Thesis for The Degree of Doctor of Philosophy

Bridging the Experimental Gap: Applying Continuous Experimentation to the Field of Cyber-Physical Systems, in the Example of the Automotive Domain

Federico Giaimo

Division of Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden, 2020

Bridging the Experimental Gap:
Applying Continuous Experimentation to the Field of Cyber-Physical Systems, in the Example of the Automotive Domain

Federico Giaimo

ISBN 978-91-7905-346-8
Doktorsavhandlingar vid Chalmers tekniska högskola, Ny serie nr 4813.
ISSN 0346-718X

Technical Report No 188D
Department of Computer Science & Engineering
Division of Software Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2020.

It doesn't make any difference how beautiful your guess is, it doesn't make any difference how smart you are, who made the guess, or what his name is. If it disagrees with experiment, it's wrong.

Richard P. Feynman

Abstract

In the software world frequent updates and fast delivery of new features are needed by companies to bring value to customers and not lag behind competition. When in cyber-physical systems the software functionality dominates in importance the hardware capabilities, the same speed in creating new value is needed by the product owners to differentiate their products and attract customers. The automotive field is an example of a domain that will face this challenge as the industry races to achieve self-driving vehicles, which will necessarily be software-intensive highly complex cyber-physical systems.

A software engineering practice capable of accelerating and guiding the software production process using real-world data is Continuous Experimentation. This practice proved to be valuable in software-intensive web-based systems, allowing data-driven software evolution. It involves the use of *experiments*, which are instrumented versions of the software to be tested, deployed to the actual systems and executed in a limited way alongside the official software version. Valuable data on the future behavior of the prospective feature is collected in this way as it was fed the same real-world data it would encounter once approved and deployed. Additionally, in those cases where an experimental software version can be run as a replacement for the official version, relevant data regarding the system-user interaction can be gathered.

In this thesis, the field of cyber-physical systems and the automotive practitioners' perspective on Continuous Experimentation are sampled employing a literature review and a series of case studies. A set of necessary architectural characteristics are defined and possible methods to overcome the issue of resource constraints in cyber-physical systems are proposed in two exploratory studies. Finally, a design study shows and analyses a prototype of a Continuous Experimentation cycle that was designed and executed in a project partnered by Revere, the Chalmers University of Technology's laboratory for vehicle research.

Keywords Continuous Experimentation, Cyber-Physical Systems, Software Engineering

Acknowledgment

I want to express all my gratitude to my supervisors Christian Berger and Ivica Crnkovic, you were a great source of inspiration and your leadership, teachings and support along these years were an irreplaceable help for me on the way to this achievement. Thank you for your advice and your trust that allowed me to grow and learn so much. I also wish to thank my examiner Michel Chaudron for his feedback and counseling that helped me develop my skills as a researcher.

A big praise the great people at the Software Engineering Division and Revere laboratory: Jan Schröder, Hang Yin, Yue Kang, Sergio García Gonzalo, Piergiuseppe Mallozzi, Magnus Ågren, Katja Tuma, Rebekka Wohlrab, Rodi Jolak, Truong Ho-Quang, Fredrik Von Corswant, Ola Benderius, Björnborg Nguyen, Arpit Karsolia, Ann Tornberg, and all the other colleagues and professors for the nice conversations we had about research, music, or anything else. You make the department the fantastic workplace it is.

A special mention to Hugo Andrade, we started this endeavor together and supported each other in our time of need. You proved to be a great colleague and an even better friend. Let's grab our motorcycles and ride off into the sunset again soon!

Many thanks to my old friends in Italy (I miss you guys!) and the new ones in Sweden who encouraged, cheered and at times tolerated me for so long. I hope to make you all meet each other at least once!

Last but not least, I will thank my family. Thank you for giving me the opportunities I had to choose my path in life and for supporting me whenever I was in need.

My research work has been supported by the 'COPPLAR - CampusShuttle cooperative perception and planning platform' project, funded by Vinnova FFI (2015-04849) and by the 'Highly Automated Freight Transports - AutoFreight' project, also funded by Vinnova FFI (2016-05413).

List of Publications

Appended publications

This thesis is based on the following publications:

[A] Giaimo, F., Andrade, H. and Berger, C. (2019, August). "The Automotive Take on Continuous Experimentation: A Multiple Case Study". In *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 126-130). IEEE.

[B] Giaimo, F., Andrade, H. and Berger, C. (submitted 2019, publishing ongoing at the time of writing) "Continuous Experimentation and the Cyber-Physical Systems challenge. An overview of the literature and the industrial perspective". In *Journal of Systems and Software.* Elsevier.

[C] Giaimo, F. and Berger, C. (2017, April). "Design criteria to architect continuous experimentation for self-driving vehicles". In *2017 IEEE International Conference on Software Architecture (ICSA)* (pp. 203-210). IEEE.

[D] Giaimo, F., Berger, C. and Kirchner, C. (2017, September). "Considerations about continuous experimentation for resource-constrained platforms in self-driving vehicles". In *11th European Conference on Software Architecture (ECSA 2017)* (pp. 84-91). Springer, Cham.

[E] Giaimo, F. and Berger, C. (2020, publishing ongoing at the time of writing) "Continuous Experimentation for Automotive Software on the Example of a Heavy Commercial Vehicle in Daily Operation". In *14th European Conference on Software Architecture (ECSA 2020).* Springer.

Other publications

The following publications were published during my PhD studies. However, they are not appended to this thesis, due to contents overlapping that of appended publications or contents not related to the thesis.

[a] Giaimo, F., Andrade, H., Berger, C., and Crnkovic, I. (2015, September). "Improving Bandwidth Efficiency with Self-Adaptation for Data Marshalling on the Example of a Self-Driving Miniature Car"
In *Proceedings of the 2015 European Conference on Software Architecture Workshops* (pp. 1-6).

[b] Andrade, H., Giaimo, F., Berger, C., and Crnkovic, I. (2015, October). "Systematic evaluation of three data marshalling approaches for distributed software systems"
In *Proceedings of the Workshop on Domain-Specific Modeling* (pp. 71-76).

[c] Yin, H., Giaimo, F., Andrade, H., Berger, C., and Crnkovic, I. (2016). "Adaptive Message Restructuring Using Model-Driven Engineering"
In *Information Technology: New Generations* (pp. 773-783). Springer, Cham.

[d] Giaimo, F., Yin, H., Berger, C., and Crnkovic, I. (2016, May). "Continuous Experimentation on Cyber-Physical Systems: Challenges and Opportunities"
In *Proceedings of the Scientific Workshop Proceedings of XP2016* (pp. 1-2).

