenario generation, testing architecture, and performance optimization will be described, among others. The chapter serves as the theoretical baseline of the thesis and is tightly coupled with the study and the collaborating team's practice.

2.1 Levels of Autonomy

There are six automation levels, according to the Society of Automobile Engineers (SAE) [3][10], which can be seen in Table 2.1. A vehicle with level 0 has no automation implemented whereas a vehicle with level 5 is fully automated. For level 1, the automation can either control the vehicle's side-to-side or front-to-back movement, but not both at the same time. The driver must be responsible for handling the rest. For instance, anti-lock braking systems, and adaptive cruise control fall under level 1 of automation. Level 2, partial driving automation, enables the automation system to control both the side-to-side and front-to-back movement of the vehicle, but the driver must supervise the active system and complete object and event detection and response (OEDR). In level 3, conditional automation, the autonomous driving system performs 100 percent of the driving task in certain conditions, for instance, highways. However, as the driver is still required to take over if the system would fail or is unable to calibrate how to operate in a certain situation, it does not reach the higher levels of automation. The last two levels, 4 and 5, do not require a driver. Though whilst automation level 4 is constrained by the type of roads it can be active on, they must have proper maps and road structures, level 5 can be active without any limitations at all.

As with most organizations, the AD functions V&V team at Volvo Cars works with these automation levels in mind. To ensure the quality and safety of AD functionalities there exist international regulations that all companies are required to comply with to be able to activate their functions on public roads. An example of such regulation is the United Nations Rule No. 157 "Automated lane-keeping systems" [6]. The role of software in achieving automation is critical, and, like any AD functions V&V team, the collaborating team is working towards software functionalities with quality and safety in mind. Testing large amounts of data is a critical part of this process, as it enables the team to identify and address any issues that may arise during testing. One of the challenges of achieving higher automation

Level of autonomy	
Level	Type of automation
0	No automation
1	Driver assistance Driving assist features
2	Partial automation Combined automated function
3	Conditional automation
4	High automation Certain condition
5	Fully automation All conditions

Table 2.1: *Levels of autonomy, 0-5, according to the Society of Automobile Engineers (SAE) [3], where a higher level of automation means more advanced autonomous driving functions.*

levels is ensuring that the software and hardware components of the system work seamlessly together. This requires a comprehensive understanding of the system's architecture and the ability to identify and address any issues that may arise during testing. An expensive toolchain, regarding time and cost, is not sustainable in the long term as autonomous functions become more advanced, which makes optimal software testing crucial.

2.1.1 Regulations of Autonomous Cars

The previously mentioned UN regulations set uniform guidelines for "automated lane-keeping systems" (ALKS) and apply to any manufacturer seeking certification in lane-keeping systems on public roads [6]. The thesis has looked at this regulation throughout the study, for example when choosing which scenarios to test. The regulation was developed in 2021 and the aim was to set uniform guidelines for ALKS. ALKS makes history as the first international regulation for "level 3" vehicle automation in heavy vehicles on roads [11], which is defined as "conditional driving automation" [3]. In summary, the regulation sets up requirements for ALKS that any manufacturer must follow to receive their certification and legally activate their ALKS on public roads.

The regulation specifies that the driver must be able to take over anytime when the ALKS is activated and that it could be active when the vehicle is on a motorway or in traffic jams, but not at speeds exceeding 130 km/h [6]. For a driver to have access to the ALKS, the regulation requires three things; a driver availability system that looks at parameters such as if the driver is on the seat or if the seat belt is being used, a black box that stores data in the event of a crash, and strict software and security updates. Additionally, the regulation mandates that the road must be closed to pedestrians and cyclists and that there must be a separation of traffic in both directions.

On top of the previously mentioned requirements, the regulation also provides several different scenarios [6] which the ALKS has to handle safely. In particular, the regulation points out three critical scenarios which are: cut-in, cut-out, and de-acceleration. These scenarios are described in more detail in Section 2.3.1. To legally deploy autonomous driving functions on the roads, companies must be capable of managing critical scenarios that frequently occur and poses a threat to safety related to ALKS. The regulations also include a driver reference model, which serves as a baseline for how a good and alert driver would behave in the defined scenarios. Given the infinite parameter sets, it would be practically impossible to design functionalities that never fail, however, ALKS systems are expected to at least avoid collisions in situations where the reference driver model would.

Because of this, software plays a big role in fulfilling these requirements and regulations, since it is the software that makes decisions about how the vehicle should travel based on sensor data. Regulations like the UN regulation No. 157 have important implications for software development in automated driving and demand that the implemented software must perform consistently with, or better than, the reference model. To meet these standards set by the regulation, software developers must ensure that their software is designed to meet specific safety and performance standards. This requires testing and validation under a wide range of conditions and traffic scenarios to ensure that the software meets the necessary criteria. The regulations provide a framework for ensuring the safety and quality of these functionalities, and understanding their requirements is essential to developing effective and reliable software for new and upcoming AD functions.

2.2 Test Environments for AD Functions

There are three types of testing involved when talking about autonomous cars; virtual testing, physical testing, and a mix of both. This Section dives more into detail about one of them, virtual testing, since the toolchain being studied is utilizing it. The tests are run in cloud environments which this Section goes through as well.

2.2.1 Model-based Design

Evaluating and assuring the safety of self-driving cars has long been a difficult task, prompting researchers in the automotive field to seek relevant architecture for systematic testing. When looking at different testing methods utilized for autonomous cars, three types of methods primarily used within the industry have been identified; Modeling and simulation testing, closed-track testing, and lastly open-road testing [12]. They all have different levels of test control and fidelity, and sometimes multiple techniques can be utilized at once, or in parallel, to evaluate a system. The study is focused on modeling and simulation testing since the AD functions V&V team uses ”software-in-the-loop” (SIL) testing in their testing toolchain.

2.2.1.1 Modeling and Simulation Testing

The method of modeling and simulation is a well-established tool within the industry that covers analysis, design, acquisition, and training [13]. Modeling and simulation offer several major advantages, such as controllability, predictability, repeatability, scalability, and efficiency [12], all of which are important when working with autonomous vehicles. One of the most popular models in the automotive industry is the V-model, also known as the verification and validation model, which is an extension of the waterfall model where processes are sequential. By utilizing virtual simulation technologies at various levels of abstraction, several types of testing can be conducted, such as model-in-the-loop (MIL), software-in-the-loop (SIL), hardware-in-the-loop (HIL), and vehicle-in-the-loop (VIL) [12][13][14]. This thesis only focuses on SIL testing since it is the testing method used by the AD functions V&V team in the case study of their toolchain.

SIL simulation operates by utilizing a portion, or the entirety, of the underlying autonomous driving system program to drive the physical response to stimuli [13]. For instance, modeled sensor input could be processed and used to generate world modeling, decision-making, and motion-planning algorithms. The output of the mentioned algorithm could be used to drive the virtual vehicle by feeding the data into the vehicle model. The use of SIL in system development provides several advantages. Firstly, it enables the system's internal state estimate to match the actual state, leading to increased accuracy [15]. Additionally, SIL is a cost-effective and safe option that can significantly reduce development time [16]. In comparison, traditional simulation and hardware emulators are becoming less versatile and less effective [17]. SIL can be used in both the design and testing phases, which is not possible with the other two techniques mentioned. SIL also offers testing ease, speed, flexibility, repeatability, and code reuse [17]. These last five benefits will be explored further in the paragraph below.

Firstly, SIL configurations can simulate networks in situations when field testing is impractical or complex, such as with vehicle testing, which simplifies and creates new opportunities for testing. Secondly, employing a SIL configuration allows for keeping all the details of the higher level, where the software being studied is situated while modeling the underlying network components in the simulator. As most of the time is consumed by lower-level functions, abstracting some of the lower-level models in the simulator can make the process go faster. Thirdly, since there are no hardware components at play in this architecture, making changes to the design or the simulated network will not become an issue. Fourthly, unlike the environment of a physical test bed, the environment in a setup with SIL is controlled which makes it easier to reproduce it internally or for external parties. Last but not least, the SIL architecture enables code reuse, which has the potential to lower development costs since it does not require duplicating efforts to create software for both testing and deploying the design. In the end, it is the same software that is applied in both steps to achieve this.

In contrast, there are four specific challenges with SIL [17]. The first challenge

is scalability since large-scale demonstrations require a lot of computers and space to set up. The second challenge is that modifying the software during the study is not desirable, as it often happens for the software to smoothly interact with the simulator. This kind of defeats the purpose of SIL since its original goal was to move away from issues of model validation. The third challenge is breaking end-to-end connections [17], regarding data transmission protocols that enable data to flow from a network to a destination. Two of these protocols are the transmission control protocol (TCP) and the user datagram protocol (UDP), which ensure data flow in different approaches [18]. SIL is preferred when it comes to connection-less applications, but not for connection-oriented applications like TCP, which requires breaking the end-to-end connections [17]. The last challenge is regarding timing. Event-driven simulators may execute events faster or slower than in real-time, and the time-driven software needs to be synchronized with the event-driven simulator.

2.2.1.2 Closed-Track Testing

Instead of running all tests in virtual environments, one could change that environment into a real-life scenario, constrained to a closed track. Physical testing entails utilizing physical objects or representations of obstacles to test a production-level vehicle, in contrast to simulation, which might not accurately reflect reality. The performance of the vehicle is assessed using real sensors on the autonomous car and software that is running on the target platforms, making it more precise and realistic. The technique of closed-track testing offers multiple benefits; controllability, improved fidelity, transferability, and repeatability [12]. For instance, as per the improved fidelity aspect, the involvement of physical and functional objects in the testing creates a more lifelike scenario. Still, as with everything, the technique has some drawbacks such as limited variability, personnel and equipment needs, potential hazards, and being prolonged and costly. There is no doubt that a real and physical environment including fast-driving cars has potential risks for the participants involved in the testing.

2.2.1.3 Open-Road Testing

The last type of technique primarily utilized in the automotive industry for testing is open-road testing, also called a ”real-world-laboratory” [12]. Today, several parties actively test their autonomous cars on public roads, such as Tesla and Google. The use of public roads allows a researcher to test autonomous vehicle systems in a wide scope of real-world conditions related to operational design domains, operational environments, and driving scenarios. This type of testing is not feasible when it comes to closed-track testing since it is limited in scope and variety of scenarios. However, the relevant drawbacks of open-road testing are the lack of controllability, replicability, repeatability, and scalability. Moreover, as a safety-critical situation, where for instance, a collision occurs, is relatively rare in open-road testing, scenario-based simulation testing is necessary [19]. Therefore, scenario-based simulation testing makes it easier to test critical scenarios, a scenario in which the autonomous vehicle’s actions potentially cause a collision or nearly result in one.

2.2.2 Cloud Computing

Cloud computing is a new and upcoming technology that is gaining popularity due to its ability to easily deploy flexible applications [20]. It simplifies the process of acquiring and releasing resources to a running application, while only charging for the resources used (pay-per-use model). Clouds are used by companies for various purposes, such as running batch jobs or hosting web applications.

When talking about resource scaling in cloud environments, it can be either of two things; horizontal or vertical [20]. Horizontal scaling, also known as scaling out or in, involves adding or releasing server replicas that run on virtual machines. On the other hand, vertical scaling involves modifying the resources given to a virtual machine that is already executing, such as increasing or decreasing its CPU or memory allocation. Most cloud services only support horizontal scalability because operating virtual machines cannot be instantly modified without restarting them.

Not only has the technology created new business opportunities, but it has also significantly affected the testing of software [21][22]. One major impact is the advent of testing as a service (TaaS) in cloud environments, which is a new service model in which a provider offers software testing services for a specific application system in a cloud infrastructure based on the demands of customers. In this model, the provider undertakes the software testing activities on behalf of the customers, thus creating a new business opportunity. Generally, the benefits of cloud testing are; elasticy, reliability, dynamism, and the pay-per-use model, amongst other things [20][21][22][23].

Currently, the AD functions V&V team is executing their toolchain and autonomous function testing in the cloud due to the limited computing power on local machines. However, due to the massive amount of data involved in autonomous driving scenarios, running in the cloud has been proven to be a very expensive task.

2.3 Case Study Context

To provide context for the assessment and optimization of the AD functions V&V team's testing toolchain, this section will cover the background information required for understanding the study environment.

The study is in collaboration with the company Volvo Cars. The company is a well-known car manufacturer and has played an important role in the economy and culture of Sweden, particularly Gothenburg. Volvo Cars is widely recognized for its strong focus on safety and is striving to achieve a collision-free future with the help of advanced technology [24].

The partnering team is referred to as the AD functions V&V team, and their responsibilities include verifying and validating autonomous functionalities developed and delivered by an outsourced company. The team is still in its early stages and tests a few functionalities but intends to scale up and add more features in the future.

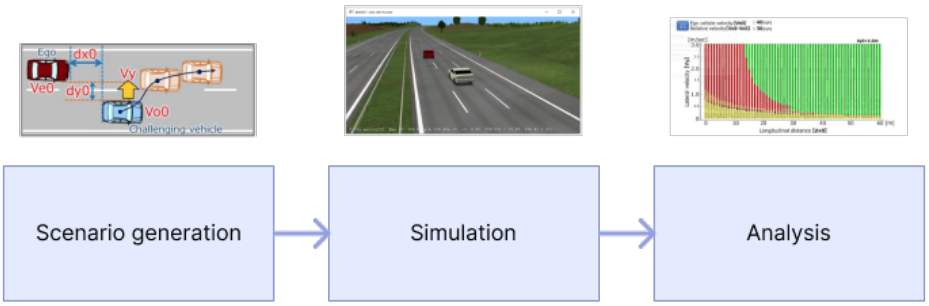


Figure 2.1: *Workflow of the collaborating company's toolchain. On the top: to the left a visualization of a scenario, in the center a screenshot of a simulation visualised with Esmini, and to the right the driver reference model graph from ALKS of a cut-in scenario which analyse if there was a collision or not.*

To scale up and achieve the future vision of Volvo Cars, which is for no collision to occur, the team will need to scale up their testing as well. Today the testing of their functionalities is primarily performed with the assistance of a toolchain. The team makes use of outsourced code for different parts of the toolchain, specifically the AD functionalities. This will be referred to as model-in-the-loop (MIL) in this thesis. Running software without this outsourced code will be referred to as without model-out-of-the-loop (MOL). The current toolchain has three primary modules: scenario generation, simulation, and analysis, which can be seen in Figure 2.1.

2.3.1 Scenarios & Scenario Generation

A scenario is a sequence in which an action or event occurs [25]. In this context, a scenario would be a traffic situation, for example, a vehicle cutting in front of the ego vehicle (the main vehicle being monitored, equipped with automated functionalities). When talking about scenarios in this context, there are different types with different abstraction levels: functional, logical, and concrete scenarios [25][26]. The highest degree of abstraction has a functional scenario, which is described in everyday language. For instance, deceleration, cut-in, and cut-out. A logical scenario's parameters contain values in the form of ranges or distributions, making it less abstract than a functional scenario. A concrete scenario has specified parameter values, thus making it the least abstract of the three.

Scenario generation is, as the name suggests, the activity where permutations of scenarios are generated. The scenarios can be generated in a variety of ways by variations in different parameter values, for instance, differences in parameters such as speed_target, and velocity_lane_change. The collaborating team generates scenarios with multiple sampling techniques, including random sampling. Random sampling is the act of generating concrete scenarios based on the ranges and distri-

butions from logical scenarios [26], as can be seen in Table 2.2 where the parameter speed consists of a range. There are other ways of generating scenarios, for example, combinatorial testing and mutation testing.

