

An architecture for enabling A/B experiments in automotive embedded software

Yuchu Liu
Volvo Cars
Gothenburg, Sweden
yuchu.liu@volvocars.com

Jan Bosch
Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden
jan.bosch@chalmers.se

Helena Holmström Olsson
Computer Science and Media Technology
Malmö University
Malmö, Sweden
helena.holmstrom.olsson@mau.se

Jonn Lantz
Volvo Cars
Gothenburg, Sweden
jonn.lantz@volvocars.com

Abstract—A/B experimentation is a known technique for data-driven product development and has demonstrated its value in web-facing businesses. With the digitalisation of the automotive industry, the focus in the industry is shifting towards software. For automotive embedded software to continuously improve, A/B experimentation is considered an important technique. However, the adoption of such a technique is not without challenge. In this paper, we present an architecture to enable A/B testing in automotive embedded software. The design addresses challenges that are unique to the automotive industry in a systematic fashion. Going from hypothesis to practice, our architecture was also applied in practice for running online experiments on a considerable scale. Furthermore, a case study approach was used to compare our proposal with state-of-practice in the automotive industry. We found our architecture design to be relevant and applicable in the efforts of adopting continuous A/B experiments in automotive embedded software.

Index Terms—A/B Testing, Automotive Software, Embedded Software, Software Architecture

I. INTRODUCTION

A/B experimentation or A/B testing is a method for evaluating software changes in a quantifiable manner. Continuous A/B testing is an important method in understanding and delivering measurable customer value. Many web-facing companies have demonstrated success from A/B experiments, such as Booking.com [1], Google [2] and Microsoft [3]–[5], just to list a few. With the digitalisation of the automotive industry, software is becoming a main differentiator of products [6]. A/B testing is an effective tool to evaluate software and support organisations in making data-driven decisions [7]. However, the adoption of continuous A/B experiments in automotive embedded software is not without challenges.

Embedded software has hardware constraints. Such constraints could manifest as limitations to computational power [8], long release cycles [6] and often dependency on suppliers [9]. Data collection and handling is also believed to be challenging in the automotive specific applications [9], [10].

Although a fair number of publications point out the challenges in A/B experiment adoption [6], [8]–[10], we identified a gap in the literature concerning architectural solutions to enable A/B experiments. Furthermore, there is little to no reports on concluded or ongoing online A/B experiments in the automotive domain.

In this paper, we present an architecture that enables A/B experiments in the automotive domain and aim to address the challenges that are unique to this industry. We present a literature review of A/B experiment architecture in embedded and web-facing environments. Moreover, we conducted a case study of the architecture applied at scale and to report the state-of-practice of A/B testing in automotive. Compared to the existing literature, the contribution of this paper is two-fold. First, we present an architecture that enables A/B testing automotive software. We reviewed the literature and did not find a similar architecture for A/B experiments. Secondly, we apply this architecture in practice, in fleets of considerable scale. We present the case study and state-of-practice of two other automotive companies.

The rest of this paper is organised as following. In section II, we introduce the unique constraints in automotive industry for A/B testing. In section III, we present our research method. We summarise the existing A/B experiment frameworks and architecture in section IV. In section V, we present our architecture design along with the case studies. Discussions and conclusion are presented in section VI and section VII.

II. BACKGROUND AND CONSTRAINTS

In this section, we introduce the background on A/B testing and list the constraints of adopting the method in automotive embedded software.

A. Background

A/B testing is a type of continuous experimentation where users or systems are split into subgroups and issued with different variants of the same software. By studying the response from each cohorts, A/B experiments can guide product

development in an effective manner [1]–[3]. Typically, eligible users are split into two groups, the A version (control) and the B version (treatment). For both user groups, their interactions with the functions are recorded and evaluated based on a set of carefully designed metrics reflecting business and/or customer values [11].

Almost all well-established A/B testing frameworks are for web-facing businesses. Such frameworks or models cannot be applied directly in an embedded environment as they do not address specific challenges. These challenges come from many aspects, they can be technical, business, and organisational as demonstrated by Mattos *et al.* [6]. As embedded software often has dependency on hardware, fast software release becomes difficult to accomplish [8]–[10]. Although challenging to adopt, many advantages of continuous experiments that were proven in the web-facing businesses are also expected in the automotive industry [10].