Research Contribution

My personal contributions to the included articles are as follows:

Paper A	I took part to the execution of the case studies and the data collection. Additionally, I participated in the data analysis and led the writing process as main author.
Paper B	I took part to the execution of the case studies, their data collection, and data analysis phases; additionally, I performed the systematic literature review and led the writing process as main author.
Paper C	I led the research and was the main author of this work: my contributions lay in the formulation of the set of architectural properties that are the main findings of the work, plus all stages of the writing process.
Paper D	This paper extended the findings obtained in a master thesis that I co-supervised. My contributions relate to the additional work that led to the definition of the architectural features of interest and all stages of the writing process.
Paper E	For this article I took part to the decision process for the system architecture of the prototype and the software architecture of the infrastructure enabling the experiment system; additionally, I developed the software experiments used to validate the experiment architecture and led the writing process as main author.

Contents

Chapter 1

Introduction

1.1 Introduction and Background

Aiming at simplifying everyone's daily tasks and lives, software and software-enhanced gadgets, appliances, tools, and machinery are nowadays ubiquitous. The majority of these items fall in the category of cyber-physical systems. Cyber-physical systems are defined as "integrations of computation and physical processes" [1], meaning that these systems are immersed in the physical world and interact with it as origin and/or result of their computation.

As the focus of customers and companies shifts from the actual hardware of the products to the software functionality they offer, it can happen over time that the hardware may eventually lose importance in the choice between different companies' products. In this case the software takes its place, meaning that the software capabilities and usability become the main decision factors for customers. In turn, this entails that the release of new software updates and functionality becomes the driver of the value-creation process, and accelerating this development becomes of strategic importance for the product owner. This need can also be found in other software fields, such as web-based systems, meaning that known successful strategies from these fields could potentially be adopted on cyber-physical systems to solve the same issue. However, the hardware side of the product is still present and influences the way any new techniques could be implemented for this type of systems, requiring some form of adaptation to the field of application first.

In this thesis the Continuous Experimentation practice, originating from the web-based software field, is highlighted as possible tool to increase the rate and the quality of future software production and delivery for complex cyber-physical systems. This technique aims at guiding the product owners and developers through the software evolution process by collecting real-world data to be used to make informed decisions about new or different features and functionality. Unfortunately, this practice cannot be simply replicated on cyber-physical systems "as is", because of the differences between web-based systems and cyber-physical systems. The automotive industry has been chosen as a representation for cyber-physical systems because of its current efforts to enable software-intensive capabilities on vehicles, which can be considered as cyber-physical systems in their own right. Additionally, some automotive companies have mechanisms in place to provide Over-the-Air (OTA) updates to their systems, implying the presence of a remote link between the company and the systems on the field which constitutes one of the prerequisites for Continuous Experimentation.

The thesis will expand firstly on what the research community and the practitioners have done for or expect from the Continuous Experimentation practice on cyber-physical systems. This will be done thanks to two studies that gathered and described both the research landscape and the automotive practitioners' perspectives on the matter. From these studies it emerges that the Continuous Experimentation practice is desirable for the industrial practitioners, although a number of challenges prevent its immediate adoption. Furthermore, the state-of-the-art shows that the engagement on Continuous Experimentation is still in its infancy in the field of cyber-physical systems. Then, two additional studies will investigate and illustrate what characteristics a cyber-physical system should exhibit in order to execute this practice, and what possible solutions could be adopted to tackle the issue posed by hardware limited in computational resources. To do so, a set of relevant design criteria for the software and its development process are provided to enable the adoption of the practice. Additionally, three execution strategies, comparable to high-level scheduling strategies, are proposed to allow resource-constrained hardware to run experimental software while limiting its influence on the rest of the system. Finally, the last study included in the thesis will describe a prototype for Continuous Experimentation on an automotive cyber-physical system developed as a proof-of-concept to show its feasibility, and it will compare it against the characteristics that were identified in the previous studies. The proposed system architecture abides to aforementioned design criteria and the software infrastructure is shown to solve a software evolution question concerning an algorithmic choice by using real-world data.

The document is structured as follows: Section 1.1.1 provides a background on the Continuous Practices and Continuous Experimentation; Section 1.1.2 details the scope of the thesis; Section 1.1.3 describes relevant works that can be found in literature and how this thesis is positioned in the research landscape; Section 1.2 expands on the Research Goal and Methods; Section 1.3 describes the projects in which the research work was performed; Section 1.4 lists and summarizes the publications included in this thesis; Section 1.5 discusses the results and their significance; and finally Section 1.6 concludes the thesis and offers possible directions for future research work.

1.1.1 Continuous Practices

The connection between software as value-maker and the need for quicker releases is both intuitive and very apparent in the software industry, especially for what concerns web-based software. In that field, a number of development techniques have been introduced to accelerate the process as much as possible by eliminating technical downtimes and improving the way users interact with the systems themselves [2]. These techniques are called Continuous Integration (CI), Continuous Deployment (CD), and Continuous Experimentation (CE). These techniques introduce automated mechanisms that allow the immediate integration of new software into the entire code base as soon as possible

(Continuous Integration), the possibility of immediate deploying of newly integrated software code into the actual systems (Continuous Deployment), and the possibility of deploying and running alongside the *official* software a number of *experiments*, i.e., different versions of the official software, in order to evaluate their respective performances (Continuous Experimentation). This experiment-based approach to software evolution is shown more in detail in Figure 1.1 and can be broken down in the following Phases:

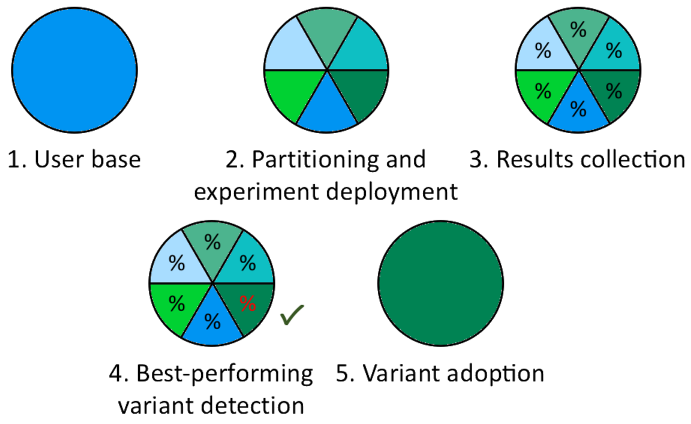

%
1. User base
2. Partitioning and experiment deployment
3. Results collection
4. Best-performing variant detection
5. Variant adoption

Figure 1.1: Phases of the Continuous Experimentation process.