The study examines four scenarios, with three of them classified as critical scenarios according to UN regulation No. 157, as explained in Section 2.1.1 (cut-in, cut-out, and de-acc). This is because UN regulation No. 157 is one of the largest international regulations teams must comply with to deploy their AD functions on public roads. The fourth scenario is called ”lane change”, which involves the ego vehicle changing lanes. Figure 2.2 illustrates the critical scenarios, which can be explained as follows; cut-in involves a vehicle cutting in front of the ego vehicle from the adjacent lane in the same direction; cut-out involves a vehicle in front of the ego vehicle cutting out from the same lane into an adjacent lane, while there is a third vehicle that appears behind the cutting out vehicle; de-acc involves a vehicle in front of the ego vehicle suddenly decreasing its speed in the same lane. In Table 2.2, one can see an example of a logical scenario, cut-in, in which four different parameters are present. However, not all parameters are required. There are also many more parameters one could utilize when running different types of scenarios, for instance, the longitudinal position of vehicles, speed profiles for vehicles, the end time for scenarios, and the type of road the vehicles drive on. One could use fewer variables as well.

Example scenario (cut-in)	
Parameter	Type/value
scenario_type	cut-in
speed	range(2, 10, 60)
speed_target	15
velocity_lane_change	5

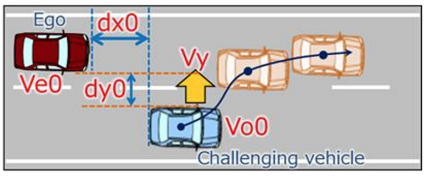
Table 2.2: Example of a cut-in scenario as written by the collaborating team, with possible parameters used for generating a scenario.

2.3.2 Simulation

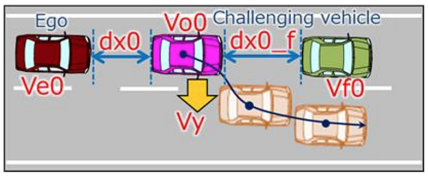
After the relevant scenarios have been generated, they need to be simulated in a controlled environment to accurately evaluate the performance of the autonomous system. The simulation enables the testing of a myriad of scenarios, some of which would impossible to recreate in real-world testing, as mentioned earlier. To simulate the scenarios the collaborating company utilizes the open-source open simulation interface (OSI), and esmini, an open-scenario player.

2.3.2.1 Open Simulation Interface (OSI)

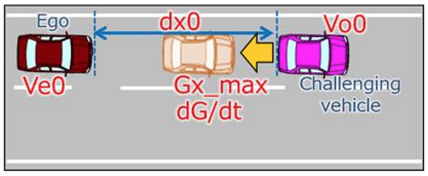
Open simulation interface (OSI) is a standardized software interface for the exchange of simulation data between different simulation tools in the automotive industry [27].



(a) Cut-in scenario



(b) Cut-out scenario



(c) De-acc scenario

Figure 2.2: Three critical scenarios, cut-in, cut-out, and de-acc, from UN regulation No 157 [6].

OSI is created by the Association for Standardization of Automation and Measuring Systems (ASAM) and is widely used in the automotive industry for the development and testing of automotive systems. OSI standard defines a set of rules and protocols for the exchange of data between simulation tools. It provides a common language and format for simulation data, allowing different simulation tools to communicate with each other and exchange data.

To increase the accessibility of simulators, connected to driving, for developers, it is necessary to establish generic and standardized interfaces that link the function-development framework with the simulation environment. The OSI standard provides a simple way to ensure compatibility between automated driving functions and the various driving simulation frameworks available [27]. OSI also allows for greater interchangeability and reusability of models across different testing techniques, such as HIL, MIL, or SIL, which is becoming increasingly relevant as multiple techniques are employed to achieve a single test goal [28]. As systems being tested grow in complexity, maintaining the same model across tests can reduce the complexity of testing, which in the end will increase transparency.

The AD functions V&V team utilizes OSI to simulate relevant scenarios, according to regulations, for instance, UN regulation No. 157, see Section 2.1.1. The simulation creates ground truth files, which are representations of the simulated environment that describes all simulated objects in the global coordinate system over time intervals. These messages are derived from data that is accessible to the

simulation environment. The global coordinates are used to calculate and validate the positions of the involved vehicles, for instance, whether the ego vehicle left the lane.

2.3.2.2 Esmini

Esmini [29] is an open-source software tool that serves as a scenario player for testing and validating advanced driver assistance systems (ADAS) and autonomous vehicles. Esmini supports OSI, which means that one can import and use driving scenarios that are created in the OSI format, and it provides a platform for creating and executing realistic driving scenarios in a virtual environment, which can help developers and researchers test and evaluate the performance of their systems under different driving conditions. For instance, developers can customize their scenarios with different weather conditions, like rain and fog, road layouts, like the number of lanes and traffic signals, and traffic patterns, like heavy or light traffic.

In the AD functions V&V team's SIL testing toolchain Esmini is utilized to simulate driving scenarios in a virtual environment, where the ego vehicle is either controlled by the AD functions or the reference driver, so that the performance of the two can be compared. Esmini is also used for sanity checks when designing and implementing scenarios in earlier stages. Figure 2.3 shows a visual representation of the scenario cut-in, by the use of Esmini.

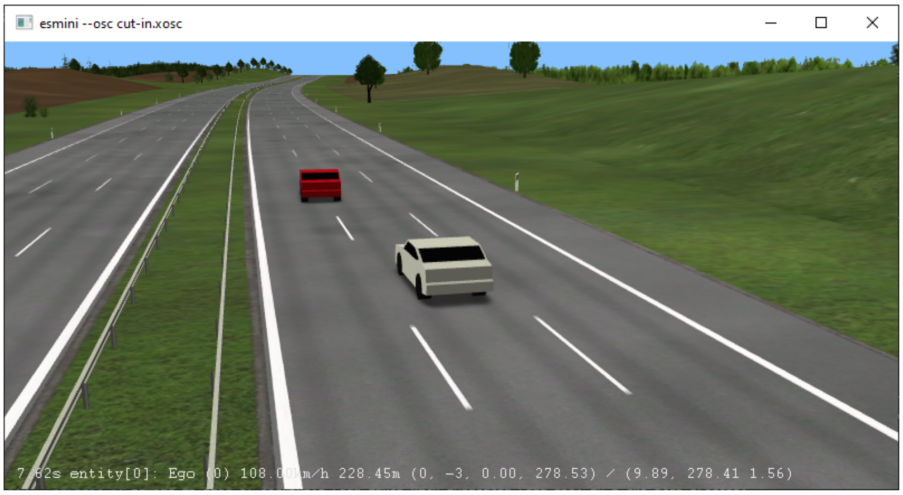


Figure 2.3: *Visualization of the simulation of a cut-in scenario by the use of esmini, where the red car cuts-in in front of the ego vehicle.*

2.3.3 Analysis of KPIs

After the simulation, the final part of the toolchain consists of analysing the scenario. It plays a crucial part in the verification and validation of AD functions and the purpose of this module is to provide insights into how the functions perform in different scenarios and to assess their performance.

To assess this, the AD functions V&V team utilized the concept of key performance indicators (KPIs); a set of values that allows the team to analyse and compare the performance regarding specified objectives [30]. In this case, the KPIs make it possible to determine whether the autonomous vehicle's behavior meets the required safety and performance standards based on regulations. The analysis module assists in the improvement of AD functions by comparing the results of the KPIs to the previously mentioned driver reference model, which can be seen in Figure 2.4. The team can also generate the same kind of graph from their data. This enables the team to determine whether the vehicle's behavior is an improvement compared to a human driver. This is essential for identifying areas that need improvement and optimizing the AD functions for safety and performance.

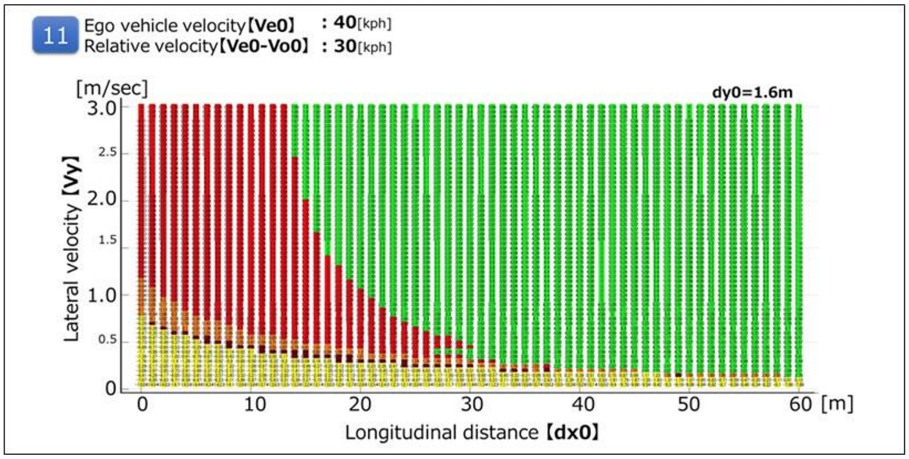


Figure 2.4: *Representation of a cut-in scenario and the results of the KPI collision analysis of the driver reference model from ALKS regulation [6]. The green points in the Figure stand for no collision, the red for a collision (front, back), the orange for collision (side), and the yellow for a cut-in scenario from behind because of a slow ego vehicle, deemed non-relevant for the scenario.*

For the collaborating team, each KPI is most often implemented as a function that returns a boolean value, which can be either true or false depending on whether the system meets a particular performance criterion. These KPIs are designed to capture various aspects of the system's behavior which are crucial for variables such as safety, for instance, its ability to stay within its lane, avoid collisions with other vehicles or obstacles, respond appropriately to traffic signals and signs, and more. Currently, the KPIs are checked after the simulation has been run, as the toolchain is run sequentially.

Overall, KPIs play a critical role in the toolchain and the analysis because they provide a quantitative measure of the system's performance and help to ensure that the function meets the required regulations. By regularly refining and improving the KPIs used in the toolchain, the team can assist in the development of safe and reliable autonomous vehicles that are capable of operating and handling critical real-life traffic scenarios.

2.4 C++ versus Python

Due to the high cost of processing large amounts of data, the AD functions V&V team is currently exploring various opportunities to optimize the execution time of their toolchain, see Figure 2.1. As the analysis module is currently written in Python, there has been interest in refactoring and rewriting it into a more efficient programming language. Since the simulation module is written in C++, they believe that exploring this option would be logical, as it could reduce execution time.

C++ is a general-purpose and object-oriented programming language that evolved from the C programming language [31]. Compared to C, C++ provides additional features such as virtual functions, operator overloading, and named functions, among others. Different programming languages are suitable for different purposes; for example, Java is ideal for web development, Python is great for rapid prototyping, and C is well-suited for GUI (graphical user interface) [32]. One of C++'s major advantages is its efficiency, unlike Python, which is generally slower [33].

Many factors contribute to C++'s quick performance. The first is that C++ is a statically typed language, whereas Python is a dynamically typed language [32]. Python does not require data types to be spelled out, so instead the interpreter must validate every type as it runs. In contrast, in C++, variables are saved with their corresponding fixed data type, eliminating the need for runtime type checking. Moreover, C++ is a compiled language, whereas Python is a scripting language [32]. Compiled languages do not need any time to parse code; however, scripting languages require an interpreter to translate the code before it can be executed [32].

On the other hand, multiple sources claim that Python has better readability than C++ [33][34], that C++ is more complex [35], or that Python is more user-friendly [36]. A comparative analysis of the two languages showed that Python is generally regarded as having a higher level of readability because of its syntax [33]. Python is designed to be closer to a human's natural language, making it more intuitive and easier to learn, particularly for individuals who are new to programming. When discussing C++, it has a more complex syntax and structure, which can make it more challenging to read and understand, especially for beginners. Furthermore, code written in Python for adding two integers and printing "Hello World" is relatively shorter than the C++ code. Logically, this trend could have a big impact on readability when scaling up the code. Another study indicates similar findings, that because of the language's simplicity, "pseudocodish syntax", easy-to-learn environment, and higher abstraction, a higher proportion of students would choose Python over C++ when it comes to their first programming language [34].

3
Research Design

The thesis will follow the action research methodology in the way it is described by Staron [37]. As the author explains it, the process is divided into five phases; diagnosing, action planning, action taking, evaluation, and learning, see Figure 3.1. The five phases are conducted iteratively, meaning that they are performed in multiple cycles. This study will be iterated three times and the cycles are presented in Figure 3.2.

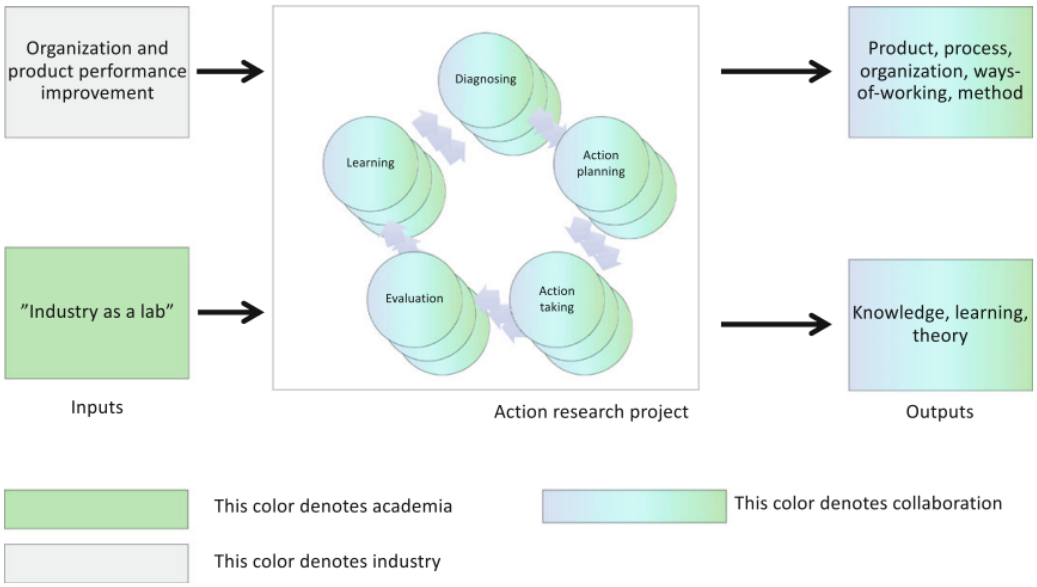


Figure 3.1: *Action research in the context of software engineering, as proposed by Staron [37].*

The purpose and goal of action research in a software setting are to concentrate and establish software engineering practices in their specific context, thus hopefully affecting these practices positively within that context [37]. The context of the research is the testing toolchain or even the automation process in which the AD functions V&V team tests their autonomous functionalities. Since the context, which in this case is the automation testing process, is the center of attention, the thesis fits into the research methodology action research.