Phase 1 : The user base available to experimentation is defined. The set of users affected by the overall experimentation phase is chosen in such a way that the results will be coherent and meaningful. Depending on the type of software and experiments, the user base could be the users of a certain geographic area, or those operating the system during a certain time of the day, etc.;

Phase 2 : The user base is partitioned into non-overlapping clusters of users per experiment. In each cluster a different experiment will be deployed, with the exception of the control cluster, which will not receive any experiment. More than one experiment can be run in a cluster only if they do not interfere with each other [3, 4];

Phase 3 : The experiments run either replacing or alongside the official stable software and produce output in form of measurements, logs, etc. In fact, they do not necessarily need to change the system behavior, especially in those cases in which the functionality being experimented has safety-critical implications. These results are collected and finally transmitted back to the experimenter to be analyzed;

Phase 4 : The collected results are analyzed and the best-performing experiment according to the set objectives is identified. The successful experiment is then developed and finalized in preparation to be deployed to the entire population;

Phase 5 : The successful experiment is finally integrated as a new software feature or version, which can be deployed to the entire user base in the form of an update. The overall system will now perform its mission in a way that is more conform to the experiment's objectives.

The advantage is that Continuous Experimentation employs real-world data to confirm or reject hypotheses about the software suitability for a given task, as opposed to relying only on speculations or simulations to steer the development of new software. Continuous Experimentation has proven very effective on web-based software systems, with many companies adopting the experiment-driven approach to guide the majority if not all their software evolution and release processes [5].

However, applying this technique onto complex safety-critical cyber-physical systems such as vehicles without adapting appropriately to the new context would be an endeavour destined to fail due to the additional challenges that are specific to the automotive field. One first challenge, for example, is the resource availability problem given by the Continuous Experimentation practice itself, which introduces the need for additional computational power [6]. Moreover, there is additional complexity given by the fact that the target systems in the case of the automotive field are not only limited in computational performances but also highly mobile physical objects, as opposed to the virtual machines in the highly available server farms typical of the web-based software field. This poses issues to the automotive industry which, being based on an economy of scale, builds vehicles with hardware that is *just enough* powerful to fulfill its tasks in order to lower production costs.

1.1.2 Scope

The scope of this work is to bring forward the adoption of the Continuous Experimentation practice in the cyber-physical systems field. The aforementioned definition of cyber-physical systems as “integrations of computation with physical processes” [1] is quite broad and includes low-power and low-capabilities devices that are an important focus in some research and industrial areas, e.g., the Internet of Things. However, due to the computational and connectivity needs that a practice like Continuous Experimentation exhibits, the cyber-physical systems that are referred to in the context of this work are those systems that are built with or that could accommodate for adequate processing power and at least sporadic connectivity capabilities.

The automotive domain provides an interesting use case in this sense due to the trend that is currently taking place in this field. Many automotive companies are in fact increasing and advancing their software capabilities in order to provide as much automation as possible to their customers, with the goal of ultimately achieve self-driving vehicles. Vehicles, which nowadays can contain more than a hundred of computational units called Electronic Control Units (ECUs) [7], can be viewed as cyber-physical systems. The achievement of self-driving technology is of relevance since the difficult and intensive computation that will be needed to make it possible would complete the transformation of vehicles into data centers on wheels, and it is likely to shift once and for all the true value of the product from its hardware platform to its software capabilities. This in turn would trigger and exacerbate the need of automotive companies for a software release cycle as fast as possible, and for techniques capable of improving the quality of the software not only on the technical level but also on the customer desirability level. For these reasons, while the general interest is to enable Continuous Experimentation in cyber-physical systems, the scope of this thesis focuses more on the automotive systems. This choice does not intend to exclude all other possible fields or systems but before Continuous Experimentation could be applied in many of the current cyber-physical systems sub-fields there are still several field-related technological challenges yet to overcome, more than the ones that the automotive systems face at the present development stage.

1.1.3 Related Works

A number of studies explore the Continuous Experimentation practice in its native application field, i.e., web-based systems. More recently, and for this reason in fewer studies, it is investigated also in the context of cyber-physical systems.

Several articles related to Continuous Experimentation report the advancements and characteristics of the experimentation processes and platforms in web-based and pure software industrial settings. A recent example of these works is Gupta *et al.* [5] which describes the First Practical Online Controlled Experiments Summit. This summit saw several experts in experimentation from a number of software and online-based industries convene to discuss and report the experimentation processes they have in place, and provided the main challenges they face, as well as some common solutions employed. Among earlier works of the same track there were Tang *et al.* [4] that described the experimental setting at Google Inc. and the way they overlap experiments that test independent factors; and Kohavi *et al.* [8], that described Microsoft Bing's own solution to run "over 200 experiments concurrently"; and also Amatriain [9], that outlined Netflix's approach to experimentation. These articles reported their companies' approaches to experimentation, describing the systematic way in which the experiments were produced, deployed, and their results analyzed. They also show how experiments are run in parallel in order to increase the efficiency of the process, and the additional care that is invested in designing them and analyzing their data in order to ensure that different experiments do not interfere with each other to avoid leading the product owners to inaccurate conclusions. Fagerholm *et al.* [10] defined an organizational model for Continuous Experimentation in the context of web-based products, comprising the tasks and artifacts that different roles involved in planning and implementation of the experimentation process of a software product should manage. All these works focus on industrial practices in the context of experiments run on web-based systems having very high numbers of users and sometimes even concurrently. While they are very relevant for what concerns the practice itself and the challenge of experimenting on a high number of target systems, they provide limited insight for the challenges specific to cyber-physical systems.

Mapping studies on the Continuous Experimentation practice show that the majority of the works they encountered explore the statistical methods sub-topic and are often rooted in the web-based applications context, which is the originating field of this practice; while only a minority of studies are addressing the Continuous Experimentation practice in the cyber-physical systems field [11, 12]. This is a likely consequence of the novelty of Continuous Experimentation in this field and perhaps a result of the lack of the strong industrial backing that can be seen for the web-based experimentation practice. The work described in this thesis aims at investigating further this new context for experimentation and possibly act as a starting point for future efforts towards a systematic approach to experimentation on cyber-physical systems.

An early study linking the cyber-physical and automotive field to post-deployment data (although experiments are not explicitly mentioned) is the one by Olsson and Bosch [13]. In their work they interview representatives from three companies, one of which is an automotive manufacturer. They show that while post-deployment data collection mechanisms are in place, the collected data is only partially used. Moreover, they report that the purpose of this feedback is normally just troubleshooting and not supporting a product improvement process. Mattos *et al.* [14] performed a literature review and identified a set of challenges for Continuous Experimentation in cyber-physical systems which was used as a starting point for a case study where they tried to identify possible solutions with industrial representatives. This case study had a similar scope as the multiple case study included in this thesis, but key differences in the methodology of the case studies and the industrial context that was involved divide the two articles. As a result of these differences, the conclusions reached by the two works diverge and return different sets of challenges; nevertheless, both studies agree on a number of key issues that are still unsolved and complicate the adoption of Continuous Experimentation in this field. A more recent work by Mattos *et al.* [15] focuses instead on the experimentation process concerning business-to-business mission-critical systems, i.e., those systems that in the case of failures or service degradation can lead to property or reputation damage as well as preventing the system's main goal to be achieved. This work provides an interesting view on mission-critical vulnerabilities and the way experimentation could be adopted, perhaps highlighting interesting leads to overcome the future challenges related to safety-critical cyber-physical systems and experimentation; however the cyber-physical systems themselves are not the focus of the work, separating the scope of the article from the one of this thesis.

Continuous Experimentation is not the only practice for software development and evolution based on testing hypotheses. For example, A/B testing is a technique that was used on web-based systems and while it retained the concept of testing one or more different or additional software features against the base software, it did not necessarily have such a strong emphasis on the Continuous practices and the systematic approach to tests which is peculiar to Continuous Experimentation. In more recent times however, it seems that practitioners use the terms A/B testing, Continuous Experimentation, and Online Controlled Experiments interchangeably [16].

To conclude, it is apparent that much work has been undertaken by industrial and academic parties along the years for what concerns experimentation on web-based systems. Not only many architectures and approaches are reported and discussed but also the need for and use of rigorous statistical tools. Modification to the organizational structures were also suggested to enable and support this practice in the organizational contexts. This scenario changes dramatically when considering cyber-physical systems instead of web-based systems, since much less scientific works are available and very few attempts at realizing these goals are reported at all. This research topic can thus be considered still open and this thesis attempts to investigate it, aiming at assessing the Research Questions and producing valuable knowledge for both future researchers and practitioners.

1.2 Research Goal and Methods

Having in mind the advantages brought by Continuous Experimentation to web-based systems, the Research Goal driving the work was

> *to introduce the Continuous Experimentation practice to cyber-physical systems, on the example of the automotive domain.*

This can be clarified as the proposal of an artifact that could demonstrate the feasibility of the Continuous Experimentation practice in the context of cyber-physical systems. The research work to reach this goal comprised several steps: (a) it collected the practitioners' impressions about the possibility to utilise such tool for their work; (b) it reported the state-of-the-art and a final investigation on the desirability of Continuous Experimentation in the automotive setting; (c) it explored possible architectural requirements that would enable Continuous Experimentation on cyber-physical systems; (d) it highlighted a number of relevant technical challenges likely to emerge and possible ways to tackle them; and (e) it concluded proposing and testing a proof-of-concept for Continuous Experimentation, whose architectural features were then compared to the ones proposed at the beginning of this research journey, in order to validate them in light of the practical work achieved at the end of this path.

The research work can be thematically divided in three phases, each comprising a subset of the ten Research Questions (RQ) that were posed to reach the Research Goal: the *Context exploration*, the *Design proposition*, and the *Prototype analysis*. These three phases, the Research Questions and the respective research methods employed to tackle them are shown in Figure 1.2.

Context Exploration. This phase aims at investigating both the literature take on Continuous Experimentation on cyber-physical systems, and the industrial level of interest and involvement. In Paper A, RQ1 and RQ2 investigate into the practitioners' prospects and ideas. The most fitting method for these two questions was deemed to be empirical research, more specifically a multiple case study [17], which was run with four companies. Paper B, comprising RQ3 and RQ4, described the status of the literature concerning the Continuous Experimentation practice, complementing this view with a multiple case study. RQ3 focused on a systematic literature review performed following the guidelines by Kitchenham *et al.* (2015) [18]; RQ4 instead extended the multiple case study reported in Paper A with another empirical investigation run with two companies. Together, these two papers provide the reader a double-sided view on Continuous Experimentation on cyber-physical systems, by analysing both the state-of-the-art and the industrial perspective. The Research Questions were:

*RQ*1 : What would be the added values for practitioners in the context of self-driving vehicles if Continuous Experimentation is successfully in place?

*RQ*2 : What would be the additional challenges for practitioners in the context of self-driving vehicles if Continuous Experimentation would be in place?

*RQ*3 : In the context of cyber-physical systems, what is the state-of-the-art of Continuous Experimentation?

*RQ*4 : In the context of cyber-physical systems and more specifically the automotive industry, what feedback do the practitioners provide about the Continuous Experimentation practice?

Design Proposition. The aim of this phase was to define architectural properties and propose solutions to some of the technical hurdles that are still to be solved in this field. Once an overview of the context was provided by RQ1 to RQ4 in Papers A and B, the Research Questions that followed had a more preparatory and practical approach. RQ5 and RQ6, in Paper C, focused both on the design of Continuous Experimentation-capable complex cyber-physical systems, searching for those qualities that such systems should express to enable the practice. For this reason they were investigated adopting an exploratory approach [19], with the goal of defining a qualitative model or set of characteristics. Similarly, RQ7 and RQ8, in Paper D, focused on what could be the impact of limited resources on the application of Continuous Experimentation and possible ways to circumvent the identified issues. To achieve this, an exploratory study [19] was devised, resulting in a set of "execution strategies", i.e., methods to execute experimental software to minimize its impact on the system's resources. The Research Questions were:

*RQ*5 : What should the software architecture provide or abide to in order to enable a Continuous Experimentation process?

*RQ*6 : What should the software development process allow in order to enable Continuous Experimentation on a system?

*RQ*7 : What technical challenges are likely to emerge when Continuous Experimentation is applied to a resource-constrained system?

*RQ*8 : What capabilities should the software architecture provide in a resource-constrained system to enable Continuous Experimentation?

Prototype analysis. In this last phase, the goal was to create and evaluate a possible proof-of-concept for a cyber-physical system, more specifically an automotive platform, running a Continuous Experimentation cycle. That is the aim of RQ9 and RQ10 in Paper E, i.e., to qualitatively evaluate a prototype for Continuous Experimentation. The system and its software were developed as a result of a design study [19] aiming at proving the feasibility of an automotive Continuous Experimentation cycle. RQ9 analyzed the prototype in light of the findings of Paper C, while RQ10 reflected on the lessons learned from the designed experiment. The Research Questions were:

*RQ*9 : What software architecture can support a Continuous Experimentation decision process on a complex cyber-physical systems such as an automotive system?

*RQ*10 : To what extent do previously identified design criteria for Continuous Experimentation in the context of automotive cyber-physical systems hold?