Staron proposes the notion that action research is a collaboration between academic researchers and industrial practitioners and as a result, the method produces research results that contribute to both the industry and academic theories. The

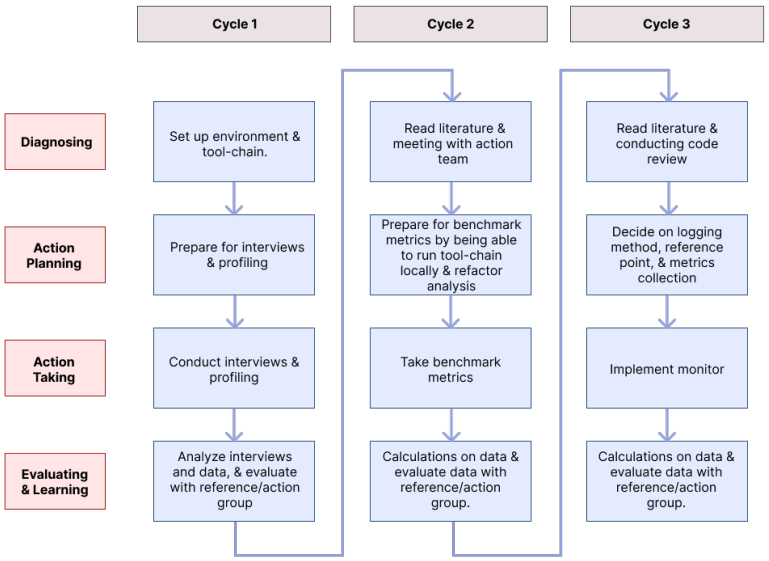


Figure 3.2: *The workflow and specifics of each phase of action research; Diagnosing, Action Planning, Action Taking, and Evaluating and Learning, all performed thrice iteratively.*

practice that the thesis is aiming to affect positively is the process of testing autonomous functionalities. The primary goal of the thesis is to improve the AD functions V&V team's automated testing process, more specifically to shorten the execution time of the testing toolchain. In keeping with action research, the secondary goal is to support academic research by aiming to provide findings and insights that can be generalized to a larger context.

The actors involved in the research are an action team and a reference team. The action team consists of the thesis writers and an industrial practitioner from the AD functions V&V team. The same practitioner is also the only member of the reference team. The action team's responsibility is to execute the research. They do this by planning the research, taking action, and evaluating the results [37]. The responsibility of the reference team is to support the action team with guidance and feedback. An actor worth mentioning is a team that will be referred to as a simulation team. As the name might suggest, this team is responsible to develop the simulation part of the toolchain. This team acted as an unofficial reference team by providing advice about the simulation module.

Ideally, one should also involve a management group. The management group's responsibility is to take important decisions regarding the organization. As the thesis is only 18 weeks long, the scope will not include implementing the solutions into the actual organization. The changes made by the action group will only be imple-

mented locally and if the collaborating company team wants to implement it into the organization, they will have to do that outside the scope of the thesis. Consequently, the function of a management team won't be significant.

Action research is fitting for this thesis because of the close relationship that has been established with the industrial collaborating partner which action research demands, thus active participation from industry partners is not an issue. Furthermore, there is a clear need to understand the problem at hand, whilst focusing on improving future work for the collaborative partner, which goes in line with Staron's guidelines about action research.

As stated, the thesis follows the action research methodology which operates in different chronological phases; diagnosing, action planning, action taking, evaluation, and learning. The workflow for the entire thesis in phases and cycles can be seen in Figure 3.2. The study consisted of a handful of different methods and procedures which are: set up and installment of the virtual environment and toolchain, semi-structured interviews, profiling of the toolchain, thematic data analysis, collecting and cleaning data, generating graphs in Rstudio with R, literature studies, writing and refactoring code in Python and C++, and lastly conducting a code review. Cycle 1 mainly aims to answer RQ1 and RQ2. The interviews provided a more in-depth understanding of the everyday work of an AD functions V&V team as well as that the members working directly with the testing toolchain could share their challenges and needs. The profiling, which was decided to do based on the interviews, gave better insights into the toolchain but also concrete information on where the main issues lie. The other two Cycles aim to answer RQ1.1. In the second Cycle, a solution was proposed and prepared which was later implemented in Cycle 3. All procedures and methods for each cycle, as well as the results of that cycle, are described more in detail in Chapter 4, Execution and Results.

4
Execution and Results

4.1 Cycle 1

The aim of Cycle 1 is to better understand the problems with and the needs of the AD functions V&V team's testing toolchain as well as get to know the organization. The cycle mainly seeks to answer RQ1: *What characterizes an optimal testing toolchain for an AD functions Verification and Validation team in the automotive industry?*, and RQ2: *What are the challenges of testing AD functionalities in automated processes in the automotive industry?*. To answer these questions the following procedures were taken; conducting interviews, collecting and cleaning data, performing a thematic analysis, and profiling the testing toolchain.

4.1.1 Cycle Goal & Research Procedure

In the first cycle, the **diagnosing phase** mainly consisted of understanding the main issues with the testing toolchain as well as getting a better insight into the practitioners' everyday tasks. To do this, the systems utilized to uphold the toolchain were explored briefly after setting up the necessary environments and downloading the software by walking through the toolchain step by step. This gave more hands-on insights into what the practitioners deal with on a day-to-day basis. The testing toolchain is installed virtually in ubuntu [38], and is accessed by a remote desktop connection. As can be seen in Figure 4.1, the toolchain is composed of three primary components; scenario generation, simulation, and analysis.

The first module generates permutations of scenarios, which the second module simulates and passes to the final module for analysis and comprehension. The scenarios simulated in the toolchain utilize the OSI environment, but they can also be visualized using Esmini. The simulation module is developed by another team within the collaboration company, which will be referred to as the simulation team. This module is written in C++. The purpose of the toolchain is to test the AD functionalities, however, an external company is developing the functionality of the AD software and the code is delivered as binary files.

The analysis module consists of KPI calculations used by the AD functions V&V team to understand how vehicles behave in scenarios. The collaborating team currently uses around 30 KPIs including if there was a collision, and if the vehicle kept enough distance from the lane markers or other vehicles. The KPIs are also used to determine whether the behavior of the vehicle is safe enough to fulfill the require-

ments of various regulations. The analysis makes use of the scenario generation output files by collecting basic information about the parameters of the scenario. The analysis also uses the simulation ground truth output files to evaluate the KPIs by calculating objects' positions and speed at every time step. The analysis module is written in Python.

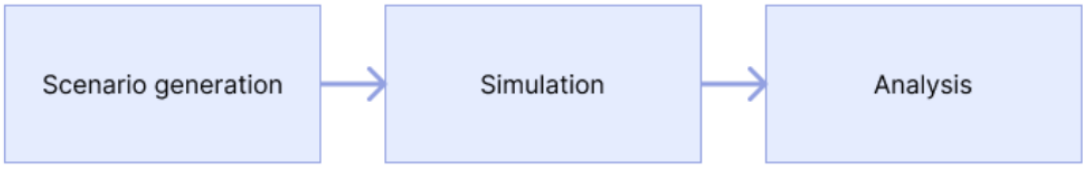


Figure 4.1: *Workflow of the collaborating company's toolchain. The toolchain executes sequentially and starts with a generation of scenarios. Afterwards, the scenarios are simulated and lastly analysed utilizing KPIs.*

During the **action planning phase**, a first interview was scheduled with a function designer. The function designer was chosen because of his close relationship to the toolchain since his everyday tasks involve verifying and validating it. Due to the unavailability of several individuals closely involved with the toolchain, such as those on vacation or parental leave, it was difficult to identify suitable practitioners to interview. Therefore, the plan was for the function designer to identify new candidates to be questioned, as per the snowball method [39], a non-probability sampling technique where the sampling technique is completely based upon judgment. The technique was used because there was a hope that the practitioners would have the most knowledge about the processes and limitations regarding the toolchain and whom to talk to since they are the ones interacting with it and with relevant employees on a day-to-day basis. Based on this technique two more candidates, a data engineer, and a developer, could be identified based on the initial participants' expertise.

The interview questions, which are listed in Appendix A, were chosen to be semi-structured and planned to follow the five steps in the way as it is described by Preece [40] in chronological order; introduction, warm-up, main session, cooling-off period, and a closing session. In addition, it was made sure that the questions were neutral and not overly long or complicated since the author also emphasizes that point. The interview was chosen to be semi-structured due to the freedom it offers; unstructured interviews are exploratory [40], which is appropriate when it's important to grasp the context of the issue at hand, on the other hand, structured interviews are appropriate when participants do not have a lot of time available, and questions have been identified. The main aim of the interviews was to get a better insight into how the toolchain works as well as to understand the problems with it.

The toolchain was profiled to identify bottlenecks and get a better understanding of what takes up time. Measurements regarding time were planned to be collected

to grasp which components of the toolchain were the most costly. The data only served as a way of understanding the problem at hand and act as a guide of where the scope should be. The data is therefore not official measurements to be used as a baseline. Three different tools were used to profile; the python module ”time”, which was used to collect the execution time end-to-end as well as the execution time of each module, the python profiler ”cProfile” which was used to investigate what internally within the modules takes time, and lastly the visualization tool ”Snakeviz”, which was used to visualize the results from cProfile. The profiling can be seen in Figure 4.2 where the python module ”time” was used between point 1 & 2, point 2 & 3, point 3 & 4, and lastly point 1 & 4. The profiling was made on the three scenarios: cut-in, cut-out, and de-accelerate, as they are the three critical scenarios from the ALKS - UN Regulation No. 157, presented in Section 2.1.1. Moreover, the profiling was performed locally on the computer. Due to this, each type of run was profiled three times to increase the accuracy of the results. Three profiling runs were considered sufficient as the purpose of profiling was to gain an understanding and guide the scope of the thesis, rather than provide official baseline measurements. Additionally, the results remained relatively stable across all runs, which indicated that more runs were unnecessary at this early stage of the thesis.

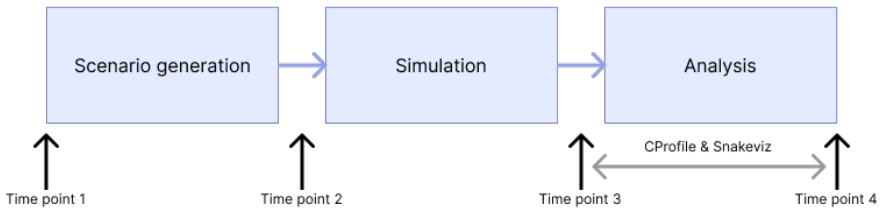


Figure 4.2: *Overview of the profiling of the toolchain by the use of CProfile and Snakeviz, with each of the time points used to capsulate each module and the entire toolchain.*

4.1.2 Cycle Execution & Results

The **action taking** phase began with three interviews, which were recorded and transcribed for a thematic analysis of the qualitative data generated. It was challenging to carry out a full analysis with several themes and codes due to the limited number of participants, but also because the interview questions varied between participants since their roles are different. Consequently, the thematic analysis was limited to a single theme: *Problems with the toolchain*. The authors of this thesis assigned codes to specific portions or complete answers as a pair to establish patterns in the qualitative data from multiple perspectives. The questions regarding how the toolchain works only supported the group with more details of the research context. The theme *Problems with the toolchain* consists of the codes: long execution times and cost (money), which can be seen in Table 4.1.

Theme: Problem with toolchain	
Long execution time	Mentioning of long feedback times
	Mentioning of large amount of data
	Mentioning of need for code optimization
	Concrete mentioning of long execution times
Cost (money)	Concrete mentioning of money
	Mentioning of that running in cloud cost money

Table 4.1: *Results of thematic analysis with the theme "problem with toolchain". The two identified codes can be seen to the left, and their underlying causes can be seen to the right.*

The thematic analysis and mapping of codes revealed that the main issue with the toolchain is execution time from end to end. When running large enough batches of permutations, it takes far too long, prompting the AD functions V&V team to run the permutations in the cloud which with the current toolchain is prohibitively expensive. Soon, the team will run these large batches regularly which led the team to run a feasibility study of a smaller batch of scenarios to assess their situation. The team wanted to know the number of scenarios needed to test, how much it would cost, and if the toolchain even can handle such a large amount of scenarios. The participant explains it:

"*A month back we did a feasibility study of a smaller batch, and said that for such a batch we would like to run hundreds of millions of scenarios (in the future) /.../, just to see if our toolchain is even capable of handling this based on the outcome.*"

The participant goes on to talk about the current situation and how the current state of the toolchain is unsustainable:

"*... it would be incredibly expensive (to run all of the scenarios) and might even jeopardy the AD as a whole because we still need to have higher business value than developing costs.*"

Given that the AD functions V&V team currently pays for the number of nodes per time unit, the execution times are the only direct cause of the cost. By reducing the execution time the team will either occupy the nodes for a shorter time which results in a lower cost, or they will be able to execute the toolchain on fewer nodes which also results in lower cost. Moreover, when running small batches of permutations, the long execution time also affects the workflow of the employees due to long feedback times. The interviews also revealed that there are trade-offs between readability and execution time, and running the toolchain locally or in virtual machines:

"*There's always a trade between optimizing code maximally to achieve the fastest speed while also maintaining the readability of the code. It should have a quite general purpose so that it is quite easy to scale up and expand and add new features.*

But also, it should be specific enough to just not confuse the people using the tool."

The last comments had to do with the toolchain's long execution time and the specific cause of code optimization. A participant mentions the importance of weighing specific programming languages:

"*A quite important thing about this is that simulation is run in C++, and the post-analysis is run with Python. Generally, C++ is way faster at doing these kinds of things.*"

On top of the interviews, two rounds of data measurements were collected. The first round of runs used the built-in Python module "time" [41] and aimed to identify which modules of the toolchain were the most time-consuming. The measurements used were the execution time, end-to-end. As mentioned previously, this only served as a way of understanding the problem at hand and act as a guide of where the scope should be and are not official measurements to be used as a baseline. Therefore, each run was only performed thrice, since there was no formal need for statistical validation this early into the process, and the results after three runs were relatively stable. The validation of the number of scenarios was carried out by consulting both the action and reference groups. This was done to ensure that the selected number of scenarios was suitable for the objectives of this phase, and the company's goals at this stage of the thesis. The time points used can be seen in Figure 4.2. The results from the first run of collected measurements can be seen in Table 4.2. The first interesting finding is that the execution times of the scenario generation module are significantly shorter when compared to the other modules. The scenario generation module takes less than 1% of the entire execution time for all runs, regardless of the number of scenarios (1 or 10) or scenario types (cut-in, cut-out, or de-acc). The scenario generation module was therefore deemed negligible in the scope of this thesis, as can be seen in Table 4.2.

Type of Scenarios	**Cut-in**		**Cut-out**		**De-acc**	
Number of Scenarios	1	10	1	10	1	10
Total run time (sec)	37	406	63	570	49	577
Scenario Generation run time (sec)	Negligible					
Scenario Generation % of total run time	Negligible					
Simulation run time (sec)	26	237	26	266	26	274
Simulation % of total run time	71	58	42	47	53	47
Analysis run time (sec)	9	167	36	302	21	302
Analysis % of total run time	25	41	57	53	44	52

Table 4.2: *The Table depicts the execution time collected from three concrete testing scenarios, namely cut-in, cut-out, and de-acc. Each run consisted of one or 10 scenarios but was performed thrice. As a result, the total of the simulation and analysis results may not equate to 100%. The average of the three runs is presented in the Table.*

Method	Cumulative time in % (sec)		
	Cut-in	Cut-out	De-acc
Main	100% (16.30)	100% (27.30)	100% (35.40)
Loop	100% (16.30)	100% (27.30)	100% (35.40)
Road coord 3	49% (8.00)	49% (13.50)	55% (19.50)
Road coord 1	47% (7.66)	48% (13.00)	41% (18.40)
Road coord 2	47% (7.65)	48% (13.00)	41% (18.40)
KPI conversion 1	30% (4.84)	31% (8.33)	32% (11.30)
Road coord 4	25% (3.99)	23% (6.24)	25% (8.78)
KPI conversion 2	24% (3.91)	23% (6.40)	26% (9.12)
init	18% (2.89)		
Road coord 5		17% (4.52)	18% (6.55)

Table 4.3: *The 10 methods in the analysis module with the highest cumulative time for running the scenario cut-in, cut-out & de-acc. Since different scenarios handle different calculations, not all methods are utilized in each scenario and are represented by a blank cell in the Table.*

Another finding of the first collection of measurements is that both the simulation module and the analysis module are time-demanding components. The main difference between the two is that in the simulation module the scenarios, regardless of type, take a similar amount of time to simulate, see Table 4.2, row "Simulation run time (sec)". In contrast, the different scenario types take a different amount of time to analyse. See Table 4.2, row "Analysis run time (sec)". Because of the variations in analysis times, the percentage that the simulation module takes of the total run time also differs.

Since the results showed that the simulation and analysis modules are time-demanding components, a second round of measures collection was made on the two modules. It consisted of profiling the modules internally and aimed to understand what within the module allocates the most time. This was done using the profiler cProfile and the visualization tool Snakeviz. The measurements collected were a cumulative time which is the time spent in a method as well as the time spent in the methods that this method calls [42]. In Table 4.3 the 10 methods with the highest cumulative time can be seen for each scenario in the analysis module. The profiling revealed that the methods for calculating global coordinates to road coordinates (Road coord 1-5) and converting scenario frames to KPIs (KPI conversion 1 & 4) take up the most time, as shown in Table 4.3. The Table only shows the top 10 methods with the highest cumulative time, but other methods with high cumulative time not included in the Table also correlate with Road coord or KPI conversion. The road coord methods are methods that calculate objects' relative positions in the simulation environment. It's noteworthy to mention that multiple employees working with the toolchain have mentioned these kinds of time-consuming conversions and have suggested investigating this issue.

When profiling the simulation module the action and reference team was initially

perplexed by the results. The teams could not understand the results as they consisted of only a few methods and components that each was revealed to be very time-consuming. Since the teams couldn't explain the results assistance was sought from the simulation team where multiple members were consulted. Ultimately, it was concluded that the profiling results were inaccurate and that it was not possible to profile the simulation module without assistance from the company writing the outsourced code. This is because the simulation module includes outsourced code being delivered as binary files. However, after speaking with the team developing the module, it was revealed that the main bottlenecks regarding the time within this module were the outsourced code. This claim will be further explored later in the study in Chapter 4.2. The action team pointed out that the outsourced code heavily affects the time it takes to simulate scenarios. However, they also mentioned that they didn't know whether this code automatically needed to be time-consuming or whether it could be optimized. Nevertheless, the outsourced code was not something that could be affected in this study. Therefore, the action team recommended putting energy into trying to run the simulation locally without model-in-the-loop and the outsourced code.

The **evaluation and learning** process continued with a meeting involving the reference and the action team that aimed to confirm the results and gain a better understanding of them. The questions asked during the meeting can be found in Appendix A.3. Firstly, the meeting confirmed that the problem had been correctly identified, and concluded that the long execution time is the core issue of the toolchain. High costs in terms of money are also a problem, but the cost of running the toolchain would be decreased by reducing the time it takes to execute it, thereby tracing back the problem to the long execution times. This resulted in the first conclusion of the cycle.

Conclusion 1: *Decreasing the time it takes to execute the toolchain is a priority.*

Moreover, the action team emphasized the fact that the usability of the toolchain is of utmost importance. The execution time end-to-end needs to be shortened, but not at the cost of the readability of the code, since the toolchain should be easily understood by employees. If the toolchain's code can't be easily understood, even though it executes fast, it would cause problems for future employees, and the toolchain couldn't be integrated into the larger organization. Proper and clear continuity when it comes to the structure of code, and names, for example, are some ways to battle this issue. This discussion resulted in the second conclusion of the cycle which will be kept in mind in the other cycles when changes will be implemented.

Conclusion 2: *The readability of the toolchain is important for all employees working directly or indirectly with it, to decrease the risk of any implications regarding scalability in the future.*

The meeting continued with a discussion of the profiling data from round one. The action and reference team agreed that the low execution times of the scenario gener-

ation module are negligible in the scope of this thesis. Furthermore, they confirmed that the major issue regarding execution time lies in the simulation module as well as the analysis module. This led to the third conclusion of the cycle.

Conclusion 3: *The Scenario Generation module is no longer relevant to investigate further.*

The meeting continued with discussing the data from the profiling round two. The action team's perception of what takes up time within the analysis module was in agreement with the conclusions of the profiling. That will say, that converting global coordinates to road coordinates and scenarios frames to KPIs takes up the majority of the time within the analysis module. Since the analysis module takes up a significant amount of time and the results suggest that it is outsourced code that takes up a lot of the rest, the action and reference team decided on focusing on optimizing the analysis for future cycles.

Conclusion 4: *Optimizing the Analysis module is the focus for Cycle 2.*

Since the action and reference teams do not work with the simulation module other than running it, they could neither confirm nor deny whether the most time-consuming parts of the simulation are outsourced codes. However, they agreed with the simulation team's recommendation to try to run the toolchain without the outsourced software, leading to Conclusion 5 of the cycle. This was decided for three reasons: 1) If one cannot affect the code, there is no need to keep analysing it. 2) By comparing the execution times with and without the outsourced software, one can confirm or deny whether it is the outsourced software that takes up the majority of the simulation module's time. 3) By removing the outsourced software, one is not dependent on the outsourced company supporting the implementations.

Conclusion 5: *Running the toolchain without outsourced Model in the Loop (MIL) should be investigated in the next cycle.*

4.2 Cycle 2

The aim of Cycle 2 is to propose and investigate a solution to the pinpointed problem and challenges in Cycle 1. Cycle 2 mainly seeks to answer RQ1.1: *How should a testing toolchain be optimized for an AD functions Verification and Validation team in the automotive industry?*. To answer this question the following procedures were taken; refactoring code, collecting baseline measures, cleaning data, and generating graphs in RStudio with R.

4.2.1 Cycle Goal & Research Procedure

The **diagnosing phase** of Cycle 2 began with a discussion with the simulation and AD function V&V team where the aim was to decide on which solution to start

investigating, and what can be done to decrease the execution time in the analysis module without jeopardizing the readability of the toolchain. They proposed a plan to implement, what they call, a ”monitor”. A monitor has been discussed by the action and simulation team for a long time and the teams are optimistic that such a solution will have a significant impact on the execution time. The monitor is simply the concept of evaluating and calculating KPIs in real-time in the simulation module, hence eliminating the need for a stand-alone analysis module. The workflow of the toolchain with a monitor implemented can be seen in Figure 4.3. Currently, when analysing a scenario, the analysis module must do complex and time-consuming calculations to calculate the KPIs, however, a significant amount of this information is already directly accessible within the simulation module. For instance, as observed in the Cycle 1 profiling, the methods regarding calculating the objects' positions take a lot of time, although, the simulation has these positions in real-time.

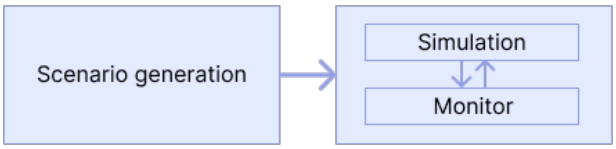


Figure 4.3: *Workflow of the toolchain with the monitor implemented. The scenario generation module to the left remains the same, whilst the analysis module will be removed and replaced by a monitor within the simulation module.*

The plan is to implement a proof of concept of a monitor, specifically, a ”stay-in-lane monitor” that will analyse the KPIs related to whether the vehicle stays within its lane, which can be seen in Figure 4.4. The monitor will operate by determining whether the vehicle remains in its original lane at each time step of the simulation. This means that the monitor can directly and simultaneously log KPIs in the simulation module, eliminating the need for a separate analysis module. Aside from the fact that the analysis module would no longer need to perform most of the complex calculations, the monitor concept is expected to reduce execution time because the analysis will be converted from Python to C++, which is generally a faster programming language [32]. This was also mentioned in the interviews in Cycle 1, see Section 4.1.2.

The remainder of the cycle aims to prepare for the implementation of the monitor concept to investigate whether the monitor is possible and worth implementing. The first step of the **action planning** phase was to be able to run the entire toolchain without the model in the loop, as per Conclusion 5 in Cycle 1: *”Running the toolchain without outsourced Model in the Loop (MIL) should be investigated in the next cycle”*. This was mainly done to prepare for the implementation of the monitor as the toolchain currently doesn't have support for implementing a monitor

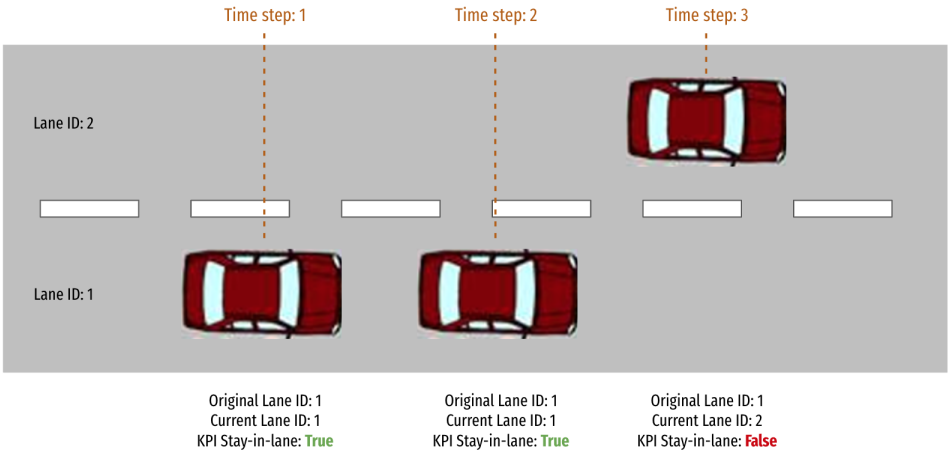


Figure 4.4: *Visualization of the concept "stay-in-lane monitor" which checks whether the vehicle has stayed within its original lane at each time step and updates the KPI "Stay-in-lane" to either True or False.*

with the outsourced code and to verify the statements from the experts, that it is the outsourced code within the simulation module that takes up the minority of the time. The entire toolchain had never been run without the outsourced code before, which required refactoring of the collaborating team's toolchain to be able to run it. The refactoring consisted of small changes such as changes in paths and running scripts. The simulation module, which had previously been run in the cloud, was also locally installed.

To accurately compare the baseline measures to the measures of the implemented monitor, the analysis can only include the KPIs that will later be evaluated in the monitor. As a result, non-relevant code from the analysis module had to be removed before the measures could be collected. Furthermore, it was decided that the measurements would be taken similarly to the profiling in Cycle 1, which meant that the built-in Python module Time would be used to calculate the execution wall time. However, the scenario generation module was excluded as per Conclusion 3 in Cycle 1: *"The Scenario Generation module is no longer relevant to investigate further."*.

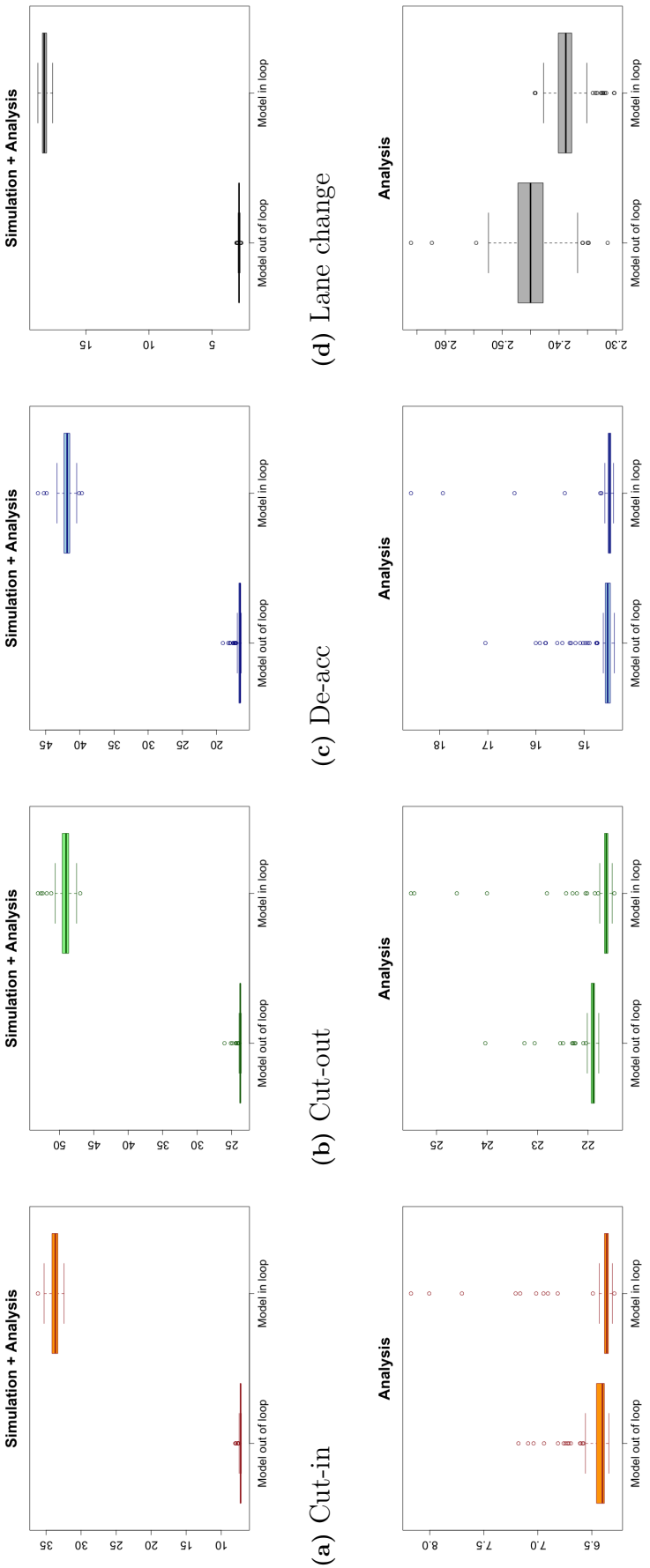
Moreover, the three previously mentioned scenarios, cut-in, cut-out, and de-acc, were chosen again this time around since it had been established that these are representative scenarios in Cycle 1. In addition, a "lane-change" scenario was added because none of the other three scenarios included the ego vehicle changing lanes, which is necessary for testing the monitor. Two different measurements were taken: with model in the loop and without model in the loop. Both of them were run on all four scenarios 100 times. Meaning that we ran 100 times for cut-in model in loop, 100 times for cut-in model out of loop, 100 times for cut-out model in loop, 100 times for cut-out model out of loop and so forth.