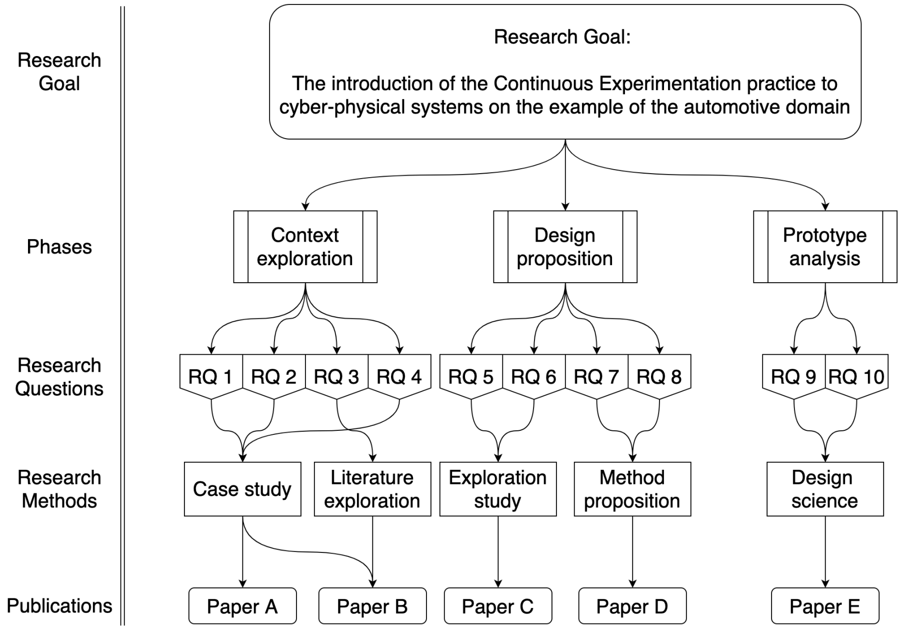
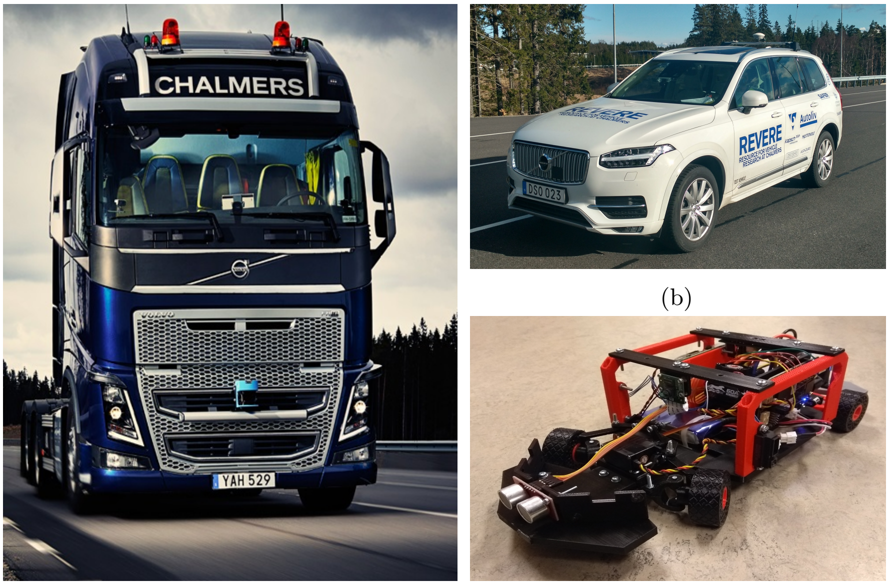


Figure 1.2: Representation of how the Research Goal was divided in thematic phases and Research Questions. Each Research Question was tackled with a Research Method and the work reported in one of the included publications.

1.3 Applied Research Context

The work was planned and performed in a research context that aimed at remaining close to real-world settings. This included using hardware capable of managing complex distributed software and the use of facilities and equipment provided by Chalmers University of Technology's vehicle laboratory "Revere" (Resource for Vehicle Research) [20].

The Revere laboratory hosts several projects in collaboration with research and industrial partners, with the goal of pushing forward the research in the context of autonomous driving and complex automotive and vehicular software systems in general. It houses a number of vehicles that are used in these projects, including for example a Volvo FH16 truck tractor, shown in Figure 1.3a, two Volvo XC90 SUVs, an active truck dolly with steerable axles, and a number of 3D-printed miniature vehicle that are used for educational purposes, shown in Figure 1.3c. Among these projects there were two that posed as context for



(b)

(a) (c)

Figure 1.3: Vehicles at the Revere laboratory: a Volvo FH16 truck tractor (a), a Volvo XC90 SUV (b), and an educational miniature vehicle (c).

the works included in this thesis: the COPPLAR project and the AutoFreight project.

The COPPLAR project focuses on a single vehicle, a Volvo XC90 SUV, shown in Figure 1.3b. The project aims at enabling the car to autonomously traverse the city of Gothenburg to connect the two campuses of Chalmers University of Technology. Since the envisioned route passes through the city center, the vehicle has to precisely and safely move on high-traffic and high-pedestrian density inner city roads. To do so, advanced sensing equipment is mounted on-board to provide the software with accurate data; it includes two industrial-grade computers, a 32-layers LIDAR scanner, a stereo vision camera, a GPS and IMU sensor, and vehicular radars. In this project context a number of personal contributions were realized. Several data collection runs were performed, instrumental middleware software layers were developed to enable high-level software communication with the vehicle's CAN network, and a number of high-level software modules were created, among which there were modules to perform drive-by-wire operations and Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication. Additionally, in the context of this project the work that resulted in Papers A to D included in this thesis was conducted.

The AutoFreight project is centered on a Volvo commercial truck tractor belonging to a logistics company which uses it for its daily operations. The tractor houses a distributed system comprising a main computer and a secondary unit, as well as cameras. The main unit is equipped with high-end consumer CPU and GPU, and is directly connected to two cameras. The secondary unit is an Accelerated Processing Unit (APU), connected to the main one and chosen to provide availability, redundancy, and troubleshooting options to the system. It is directly connected to another camera, so if an error or an issue arises on the main unit, the secondary one is still capable to provide a camera stream. Additionally, the secondary unit can be used to troubleshoot the main unit, since they are connected via an Intelligent Platform Management Interface (IPMI) port. Both units are connected to the truck CAN network, which provides GPS and IMU data. Finally, two 4G routers are available in the internal network to provide connectivity internally as well as to and from the external world. Since the vehicle is in use daily, the mobile connection is the only access to the system and the data, making this setup an ideal testbed for Continuous Experimentation themes. In this project, personal contributions were carried out to the design of the system architecture, shown in Figure 1.4, and the design, development and execution of the experiments concerning a prototype for Continuous Experimentation. In fact, in this project context the work that resulted in the Paper E included in this thesis was performed, as well as part of the work resulting in Paper B.