4.2.2 Cycle Execution & Results

The **action taking** of baseline measurements can be seen in Figures 4.5 (a-d). They represent the total (simulation module + analysis module) execution times of the toolchain for the scenarios cut-in, cut-out, de-acc, and lane-change. The box plots to the left represent the times for running the toolchain without the model in the loop, and the right box plots represent the execution times for running the toolchain with the model in the loop. As can be seen in Figure 4.5 there is a notable difference in execution time for all four scenarios between running the toolchain with and without the outsourced model in the loop, in which the time of running without it is shorter.

Figures 4.5 (e-h) represent the execution times of running only the analysis module for the scenarios cut-in, cut-out, de-acc, and lane-change. Note that the vertical scale is narrower here compared to the plots representing the total execution times. As can be seen in Figure 4.5 there is no big difference in execution time when comparing running the analysis module with (right box-plots) and without the model in the loop (left box-plots).

In contrast, the simulation module appears to be responsible for the majority of the difference in total execution times between running with and without the model in the loop. As can be seen in Table 4.4, in the rows ”Difference (MIL - w/o MIL)”, the difference between the column ”Simulation time (sec)” and ”Total time (sec)” is small. Another interesting finding is that when the module in the loop is removed the remaining code left that the AD functions V&V team can affect is the analysis module. The analysis module, therefore, becomes the most time-consuming part of the toolchain that the team can influence by a big margin.



(a) Cut-in (b) Cut-out (c) De-acc (d) Lane change

(e) Cut-in (f) Cut-out (g) De-acc (h) Lane change

Figure 4.5: *Benchmark metrics of the comparison of running the toolchain for the four different scenarios; cut-in (green), cut-out (orange), de-acc (blue), and lane-change (grey). In Figures (a-h) the box plot to the left represents the execution time for the model out of the loop, and the box plot to the right represents the execution time for the model in the loop. Figures (a-d) show the execution time for running the simulation and analysis, whilst the bottom row (e-h) shows the execution time for running the analysis on its own.*

Cut-in			
	Total time (sec)	**Simulation time (sec)**	**Analysis time (sec)**
Model in Loop	33.69 - 33.80	27.25 - 27.35	6.42 - 6.48
w/o Model in Loop	7.27 - 7.30	0.82 - 0.83	6.45 - 6.48
Diff (MIL - w/o MIL)	26.38 - 26.53	26.41 - 26.53	-0.06 - 0.03

Cut-out			
	Total time (sec)	**Simulation time (sec)**	**Analysis time (sec)**
Model in Loop	49.07 - 49.26	27.30 - 27.43	21.74 - 21.87
w/o Model in Loop	23.77 - 23.83	1.83 - 1.84	21.94 - 22.00
Diff (MIL - w/o MIL)	25.24- 25.49	25.46 - 25.60	-0.26 - (-0.07)

De-acc			
	Total time (sec)	**Simulation time (sec)**	**Analysis time (sec)**
Model in Loop	41.85 - 42.01	27.28 - 27.40	14.53 - 14.64
w/o Model in Loop	16.66 - 16.75	2.05 - 2.07	14.61 - 14.69
Diff (MIL - w/o MIL)	25.09 - 25.35	25.21 - 25.35	-0.16 - 0.03

Lane-change			
	Total time (sec)	**Simulation time (sec)**	**Analysis time (sec)**
Model in Loop	18.25 - 18.30	15.87 - 15.91	2.38 - 2.39
w/o Model in Loop	2.85 - 2.86	0.41 - 0.41	2.45 - 2.45
Diff (MIL - w/o MIL)	15.39 - 15.44	15.46 -15.50	-0.72 - (-0.06)

Table 4.4: *95% confidence intervals for the four different traffic scenarios; cut-in (orange), cut-out (green), de-acc (blue), and lane-change (grey). For each scenario, the Table displays a 95% confidence interval for the execution time in seconds of running the simulation, the analysis, or both parts of the toolchain, with the model in the loop, the model out of the loop, and the corresponding differences between them.*

The **evaluation and learning** process continued with an evaluation meeting involving the action and reference team. The questions asked can be found in Appendix A.4. The main objective of the meeting was to confirm the findings and results of the cycle, as well as to gain a better understanding of them. The teams reviewed the data from Figure 4.5 and Table 4.4, and concluded that the difference in execution time when running the toolchain with and without model-in-the-loop lies entirely in the simulation module. The meeting also confirmed the team's initial assumptions from Cycle 1 that the outsourced code (the model in the loop) consumes the majority of time in the simulation module. This led to the first conclusion of the cycle.

Conclusion 6: *The outsourced code (the model in the loop) consumes the majority of time in the simulation module.*

The action and reference team also discussed the reasons why the analysis module is not affected by running without the model in the loop, whilst the simulation module is. It was concluded that when the model in the loop is removed, the vehicle will follow a more simple dummy function that does not require complex calculations, leading to less time spent in the simulation module. However, the analysis module only looks at the positions of the vehicles, lanes, and surroundings at a given time, without considering how they got there. As a result, the calculations in the analysis module remain the same regardless of whether the model in the loop is present or not, and therefore there is no change in execution times.

Furthermore, the teams noted that when the outsourced code is removed, the only code left that the AD functions V&V team can influence is the analysis module. Among the factors that the team can influence, the execution times of the analysis module are the most time-consuming. This has further strengthened the team's confidence that the monitor concept will effectively reduce the execution times, as the concept removes the need for an analysis module. The action and reference team concluded that the results are still in favor of implementing a monitor and identified it as a priority for future work in Cycle 3. This leads to the last conclusion of the cycle.

Conclusion 7: *Implementing a proof of concept for the "monitor" should be investigated in the next cycle.*

4.3 Cycle 3

The aim of Cycle 3 is to implement the proposed proof of concept of a "stay-in-lane monitor" and evaluate whether the concept would decrease the execution time of the toolchain. Cycle 3 mainly seeks to answer RQ1.1: *How should a testing toolchain be optimized for an AD functions Verification and Validation team in the automotive industry?*. To answer this question the following procedures were taken; conducting a code review, writing code in C++, collecting data on execution time, cleaning

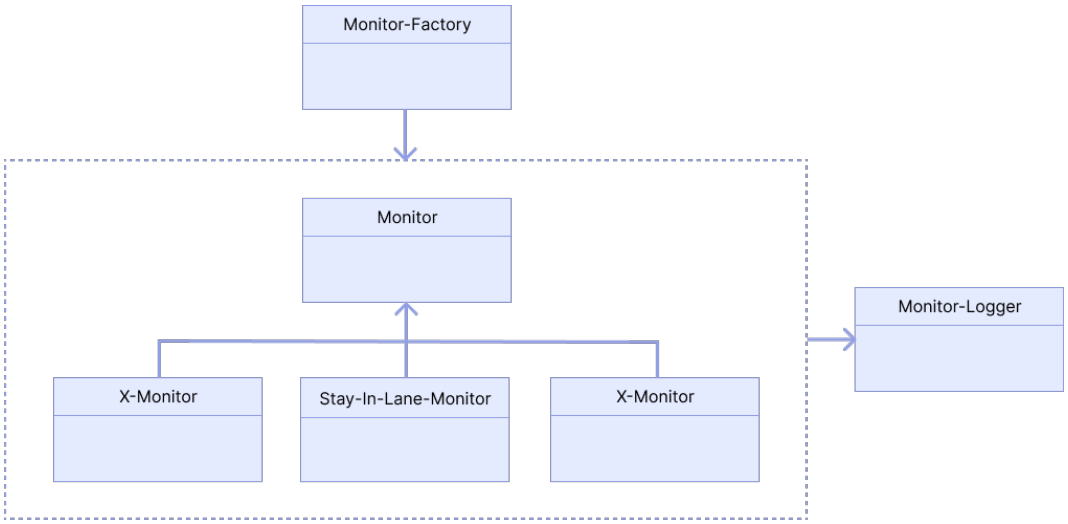


Figure 4.6: *UML diagram of the structure of monitors and loggers in the AD functions V&V teams simulation module. The monitor factory creates different types of monitors, which has a monitor logger.*

data, and generating graphs in RStudio with R.

4.3.1 Cycle Goal & Research Procedure

The **diagnosing phase** of Cycle 3 began with an investigation of the AD functions V&V team's current approach to three different aspects; support of monitors, logging of KPIs, and reference point used to determine if the vehicle has left the lane. To do this, a code review was conducted on the parts of the code related to these three aspects, to understand the hierarchy and structure of the classes, more so than specific lines of code. As previously mentioned in Cycle 2, the monitor concept had been discussed within the action and simulation team for a long time, therefore, the simulation module already had some implementations that support the concept of a monitor. For instance, a monitor factory and shells for the "stay-lane-monitor", and a monitor logger. The use of a factory pattern creates continuity of how to create more monitors in the future which would have positive effects on the readability. Since readability was found to be a characteristic of an optimal testing toolchain in Cycle 1, the monitor solution was therefore found to be suitable. The results of this code analysis can be seen in Figure 4.6, which includes the proposed "stay-in-lane monitor". The figure depicts the UML diagram of how the system supports creating monitors through the use of a factory pattern, and an existing system for connecting a logger to monitors.

The analysis module was analysed in more detail, as that is how the AD functions V&V team has been logging their KPIs until now. The current solution logs KPIs in the form of a text file with separating commas (CSV), in the analysis mod-

ule. However, as the analysis module is planned to be removed and replaced by a monitor within the simulation module, the logging of the KPIs has to be moved as well. Conversations with the action and reference group brought up that there was no preferred method of logging KPIs as long as it works well and that one should be able to log with multiple monitors at the same time.

A deeper investigation was also required to determine the reference point that would be used to detect if the vehicle had changed lanes. Upon reviewing the relevant code, it was discovered that the middle point of the vehicle was currently being utilized as the reference point in the ”monitor-shell”. This was also confirmed by the action and reference group. However, they also mentioned the possibility of altering this reference point to a bounding box, which means that any point of the vehicle that crossed over the lane marker would count as the vehicle not staying within its lane.

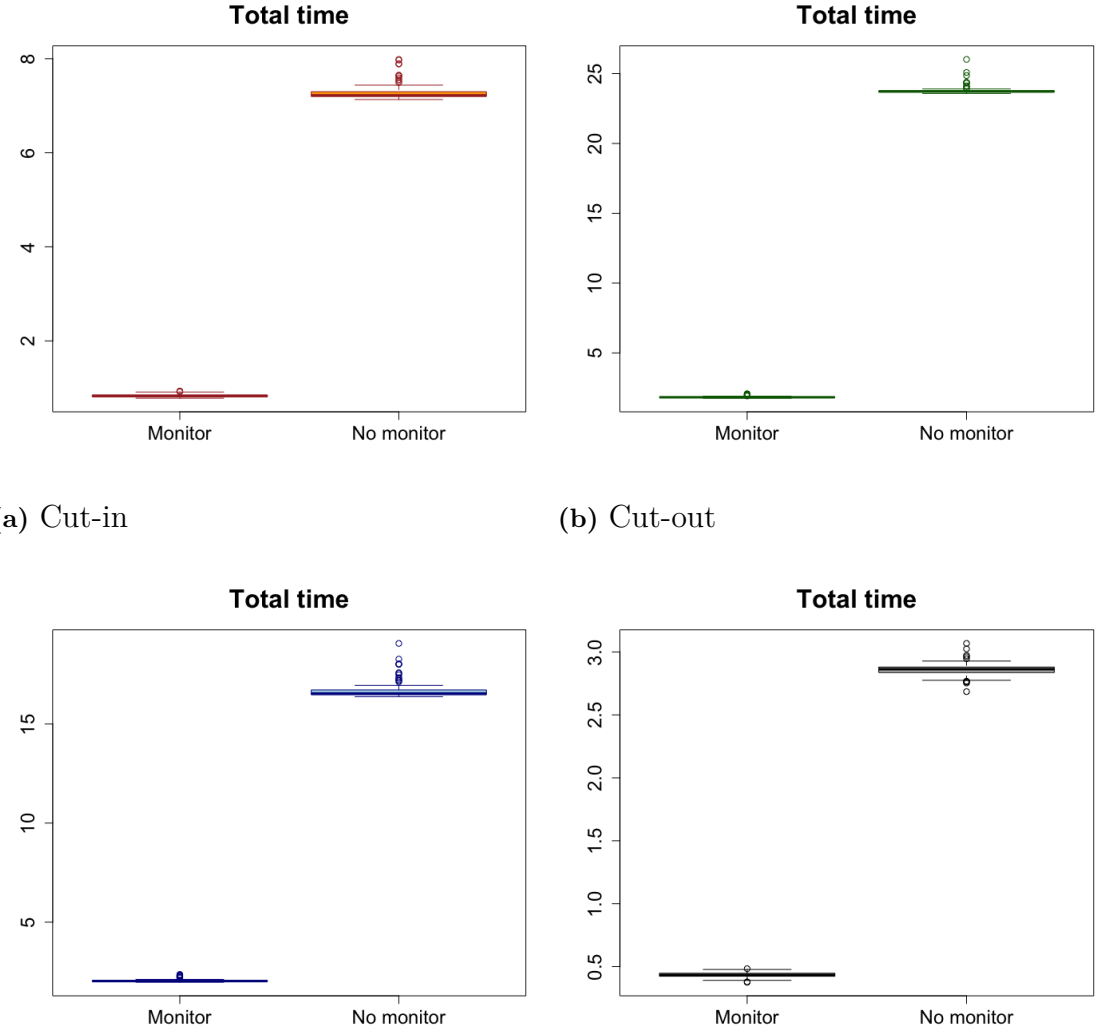
The action group's **action plan** is to implement the ”stay-in-lane monitor” that utilizes the middle point of the vehicle as a reference point and text files for logging KPIs. After weighing both options of reference point, bounding box, and middle point, the latter was chosen. This is because of the tight time constraints of the thesis. Since the solution is only a proof of concept of a monitor, it would be wasteful to spend too much time on specifics. The logging method was decided on the same premises. As there was no preferred way of logging by the reference and action team, we chose to log in text files, as it is a simple but effective way of logging.

The action planning also included deciding what data was necessary and interesting to collect in the action taking. The group decided on two measurements, which were collected a hundred times for the benchmark data in Cycle 2. They were taken on the toolchain without the model in the loop and with the scenario generation module excluded. The first measurement the group decided on was the execution time of the simulation with the newly implemented monitor. This data would be compared to the benchmark data; model out of loop simulation and analysis execution times, to evaluate if the monitor concept would decrease the execution time of the toolchain. Furthermore, the data was also compared to the benchmark data; model out of loop simulation (without monitor) execution time, to evaluate how long time just the monitor takes. The second measurement the group decided to collect was the execution time of the simulation with the monitor, however, this time the whole simulation would stop when the vehicle left the lane. This comparison would enable the action group to evaluate if there are any benefits to having support for stopping the simulation once a KPI is full-filled.

4.3.2 Cycle Execution & Results

The **action taking** consisted of implementing the ”stay-in-lane monitor” and collecting the various data described above. The **evaluation and learning** consisted of statistical calculations on the data, like the mean, and an evaluation meeting with the action and reference group. The first measurement collected was the execution time for the simulation with the implemented monitor. This data was compared

with the benchmark data taken in Cycle 2, the execution time for the simulation and analysis, and was used to evaluate whether there is a decrease in execution time with the monitor. The results can be seen in Figures 4.7, where each box plot to the left represents running the toolchain with the proposed solution, the model out of the loop, and a monitor (simulation + monitor), whilst the box plot to the right represents only running the toolchain with the model out of the loop (simulation + analysis), as in Cycle 2. The result for each scenario is described below.



(a) Cut-in

(b) Cut-out

(c) De-acc

(d) Lane-change

Figure 4.7: *Comparison of running the toolchain without the scenario generation module and the model in the loop. The box plots to the left represent the execution times of simulating with the monitor and the box plots to the right represent the execution times of running the simulation and analysis. Note that the scale vertical scale is different on all Figures and that the scenario generation is excluded.*

Firstly, looking at the cut-in scenario, as can be seen in Figure 4.7a, the median time for running the toolchain before implementation of a monitor (right box-plot) was 7.22 seconds, compared to afterward (left box-plot), which was 0.83 seconds. This represents a decrease of median execution time by 88.50% with the implemented "stay-in-lane monitor". Secondly, looking at the cut-out scenario, as can be seen in Figure 4.7b, the median time for running the toolchain before implementation of a monitor (right box-plot) was 23.72 seconds, compared to afterward (left box-plot), which was 1.82 seconds. This represents a decrease of median execution time by 92.33% with the implemented "stay-in-lane monitor". Thirdly, looking at the de-acc scenario, as can be seen in Figure 4.7c, the median time for running the toolchain before implementation of a monitor (right box-plot) was 16.54 seconds, compared to afterward (left box-plot), which was 2.03 seconds. This represents a decrease of median execution time by 87.73% with the implemented "stay-in-lane monitor". Fourthly, looking at the lane-change scenario, as can be seen in Figure 4.7d, the median time for running the toolchain before implementation of a monitor (right box-plot) was 2.86, compared to afterward (left box-plot), which was 0.43 seconds. This represents a decrease of median execution time by 84.97% with the implemented "stay-in-lane monitor". A summary of the execution time for all four scenarios before and after the implementation of a "stay-in-lane monitor" can be seen in Table 4.5.

Scenario	**Before (sec)**	**After (sec)**	**Difference (%)**
Cut-in	7.22	0.83	-88.50
Cut-out	23.72	1.82	-92.33
De-acc	16.54	2.03	-87.73
Lane-change	2.86	0.43	-84.97

Table 4.5: *Median execution time in seconds for running the simulation + analysis (before) without model in the loop and the median execution time for running the simulation + monitor (after) without model in the loop on the scenarios cut-in, cut-out, de-acc, and lane-change. The difference in execution time before and after can also be seen in percent. Note that the percent decrease is after the AD is removed.*

The second data comparison was the median execution time of simulating without the monitor which was compared to the median execution time of the simulation with the monitor to investigate how time-consuming the "stay-in-lane monitor" was. The result can be seen in Table 4.6. As can be seen in column four there is a small difference between using a monitor or not, where the median execution time is slightly higher for the scenarios cut-in and lane-change.

Scenario	Without monitor (sec)	With monitor (sec)	With monitor - Without (sec)
Cut-in	0.8185	0.8301	0.0116
Cut-out	1.8179	1.8242	0.0063
De-acc	2.0226	2.0258	0.0031
Lane-change	0.4093	0.4340	0.0246

Table 4.6: *Median execution time in seconds for running the monitor. The run is without the outsourced code on the scenarios cut-in, cut-out, de-acc, and lane-change.*

The last data collected was the execution time of the simulation with the monitor implemented but the simulation stopped once the vehicle left its original lane. Since the lane change scenario is the only scenario that leaves the lane, this measurement was only collected on this scenario. This data was compared to the execution time of the simulation with the monitor but with no stop. The results can be seen in Figure 4.8 where the box plot to the left represent the simulation with the monitor without stopping and the right represents the simulation with the monitor which stops once the vehicle leaves the lane. The mean time for running the toolchain with a monitor (left box-plot) was 0.4339 seconds, compared to a monitor with the addition of stopping the simulation after the vehicle has left the lane (right box-plot), which was 0.2802 seconds. This represents a decrease in median execution time by 35.4%.

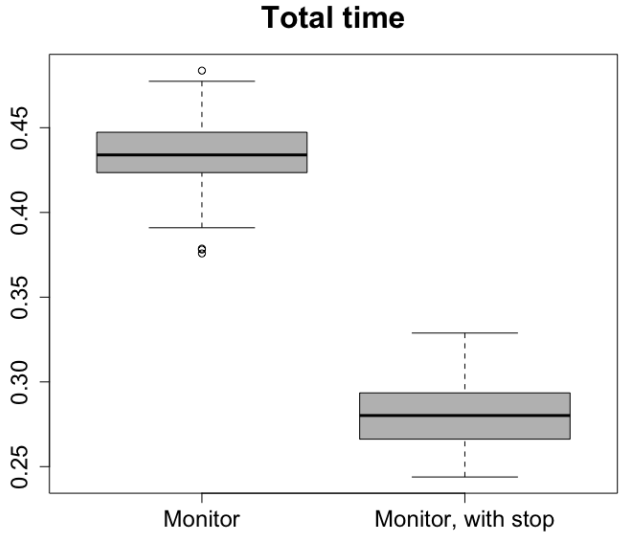


Figure 4.8: *Comparison of total times when running the lane-change scenario with model out of the loop and monitor (simulation + monitor), versus model out of the loop and monitor with stop after changing lanes (simulation + monitor).*

The evaluation and learning meeting of the cycle aimed to verify the results and evaluate the concept of a monitor. The questions asked can be seen in Appendix A.5. The meeting began by presenting the collected data to the action and reference

group, such as the Figures and Tables shown above, and addressing various variables that could affect the results. Firstly the action and reference team discussed the results from Figure 4.7 and Table 4.5, the execution times from the ”stay-in-lane monitor” compared to the execution times of the simulation and the separate analysis. The results and the action and reference team highly agree that there is a significant difference between execution times for all scenarios. The teams concluded from Table 4.5 that the implementation of a monitor has decreased the median execution time by at least around 84% for all four scenarios; cut-in, cut-out, de-acc, and lane-change. Because of this, the monitor concept was thought to be a great success and led the group to be optimistic about future expansions of the concept, due to the high decrease in execution time. However, to validate the findings we wanted to discuss two variables that could potentially skew the results with the action and reference team; choice of reference point and logging method.

When it comes to the reference point the monitor utilizes the center point of the car, which the action and reference team thought could cause issues further down the road. They explain that the results would be slightly more accurate if the monitor utilized the bounding box instead of the middle point of the car. Currently, by the use of a middle point, the car has only officially left the lane if more than half of the car has passed over the lane marking. This means that small parts of the vehicle could’ve passed over the lane marking, and the ”stay-in-lane monitor” perceived that the vehicle has stayed within its original lane. Nevertheless, the action and reference team don’t expect the calculations of using the bounding box as the reference point to involve any big conversions or loops, which led them to believe that the reference point would not affect the execution time significantly. Changing reference points was therefore deemed as future work for the action team since there was a time constraint to the thesis. The second variable that the action and reference team wanted to discuss was the choice of the logging method. The AD functions V&V team has currently been logging the KPIs by comma-separated values in text files which they’ve found to be simple yet effective and expressed that they did not want any overcomplications that could create unnecessary confusion and increase the execution time. A continuation of this type of logging was therefore appreciated since it has been proven to be a successful method to this day. The action and reference team, therefore, found no concerns regarding the logging method significantly affecting the execution time and results.

The meeting continued with discussing the time allocation of the ”stay-in-lane monitor”, more specifically the median execution times of the simulation compared to the simulation with the monitor, as can be seen in Table 4.6. The reasoning for this was to investigate whether the increase in execution time of a monitor outweighs the benefits of the decrease in execution time by the deletion of the analysis module. When looking at the results, the action and reference team was highly satisfied. For instance in Table 4.6, a monitor used on the cut-in scenario only adds 0.0116 seconds, or 1.4%, to the median execution time. For a cut-out scenario, this percent was 0.3%, for de-acc, this was 0.2%, and for lane-change it was 5.7%. The action and reference team concluded that the increase of at most 5.7% of the median execution

time does not make any significant difference, since the results from Figure 4.7 and Table 4.5 shows that the decrease in median execution time will be at least around 85%, which is a more substantial number.

The final data presented at the meeting was a comparison of operating the monitor with and without a stop when the vehicle changed lanes, as shown in Figure 4.8. The results were addressed briefly, where the team found the data interesting and investigated the potential use cases. The option to stop the simulation was appreciated by the reference team, and they expressed that the solution to do this was easily understandable and efficient. To summarize, all of the three results led the action and reference team to be very pleased with the results and was confident that the proof of concept is a success, leading to the first conclusion of the cycle.

Conclusion 8: *The "stay-in-lane monitor" decreased the execution time of the toolchain significantly when evaluating one KPI.*

The success of the "stay-in-lane monitor" was extremely clear to the reference team, but this sparked the conversation about the bigger picture of fully implementing the concept. For the concept to be beneficial to the AD functions V&V team, all KPIs that are time-consuming to calculate need to be evaluated in a monitor so that the execution time of the analysis module is significantly shorter. Optimally, the analysis module would be completely removed by moving all KPI evaluations into monitors. Based on this, the meeting discussed how the execution time would be affected if all KPIs were to be evaluated in a monitor. The simplest way to think about this would be to take the time of the monitor and multiply it by the number of KPIs. In that case, the time added to all the monitors would be short. The AD functions V&V team currently evaluates around 30 KPIs and the median time of the monitor lies somewhere between 0 to 0.0246 seconds, meaning that the execution time of evaluating all KPIs would be less than one second. This is a significant improvement compared to the analysis module which takes anywhere between 2.45 to 22, see Table 4.4, seconds dependent on scenario type when evaluating only one KPI. However, evaluating the concept is most likely not this simple.

Firstly, a monitor can evaluate more than one KPI. As mentioned previously, the collaborating team currently evaluates around 30 KPIs, which can be grouped naturally into 5-6 groups according to the action and reference team. Instead of adding 30 more classes that represent each KPI, it would make much more sense, both from a time allocation and readability perspective, to create 5-6 more general monitors which evaluate multiple KPIs. Moreover, the thesis has not studied how the execution time would be affected by a monitor evaluating multiple KPIs but the action and reference team don't believe that adding multiple KPIs into one monitor would have a big impact on the execution time. This is because the time allocation of the monitor is low, to begin with, see Table 4.6, and adding KPIs would not add any extra complexity.

Another factor that makes it hard to evaluate the concept as a whole is that each

monitor has a different complexity. The ”stay-in-lane monitor” is a relatively simple solution as it only assigns the lane ID at the first time step of the simulation and checks whether it has a new lane ID at the following time steps. It is difficult to properly assess how complex other monitors would be so at this time we can't say whether the ”stay-in-lane monitor” has an average complexity. Even though there is some uncertainty in how multiple monitors would affect the execution time the action and reference team are confident that the concept would still be majorly beneficial to the execution time since the margins between the proof of concept and the analysis module are large.

Finally, the reference and action group once again brought up the fact that they've had a suspicion for a long time that the utilization of monitors would help decrease the execution time of the toolchain but they were happily surprised by how positive the results were. However, due to a lack of capacity manpower, they have not been able to quantify the exact impact of the implementation of monitors to this day. Even though the exact results from Cycle 3 were new to the reference and action team, there was no doubt in the end that the concept of a monitor/monitors is worth implementing, based on the data from Figures 4.7 and 4.8, and Tables 4.5 and 4.6. This leads to the last conclusion of the cycle and the last conclusion of the thesis.

Conclusion 9: *The concept of a monitor/monitors could decrease the execution time of the toolchain.*

5
Discussion

In this Chapter, the findings of the thesis will be addressed and discussed. The first Section aims to answer the three research questions of the study. The second will address the validity of the research. The results and interpretations are discussed more in detail in Sections 4.1.2, 4.2.2, and 4.3.2; Cycle Execution & Results.

5.1 Answers of Research Questions

This Section aims to discuss the findings of the study by answering and discussing the three research questions. The answers for RQ1 are mainly based on the findings from Cycle 1, the answers for RQ1.1 is based on the findings and procedures from Cycle 1, 2, and 3, and lastly, the answers for RQ2 are mainly based on the findings from Cycle 1.

5.1.1 Research Question 1

RQ1: *What characterizes an optimal testing toolchain for an AD functions verification and validation team in the automotive industry*?

The entire thesis, but most specifically the interviews from Cycle 1, see Section 4.1.2, indicate that a low execution time, high readability, and low running costs are three important characteristics for an optimal testing toolchain for an AD functions V&V team in the automotive industry. When interpreting the results of RQ1 it is important to note that the results were mostly based upon the three interviews from Section 4.1.2. It could therefore exist other relevant characteristics to consider when talking about an optimal testing toolchain in the automotive industry. In the future, more in-depth interviews should be conducted to establish a more valid and comprehensive understanding of the characteristics. While the three interviews provided valuable insights, a larger and more diverse sample size would increase the robustness of the findings by gathering multiple perspectives and experiences. Fortunately, the insights gained from the interviews could many times be corroborated by the literature in Chapter 2, Background. The results of the three characteristics will be discussed in greater detail below.

Low execution time
Due to the strict safety requirements for autonomous functions in the automotive industry, functionalities deployed on the market must be thoroughly tested, as written

in Section 2.1.1. Manufacturers all over the world must ensure that their AD functions have undergone thorough testing, for instance, that all reasonable scenarios have been accounted for, and that the safety risks are sufficiently low to ultimately save lives and decrease the number of accidents that occurs. To guarantee this, the software needs to be tested on a large number of scenarios. To give an approximate number, the collaborating AD functions V&V team estimated around a hundred of million scenarios as seen in Section 4.1.2, though the exact number can differentiate between contexts. The scenarios consist of complex data, and thus testing large amounts of it takes time.

A low execution time is a crucial characteristic of an optimal virtual testing toolchain for an AD functions V&V team in the automotive industry. The faster the simulation runs, the more scenarios can be tested in a given time frame, allowing teams to identify and resolve issues faster, resulting in safer AD functions in vehicles and a competitive advantage on the market. SIL testing allows teams to identify and address issues before moving to physical testing, which can help to reduce the overall testing time and cost.

To exemplify, as mentioned briefly in the interviews in Section 4.1.2, the team has done a trial run to calculate the number of scenarios required to ensure the safety and quality of their functions, and how much it would cost and if the toolchain even can handle such large amount of scenarios. The results from the trial were that they would need to run hundreds of millions of scenarios to ensure the safety of a functionality. Due to confidentiality, the study can't mention exact numbers. Nevertheless, we can still do some approximate calculations. To not boost the numbers, we will calculate low thus using 100 000 000 scenarios.

The results from Cycle 2 reveal that a scenario, dependent on which type, takes anywhere from 18 seconds to 49 seconds in the toolchain. If we would need to run 100 000 000 scenarios, that would result in 500 000 hours (180 000 0000 sec) to 1 361 000 hours (4 900 000 000 sec). Since we are talking about such a large amount of scenarios, even a very small decrease in execution time would still result in a significant amount of decreased execution time. Looking at the solution of the "stay-in-lane monitor", the results from Cycle 3 reveal that a scenario is anywhere between 2.422 seconds to 21.96 seconds faster with the implemented monitor than without. Doing the same calculations, the monitor would decrease the total execution time by at least 67 278 hours (242 200 000 sec).

When talking about such long running times, the batches are run in parallel processes by buying computing power. As the interviews also revealed, execution time is highly coupled with a low running cost, so the decrease in execution time will directly affect the running cost as we can run more scenarios on the same amount of parallel processes given the same time frame.