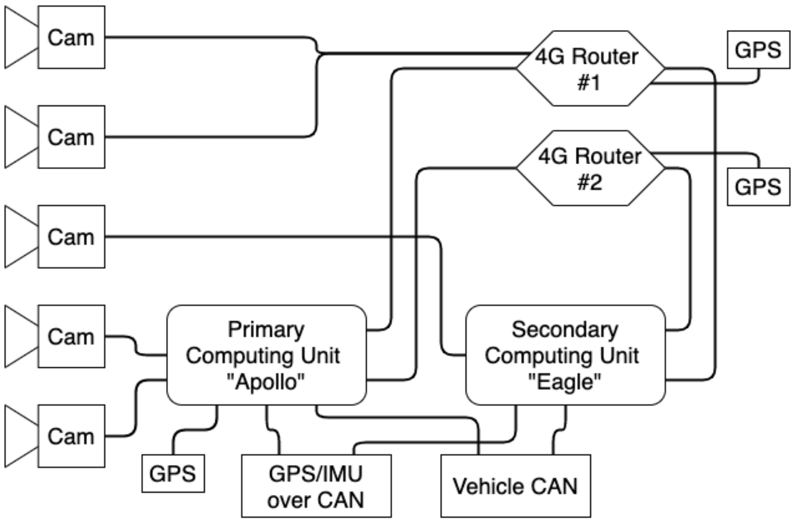


Figure 1.4: System architecture for the Voyager truck, in the AutoFreight project run at the Revere automotive laboratory. Picture included in [21].

1.4 Included publications

The present thesis includes five peer-reviewed publications, describing and reporting the work that took place in the past years. The articles are summarised in the following.

1.4.1 Paper A: The Automotive Take on Continuous Experimentation: A Multiple Case Study

This multiple case study aimed at collecting feedback and impressions from industry on the desirability of Continuous Experimentation in their field.

To achieve this, a series of workshops was organized at a number of automotive companies. Each workshop was structured in four phases as follows:

Phase I: After each participant presented himself and his role to the group, an initial presentation was shown. The goal of the presentation was to establish a common understanding and vocabulary of the Continuous practices, namely Continuous Integration, Continuous Delivery/Deployment, and Continuous Experimentation. This phase would take around 20 minutes;

Phase II: At the end of the presentation, the participants were asked the two aforementioned questions. They were given time to individually devise their answers, writing each idea on a note. This phase would take around 30 minutes;

Phase III: The participants took turns to go through their notes in order to explain to the group their meaning and the reasoning behind them. Each note was then placed near others on the same theme on a whiteboard, thus creating thematic clusters. This phase would take around 40 minutes;

Phase IV: An infrastructure model for Continuous Experimentation devised for companies with web-based products [10] was presented to the participants. They were asked to jointly discuss the model with the aim of identifying critical points and necessary changes if it had to be applied to the automotive industry. This phase would take around 15 minutes.

The data collected were divided in Advantages and Challenges, and the one or more companies that proposed each item was noted, allowing to highlight which ideas were more or less universally shared by practitioners.

This work concluded that while Continuous Experimentation was deemed desirable by the industrial practitioners there were many issues, such as technical and organizational challenges, still hindering the application of such techniques on real-world systems.

1.4.2 Paper B: Continuous Experimentation and the Cyber-Physical Systems challenge. An overview of the literature and the industrial perspective.

This article aimed at exploring the existing relationship between Continuous Experimentation and complex cyber-physical systems, adopting a dual approach. This work in fact approached both literature and industry via respectively a literature review and a multiple case study based on a previously published work.

The literature exploration was performed by searching relevant databases complemented by a snowballing phase [18]. The case studies were performed in a workshop format following the methodology of previously published work for homogeneity.

Together, the results provided an overview of the engagement of the Continuous Experimentation practice in the field of cyber-physical systems, uniting the analysis of the state-of-the-art achieved through the systematic literature review to the results of the multiple case study conducted with automotive industrial representatives.

1.4.3 Paper C: Design Criteria to Architect Continuous Experimentation for Self-Driving Vehicles

This design study explored the architectural properties that a software for cyber-physical systems, in the example of autonomous vehicles, should offer and fulfill in order to enable the Continuous Experimentation practice as a tool for quality improvement.

The proposed set of properties relate to both the software architecture and the software development process that needs to be in place to enable and facilitate Continuous Experimentation. These properties were chosen based on both the analysis of relevant literature of similar systems achieved in the past and the practical experience gained in the Revere automotive laboratory at Chalmers University of Technology [20].

Finally, the work described the software system that was used in the context of the aforementioned automotive laboratory, which was found abiding to the identified conditions.

1.4.4 Paper D: Considerations about Continuous Experimentation for Resource-Constrained Platforms in Self-Driving Vehicles

This exploration work aimed at identifying and assessing the effects that the lack of resources could have on the execution of the Continuous Experimentation practice.

To cope with the limitedness of the computational resources on resource-constrained systems aiming at performing Continuous Experimentation, this study defined and proposed three “execution strategies”, i.e., high-level scheduling strategies, to allocate execution time to an experiment while a different, more critical, software module runs at the same time. These strategies were called the *Parallel*, *Serial*, and *Downsampled* execution strategy. Lastly, the study proposes a new professional figure to the Continuous Experimentation model proposed by Fagerholm *et al.* [10], with the task of deciding which execution strategy is best suited to the envisioned experiment.

1.4.5 Paper E: Continuous Experimentation for Automotive Software on the Example of a Heavy Commercial Vehicle in Daily Operation

The aim of this design study was to provide and evaluate a proof-of-concept that would show the feasibility and benefit of a Continuous Experimentation decision cycle, based on previously identified design criteria, in the context of an automotive system.

To achieve this goal, a commercial truck tractor was equipped with a computing system comprising several cameras and GPS sensors, an IMU unit, as well as a mobile data connection. The vehicle was used in daily commercial operations, making the remote mobile connection the only entry- and exit-points for all communications to and from the system.

The system was used to demonstrate the execution of a prototypical Continuous Experimentation cycle to answer with a data-driven approach a software development question for an automotive system, used in daily operations by a logistic company. A set of previously identified design criteria to enable Continuous Experimentation on complex cyber-physical systems such as automotive vehicles was discussed in light of the system that was built for this work.

1.5 Discussion

The goal of the work reported in this thesis was to demonstrate that the Continuous Experimentation practice is feasible to be adopted in cyber-physical systems. Among the cyber-physical systems sub-disciplines, the automotive context was chosen because of its renewed interest towards software practices due to the recent efforts to achieve autonomous driving. Additionally, as the automotive companies increase their efforts to achieve new methodologies and technologies, the research laboratory Revere at Chalmers University of Technology offered the opportunity to explore this field in the context of the many projects run in collaboration with industry.

The work to achieve the Research Goal was articulated in several stages which build upon each other, as it is shown in Figure 1.5. A first step was to probe the field, which meant that investigations were run in the industrial context to register interest in the goal and to discover the obstacles that would prevent a straightforward adoption of the Continuous Experimentation practice. Then, the basis for a concrete solution was established. In this phase a number of properties deemed necessary in a system aiming at enabling Continuous Experimentation were defined, and possible solutions were proposed to counter some foreseeable issues. Finally, a prototype of a system capable of executing a Continuous Experimentation cycle was designed, developed, and tested in a real-life setting. It was evaluated against the proposed properties and it offered a first look into what capabilities are and will be necessary to facilitate and automate this process in future, more sophisticated systems.