Low running cost
Low running costs are critical for the sustainability and profitability of any AD func-

tions V&V team in the automotive industry. This is especially important for SIL testing of toolchains, which can quickly become expensive due to the large amount of data being tested, as the millions of scenarios required to be tested in the future are revealed in this thesis. Logically, a higher cost could therefore force teams to run their toolchain more seldom. As the participant from the interviews also mentioned in Cycle 1, this could diminish the value of the end product, and possibly jeopardize the future of the team, since developing costs need to be lower than the market value of the product for a company to profit.

On the contrary, if the cost of running the toolchain is low, the team will have no problem running it regularly, resulting in more tested functionalities, thus safer functionalities and faster deployment of these functionalities. Low cost is, therefore, something all companies in the automotive industry should strive for, to make the most profit out of their product or service.

High readability
High readability of the code in the toolchain is also an important characteristic of an optimal testing toolchain for an AD function V&V team in the automotive industry. The reason for this is that testing toolchains often involves multiple modules, and different team members, including developers, testers, and experts, who all need to be able to read and understand the toolchain and its results. If there are difficulties in how to manage and run the toolchain, it could lead to confusion, errors, and delays in the testing process, which can at last impact the quality of the final product. The discussions and results of the first evaluation meeting, see Section 4.1.2, indicate that no matter how fast the toolchain executes, it will not be able to be integrated into the organization if the code is not understandable by other current and future employees.

Moreover, if the results are difficult to interpret, team members may struggle to identify the root cause of any problems and come up with effective solutions. Establishing consistent procedures like design patterns, for instance, a factory pattern, comments, and formatting standards could therefore be examples to reduce the risk of decreased readability when scaling up their functions. This allows team members to quickly identify each code snippet's purpose and meaning, making it easier to understand and modify.

5.1.2 Research Question 1.1

RQ1.1: *How should a testing toolchain be optimized for an AD functions Verification and Validation team in the automotive industry?*

The answer to this research question can be divided into two parts; how a testing toolchain can be optimized in terms of how it should be composed, and how a testing toolchain can be optimized in terms of approach and method. This section will address both starting with the first mentioned. However, it's important to understand that the methods presented are based upon the experience with the AD

functions team at Volvo Cars and therefore can't be directly applicable to all research settings. Thus, there might exist methods for the optimization of toolchains in the industry that perform even better. Still, the method produced great results and could be used as guidelines when optimizing and assessing similar toolchains.

A testing toolchain used by an AD functions V&V team in the automotive industry can be optimized by analysing the scenarios while they are being simulated, allowing components to be computed simultaneously rather than sequentially. The measurements collected in Cycle 2 of the study indicate that analysing scenario simulations can be time-consuming and thus analysing them in a separate module after the whole simulation is unnecessary time spent. The solution presented in the study, a monitor, removes the need for a separate analyse module and moves the analysis into the simulation module.

Because each organization and AD functions team has different internal objectives and aims to test different functionalities, there is no general or best way to compose a testing toolchain used in the automotive industry. Since the testing toolchains are composed in a variety of ways there is not possible to claim that the solution presented in this study will have a positive impact on all testing toolchains used by an AD functions V&V team in the automotive industry. Even though the proposed solution might not apply to all contexts, the study has presented a general approach to how to investigate and optimize a testing toolchain which consists of the following four steps:

1. Conduct interviews

Conducting interviews and talking with employees that work directly or indirectly with the toolchain can provide valuable insights into the organization, understand the team's needs and get to the core issues of the toolchain. This can offer an overview of how the toolchain is set up, what challenges there are to optimizing the toolchain, and what efforts have already been made to overcome these challenges.

2. Profile toolchain

Profiling the toolchain is an effective approach to gaining valuable insights into the time allocation of different components and modules. A first good step is to get an overview of the time allocation of the toolchain by collecting the execution time of each module externally. By getting a brief understanding of the time allocation of each module teams can exclude modules at an early stage that are not worth putting more effort into. The profiling continues with profiling the remaining modules internally, meaning that the profiling aims to understand what within each module takes up the most time. By getting a better understanding of the time allocation internally, teams can identify bottlenecks and inefficiencies within the code which can be resolved and thus enables teams to put their efforts into components where they will be the most effective. Combined with the interviews, the challenges of optimizing a specific toolchain are now more established, making it possible to start looking at concrete solutions.

3. Find solution

The third step is to propose a specific solution based on the profiling and interview results. However, before implementing any changes, it is recommended to take benchmark measurements to establish a baseline performance.

4. Evaluate solution

To be confident in if the solution was successful or not it's necessary to evaluate it. A first good step is to validate the result with knowledgeable employees to make sure that the results are first and foremost correct and that the team has not missed any important aspects. Secondly, collecting new measurements after the implementation and running statistical tests will provide objective measurements and validate the effectiveness of the proposed solution.

5.1.3 Research Question 2

RQ2: *What are the challenges of testing AD functionalities in automated processes in the automotive industry*?

The results and discussions of Section 4.1.2 indicate that strict regulations and safety requirements, a large amount of data, and trade-offs are three challenges for testing autonomous drive functionalities in automated processes in the automotive industry. The same set of reasonings regarding the validity of the results with RQ1 can be applied to this research question as well since they are based on the same three interviews. Thus, there may be other challenges in testing AD functionalities in automated processes in the automotive industry. The results of the three challenges will be discussed in greater detail below.

Regulations & safety requirements

The first challenge of testing autonomous drive functionalities in automated processes in the automotive industry concerns safety regulations and standards, as mentioned in Section 1.1, and 2.1.1. The deployment of AD functions on public roads requires all companies to meet specific safety requirements, which could differ a lot depending on the country or region the company plan to launch its functions. Following these regulations is therefore not optional, forcing companies to comply and develop their AD functions to satisfy these requirements. To do this, companies must test their autonomous systems rigorously. SIL testing has been proven to be fitting for situations like this, see Section 2.2.1.1, to simulate and analyse millions of complex scenarios to ensure the safety and reliability of the system to keep future users safe. This does not only take up a large amount of time, but also cost a large amount of money for such things as maintaining, running, and setting up the testing functionalities virtually.

Moreover, as the automotive industry continues to advance and autonomous vehicles become more prevalent, regulators are continuously updating safety regulations and standards to reflect the latest technological advancements and local laws. Making sure that the organization is keeping up with these updates could be a significant

challenge, as compliance with new regulations may require further testing and adjustments to the autonomous system and the testing tool.

Large amount of data
The results from the thematic analysis in Cycle 1, as described in Section 4.1.2, confirm the notion from the problem statement of this thesis, that a large amount of data is a major challenge when testing automated autonomous driving functionalities in the automotive industry. The interviews from Cycle 1 also revealed that the collaborating V&V team conducted a feasibility study of how many scenarios their toolchain needed to be able to handle to ensure the safety of their function. The results of that study were hundreds of millions of scenarios.

Logically, more data takes more time to process, and the aforementioned safety requirements for autonomous functions in the automotive industry do not assist on that front. All functionalities deployed on the market must undergo thorough testing and fulfill certain regulations. To guarantee this, it has been seen that in some cases software needs to be tested on hundreds of millions of scenarios consisting of complex data. However, testing such a large volume of scenarios is time-consuming and requires advanced testing methodologies and tools to manage and analyse the data effectively. Companies must therefore spend resources, including time and money, to ensure that their systems can operate safely and reliably under different scenarios and conditions.

Trade-offs
The last challenge of testing autonomous drive functionalities in automated processes in the automotive industry in this thesis concerns two trade-offs; low execution time versus readability, and low cost versus low execution time when running the toolchain in the cloud versus running it locally on the organization's machines.

Regarding the first trade-off, on the one hand, reducing the execution time of the testing process is crucial for ensuring that the autonomous drive functionalities are tested efficiently. This could lead to faster deployment of new software updates, which would ultimately result in safer autonomous vehicles on public roads. This could be done by doing various things, for instance, employing a different and more efficient programming language, or analysing the simulations in real-time, instead of doing it sequentially. However, this can come at the expense of readability, where it may be challenging to understand the logic and flow of the testing process which might affect future scaling of the function. In the end, no matter how fast the toolchain is or what other benefits it has, it will not be able to be integrated into the organization if the code is not understandable by other employees, as hinted to by participants in the study, see Section 4.1.2

The second trade-off involves what environment to test the toolchain in, either cloud or locally. Running the toolchain in the cloud can provide several advantages such as increased computing power, flexibility, and scalability, see Section 2.2.2. This could result in faster execution times for executing all scenarios in the toolchain. Though,

as has been seen with the team at Volvo which pays for the number of nodes per time unit, the cost will be significantly higher with this option. This could potentially become a bigger issue if the testing process is run frequently or for long periods. On the other hand, running a toolchain locally on the organization's machines may cost less money, as there are no additional costs associated with paying for cloud services. However, this can result in longer execution times, as the organization's machines may have limited computing power or may not be optimized for the specific testing process. Local machines may also require maintenance and upgrades, which can add to the cost of running the toolchain locally. It is not possible to run hundreds of millions of scenarios locally, but when it comes to smaller batches of permutations, long execution times will result in long feedback times for employees.

When it comes to execution time vs readability, in the short term, reducing execution time may be crucial to meet testing deadlines and ensure the successfully timed deployment of updates. However, in the long term, sacrificing readability and things such as well-documented functionalities and code may lead to future issues that could impact the scalability and sustainability of the testing process. The trade-off between low cost and high execution time is something that will depend on the specific needs and resources of the organization, and as mentioned earlier, long and short-term goals. Therefore, it is essential to recognize that trade-offs are not one-size-fits-all solutions and require a thorough analysis of the company's current situation and long-term objectives. Teams also need to think about factors such as the frequency and duration of the testing toolchain, the complexity of the testing process, the computing power required, the budget available for the testing process, and how to communicate between team members and the larger organization. All of this, however, is out of scope for this project and need future research to establish best practice.

5.2 Research Validity

The external validity of the research could be threatened because there are numerous approaches to setting up a toolchain that is designed for testing autonomous functionalities in an automotive setting. This is important because components could be of numerous kinds and also vary internally. Due to the study's unique research context, generalizability therefore might become an issue for practitioners and academics seeking this research for inspiration. In addition, the research context is an industrial setting in collaboration with a company, making it challenging to provide specific information about the technical aspects of the toolchain due to confidentiality.

More specifically, another external validity is the potential threat to construct validity that emerges due to the thesis only focusing on a single method to shorten the execution time of the toolchain, which is real-time analysis. This could lead to limited or biased conclusions. For example, there might be other analysis techniques (besides a monitor), optimization algorithms, or solutions that could be utilized to improve the execution time. Despite this concern, the results still show that the

method of real-time analysis heavily affects the execution time positively. Though, when conducting future research, the goal should be to explore and evaluate alternative methods to make sure that the most effective and efficient methods are identified and implemented since that will make the research more generalizable.

In contrast to external validities, multiple internal validities needs to be taken into consideration when interpreting and making a conclusion based on the results. Firstly, the internal validity of the data collected through interviews with participants could be threatened by the subjective experiences and biases to influence their perception of the current automated process. This is particularly relevant in light of the limited perspective offered by only one expert in the collaborating company's team who has a comprehensive understanding of the toolchain, as other team members are absent due to parental leave and vacation.

Secondly, given the limitations posed by the absence of multiple experts, it was not feasible to obtain a diverse range of perspectives on the problem. To mitigate this issue, the expert was still asked to identify a participant who could provide additional information to supplement their insights. Additionally, this led to the action and reference groups that were involved in the Action Research methodology being comprised of the same individuals, as there was only one expert proficient in the toolchain. These factors could impact the validity of the findings and must be taken into consideration in the analysis and interpretation of the results.

Another aspect to consider is that one of the components of the toolchain that is being studied is developed and maintained by another in-house team at the collaboration company. This means that the action team's knowledge and understanding of this component is limited. To mitigate this concern, the action team sought support from this team and received guidance from them through interviews and informal conversations. The action team, however, can not rely on this team's assistance because they are under no responsibility to engage with the thesis work.

Lastly, due to the thesis only proposing a proof-of-concept and time constraints, fewer data points were utilized a few times throughout the study. This potentially poses a threat to the validity of the research. For example, the interviews in Cycle 1 only consisted of three participants, and each participant was asked different questions based on the role they had at the company. This leads to the research being based upon individuals' opinions, which could vary in a broader sense, as with other AD function teams in the automotive industry. Ideally, the interviews would involve more participants, but since there were no other relevant employees to interview due to parental leave and sickness, the decision was made to prioritize quality rather than quantity. When implementing the entire monitor concept in the future, a more established base of participants should be the focus. Furthermore, only one type of monitor has been implemented which means that one can only speculate on how implementing more monitors in the future would affect the execution time. Moving past the proof-of-concept and implementing multiple monitors would help us be more confident that the fully implemented monitor concept would decrease

the execution time of the toolchain. However, we have been clear throughout the study that we are only implementing a proof of concept and that we can't guarantee that the monitor concept as a whole would be a success.

6
Conclusion

In this thesis, a proof of concept of a ”stay-in-lane monitor” was implemented to analyse an automated testing toolchain for an AD functions V&V team in the automotive industry in real-time during the simulation, instead of afterward in a sequential fashion. This resulted in a significantly decreased execution time of the toolchain. Even though the results were highly positive, the fact still stands that the thesis only presented a proof of concept of one single monitor. However, the results suggest that implementing more monitors in the future would not significantly increase the execution time. Still, the concept will need more investigation and assessment in the future to ensure that realizing the concept fully is worth it. Whilst assessing and optimizing the automated testing toolchain process, it was found that the characteristics of an optimal testing toolchain were low execution time, low running cost, and high readability. The challenges of testing these functionalities were found to be strict regulations and safety requirements, a large amount of data, scenario complexities, and trade-offs between low execution time versus readability, and low cost versus low execution time.

6.1 Future Work

Some aspects of the study could not be investigated or executed due to the short period. First, while the thesis just investigated a proof of concept, which was found to be successful, it would be interesting to build upon it and implement monitors that cover and evaluate more KPIs. Future work could involve expanding the monitoring capabilities of the toolchain to include a broader range of KPIs that are relevant to the safety and performance of autonomous vehicles. For instance, a speed monitor that checks whether the vehicle keeps its designated speed could be another interesting KPI to look at. Other ones could include metrics related to acceleration, braking, proximity to other vehicles or objects, or if the vehicle has crashed or not. At last, to get a completely accurate result, all KPIs need to be evaluated within a monitor so that the analysis module can be removed completely.

Furthermore, another interesting aspect to look at would be how the result gets affected by a change of reference point. As previously mentioned, the proof of concept uses the middle point of the vehicle as a reference point to check whether the vehicle has left the lane. Meaning that the vehicle only left lane if more than half of the car has passed over the lane marking. This reference point makes sense when looking at certain KPIs such as if the vehicle has changed lanes completely. How-

ever, with the ”stay-in-lane monitor”, which checks whether the vehicle has stayed within its original lane, one can argue that the bounding box of the car would be a more suitable reference point. This was not implemented due to time constraints, and also the action team did not believe that the choice of reference point would affect the results significantly.

On the other hand, if the AD functions V&V teams decided on using monitors to test their AD functionalities in the future the reference points should be the bounding box of the vehicle to get the most accurate KPI result. Therefore, for future work, the ”stay-in-lane monitor” should be refactored to use the bounding box as a reference point rather than the center point.