The field investigation highlighted how practitioners in a complex cyber-physical systems sub-field such as the automotive field think about the adoption of Continuous Experimentation. The results of the multiple case studies, described more in details in the corresponding articles, reported the practitioners' interests and expectations in terms of achieving faster time-to-market and shorter feedback loops from the systems themselves, the possibility of monitoring their products, and for what concerns the customers, an increased satisfaction for the provided software features. It was also apparent that they expect several challenges to arise before the benefits of Continuous Experimentation could be reaped, and interestingly there was more agreement across companies on the challenges than on the possible advantages. From the discussions in the context of the multiple case studies it never emerged that the interviewed companies had a development practice similar to Continuous Experimentation already in place.

The systematic literature review, also part of the context investigation phase, highlighted the state-of-the-art of Continuous Experimentation as applied to cyber-physical systems. The result of this review was a research landscape focused mainly on empirical studies gathering information from practitioners, conceptual approaches to the main challenges that they do or will face, and architectures for future systems capable of performing Continuous Experimentation. Only a smaller number of studies instead proposed new techniques to practically solve or circumvent challenges. These findings concurred to a very large extent with what other secondary studies reported about the research landscape for the Continuous Experimentation practice generally [12].

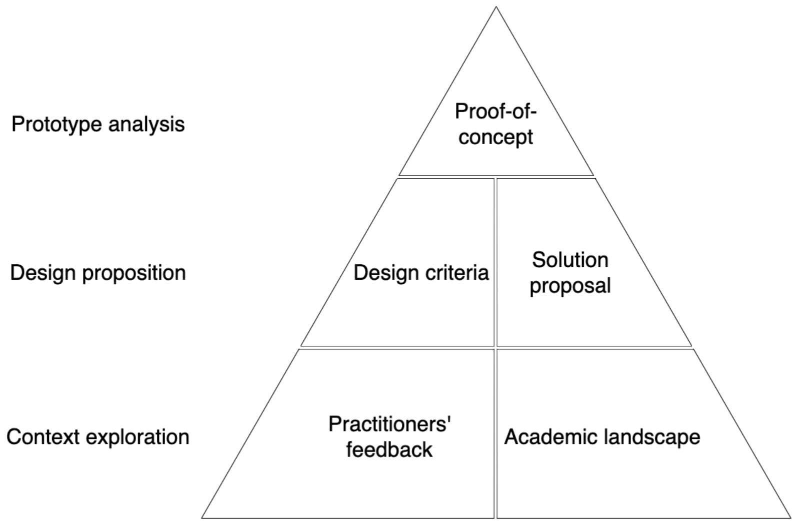


Figure 1.5: The results of the included papers, with each phase building on the previous one. The base layer comprises the exploration of the field, performed with empirical research and a literature study. On top of that, theoretical contributions aim at helping future efforts by proposing enabling characteristics and solutions that could enable Continuous Experimentation on cyber-physical systems. Finally, a software artifact is proposed as a proof-of-concept for Continuous Experimentation on cyber-physical systems, on the example of automotive systems.

The following phase of the work consisted in laying down the theoretical groundwork in light of future artifacts fulfilling the overarching goal of the thesis. Comprising this stage are two studies, both of exploratory nature. The first one examined in literature the architectures of complex cyber-physical systems such as self-driving autonomous systems and their characteristics, in order to extract properties that would enable both that autonomous functionality and the Continuous Experimentation practice. Additionally it described useful properties for the software development process that would enable and facilitate the Continuous Practices at the root of Continuous Experimentation. Thus, the result is a set of desirable properties for both the architecture and the development process. Finally, the software architecture adopted in the Revere automotive laboratory at Chalmers University of Technology is described. The development of this architecture was influenced by many of these qualities to begin with, so it complies naturally to the extracted properties.

The second study of this stage, Paper D, focused on a different aspect. The Continuous Experimentation practice introduces a computational overhead into the system performing it as it requires to perform data monitoring and/or collection, in addition to running and managing the experiment software module that is running alongside or in substitution of the production software module. This is not an issue when Continuous Experimentation is performed on web-based systems as those systems are running on server farms, which are high-performance systems composed of many servers working together and well capable of spawning additional virtual machines that unlock new hardware computational resources when necessary. Of course there are no such luxuries when the same goal has to be achieved in the field of cyber-physical systems, where the computational units are usually low-energy and/or low-performing platforms. Cyber-physical systems are physically bound systems and it is often the case that they are packaged in small hardware platforms, which has an impact on how much computational power they can have access to. The same issue is present and even exacerbated when considering the automotive field since the computational units used in this context are employed in an economy of scale where costs are multiplied for each produced unit, meaning that to keep the costs as low as possible the hardware capabilities for each unit are kept at the minimum level to run reliably. The result of this work was the proposal of three *execution strategies*, i.e., experiment scheduling patterns, to allow software experiments to run despite the limitation of hardware resources. The choice of which strategy is possible or advisable to adopt was deemed a strategic one; for this reason, related to the introduction of this new parameter, a previously established organizational model for Continuous Experimentation was extended with a new professional figure managing the newly introduced decision point.

The last phase of the work comprised one article, which described an artifact representing the culmination of this research path. The artifact is a computing system housed in a commercial vehicle, connected to cameras, GPS sensors, and 4G routers that provide external connectivity. As the vehicle was in daily operation by its owner, a logistics company, the remote data connection was the only access point to the whole system. The system architecture was designed with redundancy in mind, to minimize the downtime should a hardware issue arise. The software was instead designed to use the camera feed to demonstrate the feasibility of a Continuous Experimentation cycle to answer an algorithmic question. As shown in Figure 1.6, a *Supervisor* module manages the interactions between the external world and the experimentation context, being the recipient of an *Experiment protocol* describing the parameters of the experiment and later sending back relevant data about the session. The Supervisor is in charge of monitoring the *Production* software module and one or more *Experiment* software modules, deciding when an experiment can start, when it should *degrade* its performance to utilize less resources thus leaving more to the Production software, and when instead the experiment is in violation of safety parameters and must be aborted. In addition, the Supervisor module is the one collecting and sending the experiments' results back to the developers, represented by the *HQ* box in the image. It is worth noting that the dashed lines in Figure 1.6 represent Over-the-Air communication. Once received, the resulting data are analyzed to compare experiments' and Production software's performances to decide in a data-driven way which software module achieved better performances according to the experiment goals.

This proof-of-concept was additionally compared against the properties identified in a previous work during the Design proposition phase and those criteria were validated in light of the proposed artifact. Connecting to the previous work on the execution strategies, in this context the choice was made to adopt the *parallel* strategy, meaning that both production software and experimental software were running at the same time. This was made possible by the fact that the dedicated computing hardware that was installed in the system had enough resources to allow to execute both types of software modules without having them to compromise on their utilization.