Bibliography

[1] Rasheed Hussain and Sherali Zeadally. "Autonomous Cars: Research Results, Issues, and Future Challenges". In: *IEEE Communications Surveys Tutorials* 21.2 (2019), pp. 1275–1313. DOI: 10.1109/COMST.2018.2869360.

[2] Staron Miroslaw. *Automotive Software Architectures : An Introduction.* Springer, 2021. ISBN: 9783030659387. URL: https://search.ebscohost.com/login.aspx?direct=true&db=edsebk&AN=2802601&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[3] *SAE J3016 - Level of Driving Automation for On-Road Motor Vehicles.* SAE International, 2021. URL: https://www.sae.org/standards/content/j3016_202104/.

[4] Alessia Knauss et al. "Paving the roadway for safety of automated vehicles: An empirical study on testing challenges". In: *2017 IEEE Intelligent Vehicles Symposium (IV).* 2017, pp. 1873–1880. DOI: 10.1109/IVS.2017.7995978.

[5] Nidhi Kalra and Susan M. Paddock. "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" In: *Transportation Research Part A: Policy and Practice* 94 (2016), pp. 182–193. ISSN: 0965-8564. DOI: https://doi.org/10.1016/j.tra.2016.09.010. URL: https://www.sciencedirect.com/science/article/pii/S0965856416302129.

[6] *UN Regulation No 157 – Uniform provisions concerning the approval of vehicles with regards to Automated Lane Keeping Systems.* https://unece.org/transport/vehicle-regulations-wp29/standards/addenda-1958-agreement-regulations-141-160. Mar. 2023.

[7] Florian Rohde. *Continuous Integration as Mandatory Puzzle Piece for the Success of Autonomous Vehicles.* Tech. rep. SAE Technical Paper, 2020.

[8] Mike Roberts. "Enterprise Continuous Integration Using Binary Dependencies". In: *Extreme Programming and Agile Processes in Software Engineering.* Ed. by Jutta Eckstein and Hubert Baumeister. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 194–201. ISBN: 978-3-540-24853-8.

[9] Daniel Ståhl, Torvald Mårtensson, and Jan Bosch. "The continuity of continuous integration: Correlations and consequences". In: *Journal of Systems and Software* 127 (2017), pp. 150–167. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2017.02.003. URL: https://www.sciencedirect.com/science/article/pii/S0164121217300328.

[10] Darsh Parekh et al. "A Review on Autonomous Vehicles: Progress, Methods and Challenges". English. In: *Electronics (Switzerland)* 11.14 (July 2022). Funding Information: This work is supported by the Korea Agency for In-

frastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant 21AMDP-C161756-01). Publisher Copyright: © 2022 by the authors. ISSN: 2079-9292. DOI: 10.3390/electronics11142162.

[11] *UN regulation for Automated Lane Keeping Systems (ALKS) extended to trucks.* United Nations Economic Commission for Europe (UNECE), 2021. URL: https://unece.org/sustainable-development/press/un-regulation-automated-lane-keeping-systems-alks-extended-trucks.

[12] Eric Thorn, Shawn C. Kimmel, and Michelle Chaka. "A Framework for Automated Driving System Testable Cases and Scenarios". In: 2018.

[13] Francisca Rosique et al. "A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research". In: *Sensors* 19 (Feb. 2019), p. 648. DOI: 10.3390/s19030648.

[14] Changwoo Park, Seunghwan Chung, and Hyeongcheol Lee. "Vehicle-in-the-Loop in Global Coordinates for Advanced Driver Assistance System". In: *Applied Sciences* 10.8 (Apr. 2020), p. 2645. ISSN: 2076-3417. DOI: 10.3390/app10082645. URL: http://dx.doi.org/10.3390/app10082645.

[15] Oliver Meister et al. "Software-in-the-loop simulation for small autonomous VTOL UAV with teaming capability". In: *The Navigation Conference&Exhibition, October, 28-30.2008, London, UK*. The Navigation Conference & Exhibition. 2008 (London, Vereinigtes Königreich, Oct. 28–30, 2008). Royal Institute of Navigation, 2008, 7 Seiten.

[16] Mohamed Fasil Syed Ahamed, Girma Tewolde, and Jaerock Kwon. "Software-in-the-Loop Modeling and Simulation Framework for Autonomous Vehicles." In: *2018 IEEE International Conference on Electro/Information Technology (EIT), Electro/Information Technology (EIT), 2018 IEEE International Conference on* (2018), pp. 0305–0310. ISSN: 978-1-5386-5398-2. URL: https://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.8500101&site=eds-live&scope=site&authtype=guest&custid=s3911979&groupid=main&profile=eds.

[17] Stephanie Demers, Praveen Gopalakrishnan, and Latha Kant. "A Generic Solution to Software-in-the-Loop". In: *MILCOM 2007 - IEEE Military Communications Conference*. 2007, pp. 1–6. DOI: 10.1109/MILCOM.2007.4455268.

[18] Fahad Al-Dhief, Naseer Sabri, and Musatafa Albadr. "Performance Comparison between TCP and UDP Protocols in Different Simulation Scenarios". In: *International Journal of Engineering Technology* (Dec. 2018).

[19] Wenhao Ding et al. "A Survey on Safety-critical Scenario Generation from Methodological Perspective". In: (Feb. 2022).

[20] Tania Lorido-Botran, Jose Miguel-Alonso, and Jose A. Lozano. "A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments". In: *Journal of Grid Computing* 12 (2014), pp. 559–592.

[21] Jerry Gao, Xiaoying Bai, and Wei-Tek Tsai. "Cloud testing-issues, challenges, needs and practice". In: *Software Engineering: An International Journal* 1.1 (2011), pp. 9–23.

[22] Amandeep Kaur, Navjeet Singh, and Dr Gurdev Singh. "An overview of cloud testing as a service". In: *International Journal of Computers & Technology* 2.2 (2012), pp. 18–23.

[23] Abdallah Qusef et al. "Challenges and opportunities in cloud testing". In: *Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems*. 2019, pp. 1–7.

[24] *Safety - Highlights | Volvo Cars*. Last accessed 28 Mars 2023. URL: https://www.volvocars.com/intl/v/safety/highlights.

[25] Stefan Riedmaier et al. "Survey on Scenario-Based Safety Assessment of Automated Vehicles". In: *IEEE Access* 8 (2020), pp. 87456–87477. DOI: 10.1109/ACCESS.2020.2993730.

[26] Jinkang Cai et al. "A Survey on Data-Driven Scenario Generation for Automated Vehicle Testing". In: *Machines* 10 (Nov. 2022), p. 1101. DOI: 10.3390/machines10111101.

[27] Association for Standardization of Automation and Measuring Systems. *ASAM OSI® (Open Simulation Interface)*. Last accessed 12 Mars 2023. URL: https://www.asam.net/index.php?eID=dumpFile&t=f&f=5040&token=b03e297c786c196257f3609f1c5575eda908059f.

[28] Association for Standardization of Automation and Measuring Systems. *ASAM Open Simulation Interface (OSI)*. Last accessed 14 Mars 2023. URL: https://report.asam.net/asam-open-simulation-interface-osi.

[29] *Esmini User Guide*. Last accessed 12 Mars 2023. URL: https://esmini.github.io/#_introduction.

[30] Cem Sürücü et al. "Establishing Key Performance Indicators for Measuring Software-Development Processes at a Large Organization". In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ESEC/FSE 2020. Virtual Event, USA: Association for Computing Machinery, 2020, pp. 1331–1341. ISBN: 9781450370431. DOI: 10.1145/3368089.3417057. URL: https://doi.org/10.1145/3368089.3417057.

[31] Zakaria Alomari et al. "Comparative Studies of Six Programming Languages". In: *CoRR* abs/1504.00693 (2015).

[32] Farzeen Zehra et al. "Comparative Analysis of C++ and Python in Terms of Memory and Time". In: 2020.

[33] Kumar C Ramesh. "A Comparison between Python and C++". In: *International Journal of Emerging Technologies and Innovative Research* 6 (1 Jan. 2019), pp. 40–46. ISSN: 2349-5162.

[34] Muhammad Ateeq et al. "C++ or Python? Which One to Begin with: A Learner's Perspective". In: *2014 International Conference on Teaching and Learning in Computing and Engineering*. 2014, pp. 64–69. DOI: 10.1109/LaTiCE.2014.20.

[35] MO Balogun. "Comparative Analysis of Complexity of C++ and Python Programming Languages". In: *Asian J. Soc. Sci. Manag. Technol* 4 (2022), pp. 1–12.

[36] Sabah A. Abdulkareem and Ali J. Abboud. "Evaluating Python, C++, JavaScript and Java Programming Languages Based on Software Complexity Calculator

(Halstead Metrics)". In: *IOP Conference Series: Materials Science and Engineering* 1076.1 (Feb. 2021), p. 012046. DOI: 10.1088/1757-899X/1076/1/012046. URL: https://dx.doi.org/10.1088/1757-899X/1076/1/012046.

[37] Miroslaw Staron. *Action Research in Software Engineering: Theory and Applications*. Jan. 2020. ISBN: 978-3-030-32609-8. DOI: 10.1007/978-3-030-32610-4.

[38] *Ubuntu*. Last accessed 14 Mars 2023. URL: https://ubuntu.com/.

[39] Gaganpreet Sharma. "Pros and cons of different sampling techniques". In: *International Journal of Applied Research* 3.7 (2017), pp. 749–752. ISSN: 2394-5869. URL: https://www.allresearchjournal.com/archives/2017/vol3issue7/PartK/3-7-69-542.pdf.

[40] Helen Sharp, Yvonne Rogers, and Jennifer Preece. *Interaction Design - Beyond human-computer interaction*. 5th Edition. John Wiley Sons, Inc, 2019. ISBN: 9781119547259.

[41] Python. *Python time Module*. Last accessed 1 Mars 2023. URL: https://docs.python.org/3/library/time.html.

[42] Python Software Foundation. *The python profilers*. URL: https://docs.python.org/3/library/profile.html.

A

Interview Material

Appendix A consist of the material used for the interviews of Cycle 1. It includes the interview questions from the interviews conducted in Cycle 1 (A.1 & A.2), the interview questions from the evaluation meetings conducted in Cycle 1 (A.3), Cycle 2 (A.4) & Cycle 3 (A.5), and lastly the consent form (A.6) used for the interviews in Cycle 1.

A.1 Interview 1: Questions

Below is the question prepared during the action planning in Cycle 1 and later asked during one of the interviews in action taking Cycle 1.

1. What is your stated role at your company, and what are your everyday tasks?
2. Which people are you primarily working with?
3. Can you briefly explain the tool-chain from the beginning to the end?
4. Which metrics do you use today connected to the tool chain?
   (a) What are the different metrics used for?
   (b) Are there any metrics you feel you are missing?
5. Do you see any limitations or potential issues with how the automated process/tool-chain looks today?
   (a) If you would solve these limitations/issues, where would you start?
   (b) If you could improve the automated process/tool-chain in any capacity, what would you improve? and why?
6. Is there anything else that you would like to add to the topic?
7. Is there anyone at your company that could give us any additional information about the topics we've talked about today? In that case, who?

A.2 Interview 2: Questions

Below is the question prepared during the action planning in Cycle 1 and later asked during two of the interviews in action taking Cycle 1.

1. What is your stated role at your company, and what are your everyday tasks?
2. How and when do you work together with the AD function team?
3. How are you deploying nodes in the cloud and how are you running the components in parallel?
4. Why is the set-up part of the nodes so time-consuming?
   (a) What can be done about it? Can anything be optimized?
5. We've heard that you have tried to implement a "monitor" in the simulation step, can you tell us about that?
   (a) What worked well? What didn't?
   (b) Can it be scaled up nicely? In that case, how?
   (c) Do you think that it would be feasible to run the analysis step online in real-time during the simulations?
6. Is it anything else that you would like to add to the topic?
7. Is there anyone at your company that could give us any additional information about the topics we've talked about today? In that case, who?

A.3 Evaluation Meeting 1: Questions

Below are the question asked during the evaluation meeting in Cycle 1.

1. How well do you think that we have perceived the actual problem at hand?
2. Have we missed any important variables? In that case, which variable?
3. Have we perceived anything incorrectly?
4. How representative are the chosen scenarios?
   (a) Is there any important scenario missing?
5. Is 1 & 10 scenarios fitting to get a good understanding of your problem?
6. How well do our results agree with your perception of the tool-chain regarding ...
   (a) ... Wall time?
   (b) ... Profiling simulation?
   (c) ... Profiling analysis?
7. Is any result new for you? In that case, do you think that the result is correct? Why? Why not?

A.4 Evaluation Meeting 2: Questions

Below are the question asked during the evaluation meeting in Cycle 2.

1. Is any result new for you? In that case, do you think that the result is correct? Why? Why not?
2. In your opinion, why is it such a big difference between model in and out of loop regarding the simulation?
3. In your opinion, why is it such a big difference between model in and out of loop regarding the Analysis?
4. After seeing these results, do you still believe that implementing a monitor is the most appropriate action to take?
5. Do you have any more comments on the results?

A.5 Evaluation Meeting 3: Questions

Below are the question asked during the evaluation meeting in Cycle 3.

1. How does the choice of reference point affect the results?
2. How does the choice of logging method affect the results?
3. Is any result new for you? In that case, do you think that the result is correct? Why? Why not?
4. Is the monitor concept worth implementing?
5. How will the execution times be affected if...
   (a) ... more monitors were added?
   (b) ... the ''stay-in-lane-monitor" evaluated more than one KPI?
   (c) ... if more than one KPI was evaluated in the analysis module?
6. How many monitors do you estimate will be needed to be able to remove the analysis?
7. With that amount of monitors, in your opinion, is the monitor concept worth it?

A.6 Consent form

Below is the consent form used for the interviews in Cycle 1 of the study.

Consent and information about processing of personal data in student thesis

I agree to my personal data in the form of:

Name, job title, work routine & audio recording

may be treated by Chalmers University of Technology for the study:

This study is a master thesis written and conducted by Johanna Wiberg and Oscar Forsberg, students at Chalmers University of Technology. The purpose of the study is to assess and optimize an automated process at an automotive company. The data will be used to get a better understanding of the automated process current state and potential limitations.

Information

Your personal data will be handled as follows:

- *The data will be analyzed and will be published as a part of an aggregated analysis in a master thesis.*
- *The data can be discussed with a supervisor outside the company.*
- *The data will be stored within the company's systems.*

Your consent is valid until further notice. You have the right to withdraw your consent at any time. You do this through contacting *Johanna Wiberg* at jwiberg@student.chalmers.se.

If you withdraw your consent, we will cease processing personal data we have collected with the support of your consent. Some information may be saved due to Chalmers obligations under Swedish archive legislation.
Chalmers University of Technology, org. No. 556479-5598 is
personal data controller. You can find Chalmers privacy policy at www.chalmers.se.

As a participant you have the right to receive information about how your personal data is processed. You have the right to have incorrect information corrected, redundant data deleted, request that processing shall be restricted and data transferred to another actor. You also have the right to submit a complaint to the Swedish Authority for Privacy Protection (Integritetsskyddsmyndigheten). Do you have any questions about Chalmers processing of personal data contact Chalmers' data protection officer at dataskydd@chalmers.se.

I agree that Chalmers University of Technology processes personal data about me in accordance with the above.

Place:	Signature
Date:	Name clarification

The form is drawn up in duplicate.