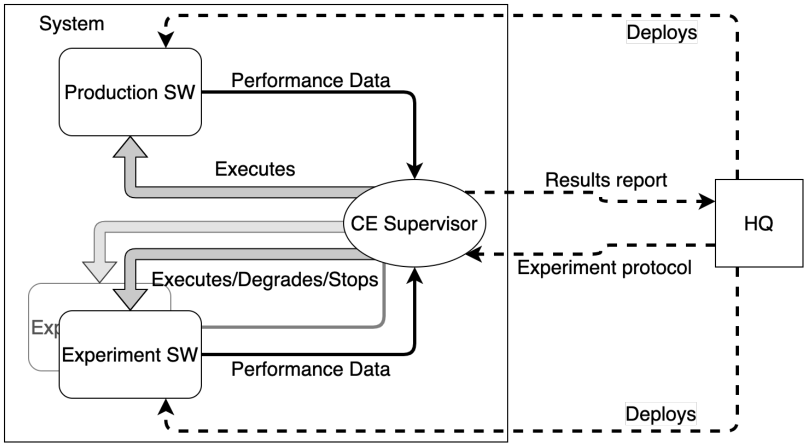


Figure 1.6: Software architecture for the experimentation system running in the AutoFreight project. Picture included in [21].

The relevance of the work here described lies firstly in the study of Continuous Experimentation as an option for a different field than web-based software. This not only pushes forward our knowledge of the interest and obstacles for practitioners, but also shows how primary studies approached this practice so far when considering a field of application much more dependent on physical constraints such as cyber-physical systems. Additionally, it proposes new ideas and principles on how such applications could be achieved, with the goal of posing the basis for future efforts aiming at further extending our knowledge and the practitioners' command of this practice. Finally, it validates their feasibility with a real-world proof-of-concept, showing that such a system is already a practical possibility under relaxed constraints.

To the best of the author's knowledge, systematic adoption of this practice on automotive systems or cyber-physical systems is still not a reality due to the several unsolved challenges. Among these, the need to understand the implications that running additional control software could have from a safety point of view; or the unpreparedness of the great majority of vehicles by traditional manufacturers currently in traffic, many of which are several years if not decades old and were not meant to have the hardware nor the software capabilities to run advanced software practices. However, there are indications that at least one manufacturer of electric vehicles is moving in this direction and performs operations that sound quite similar to what was described so far. A US-based electric vehicle manufacturer mentioned already the systematic gathering of driving and sensor data via “field data feedback loops” that are used to “enable the system to continually learn and improve its performance” [22]. While they do not mention explicitly software experiments, one of their representatives mentioned the policy of installing “an ‘inert’ feature on vehicles” in order to “watch over tens of millions of miles how a feature performs” by logging the new software’s behavior in a real-world scenario before rolling it out to production [23]. These revelations validate the line of thought emerging from the work so far performed, i.e., that the Continuous Experimentation practice is indeed a useful tool that can prove as beneficial to cyber-physical systems and automotive systems as it is to web-based software, although there are technical challenges to overcome in terms of performances and safety implications.

1.6 Conclusions and Future Work

This thesis reports the work and contributions achieved by the author in the field of Continuous Experimentation on cyber-physical systems, with a focus on automotive systems. A number of articles are included to offer greater details on how the respective studies have been conducted. The Research Questions that guided the work are gathered as follows, together with the contributions that aim to answer them:

RQ1, *RQ2*, and *RQ4*. To answer these questions a number of case studies were performed with the aim to show the interest and concerns about Continuous Experimentation of practitioners in the automotive field, which was chosen as a practical example of cyber-physical systems. From these meetings a number of prospective advantages and challenges were obtained that allow to have a detailed understanding of the practitioners’ expectations and concerns towards the topic. Further details can be found in Papers A and B;

RQ3. A systematic literature study tackled this question. The study offered a view on the focus of literature on the topic of Continuous Experimentation applied to the cyber-physical systems context. While Continuous Experimentation is now known and increasingly examined in its native field of web-based systems, the limited amount of relevant literature found when investigating it in the cyber-physical systems field is an indication of the presence of a research gap due to the novelty of this practice. This work can be found in Paper B;

RQ5. A set of design criteria for cyber-physical systems aiming at achieving Continuous Experimentation were investigated to answer this Research Question. These items were obtained by analyzing previous works in literature and from the experience collected in the projects that run in the context of the automotive research laboratory at Chalmers University of Technology, Revere. The criteria collected in this work can be found in Paper C;

RQ6. A number of desired characteristics to enable Continuous Experimentation in the software development process for cyber-physical systems were suggested. These characteristics were devised and collected similarly to what was done with the previous Research Question, and can be found in Paper C as well;

RQ7 and *RQ8*. The issue of low computational resources was tackled by these Research Questions which focused on how to enable Continuous Experimentation on a resource-constrained device given that this practice requires some additional computing efforts to manage the experiments and data collection. This challenge and envisioned solutions were investigated in Paper D. Additionally, the work proposes a modification to a previously known model for Continuous Experimentation in order to adapt it to cyber-physical systems. This model was not developed for cyber-physical systems, so it was not considering the added complexity of the decision process involved in the choice of a suitable strategy to run Continuous Experimentation when the target systems faces these technical challenges;

RQ9 and *RQ10*. These two Research Questions analyse the proof-of-concept for Continuous Experimentation that is proposed in Paper E. A software architecture is proposed to run and manage the execution of Continuous Experimentation and its characteristics are compared to the design criteria identified in Paper C. Reflections on the prototype and the steps that are still missing to make it a viable option for commercial systems are discussed.

While these contributions aim to bring academia and industry closer to the application of Continuous Experimentation in this field, it can be expected that achieving this goal will take more research work ahead in order to solve the technical and practical challenges preventing a systematic adoption, e.g., the resources management on vehicles running experiments, the certification problem to make sure experiments are safe to run, or even the organizational aspects of companies used to more traditional ways of working that may need to adapt to this novel approach.

For this reason possible directions for future works will not involve only technical aspects. One clear problem that needs more research and validation is surely the issue of running experiments on vehicles without having to upgrade its computational equipment. Doing so would enable many of the currently existing vehicles to run Continuous Experimentation without the need to wait for next generation smart(er) vehicles. Such findings would likely transfer easily to cyber-physical systems sub-fields different than the automotive systems, enabling even more products to adopt Continuous Experimentation as a tool to improve and guide the software evolution process. Moreover, the safety aspects should be investigated further as it will be necessary to certify the safety of a system capable of running experimental software. A possible first solution would be to *sandbox* the experimental functions in order to guarantee that they cannot interfere with the rest of the system, but perhaps there could be more sophisticated ways to approach the problem that would allow the software under test to take a more active role in the system without compromising safety. Finally, it could prove interesting to investigate the organizational changes that could be required to adapt automotive companies, which come from a highly regulated hardware-focused background, to a software-centric and more agile approach to product development.