B. Constraints

In addition to the challenges summarised by relevant literature [6], [9], [10], we list the specific constraints in automotive which motivate our architecture design. Automotive embedded software is distributed to hundreds of Electronics Control Modules (ECUs). These software are traditionally developed using the "V-model" where the OEMs deliver specifications and suppliers deliver implementations [12]. This model has exhibited its limitations.

1) *Release cycles and speed*: Combining the strict standards with the growing complexity, the automotive software release process is rigid. First, the development and release of automotive embedded software is usually strongly dependent on suppliers. Secondly, automotive companies have traditionally designed software release cycles based on their hardware release process [13]. This process cannot handle rapid changes, as all integration and tests are planned at fixed periods. Moreover, the most commonly adopted automotive software architecture AUTOSAR¹ lacks flexibility in partial updates [9]. If the new software is not backwards compatible, all ECUs in the vehicle need to be updated. Last but not least, updating software which are governed by legislation might require renewal of certifications, which will add delays to the software release process.

2) *Sample size and management*: Controlling boundary conditions is impossible for online experiments, as vehicles can be driven to everywhere and at anytime. Therefore, to conclude sufficient treatment effects, A/B experiments need be conducted on large and randomly selected sample groups. This large group of users needs to be managed as online experiments require a flexible configuration of A/B or A/B/n groups. However, the sample groups are difficult to manipulate when the software needs to be updated through physical contact with the cars. Same challenge could be experienced when an A/B test is concluded, and the software needs to be inverted to the original version.

Managing sample groups longitudinally can be burdensome. Performance of some automotive functions depends on temporal factors and has seasonality effects, thus experiments need to be conducted longitudinally. Therefore, the ability to orchestrate the A/B groups over time is beneficial.

3) *Data infrastructure*: To conclude a casual effect of the treatment, data collection for A/B experiments requires certain level of accuracy. Storing such data locally in each vehicle is not feasible, as it becomes difficult to access and it will require a large memory on-board. The success of an A/B experiment is largely relied on appropriate assumptions when designing an experiment and fast feedback when conducting one. Sharing data within a large organisation can be problematic [14]. In order to maximise the data, all development teams need to have easy access to relevant data. As a result, companies suffer from misrepresentation of customer values.

4) *Safety requirements and fallback*: Automotive software has high safety requirements. In an A/B test, all alternative versions can never obstruct such requirements which might affect road safety and/or legal compliance. The functional requirements need to be safeguarded while ensuring a continuous release of alternative versions seems impossible today. Another practice to decrease hazards on the road is to have built-in fallback for safety critical functions. For instance, one could install both the A and B alternatives on-board. Then the A alternative can be used as a fallback when it is thoroughly tested and validated.

III. RESEARCH METHOD

In this paper, we combine a literature review with case studies. We studied several existing A/B experiment frameworks inside and outside of the industry through literature reviews, to compare our approach to existing frameworks. Furthermore, to validate the architecture designed, we conducted case studies based on a series of ongoing efforts in A/B experiments from three separate automotive manufacturers.

We explore the following research question:

RQ How can we continuously experiment with automotive embedded software providing the challenges and limitations that are unique to this industry?

A. Literature review

This literature review is done to understand existing A/B experiment frameworks within and outside of the automotive domain. To identity and explore work that is relevant for the research question, we follow the methodology described by Kitchenham [15].

1) *Data collection*: We included the following terms in our search query: ("A/B testing" OR "A/B experiment" OR "online experiment" OR "bucket testing" OR "continuous experiment") AND ("software architecture") AND ("embedded software" or "automotive software"). Alternative terms are included as there is no standard terminology. Keyword combination with "automotive software" yield no meaningful results, thus we expanded the search query to also include embedded software. The databases included in our search

¹<https://www.autosar.org/>

process are IEEE Xplore, ScienceDirect, and Google Scholar, returning a total of 104 results excluding duplicates. To ensure the results are relevant today, we limit the publications to the recent ten years.

2) *Inclusion criteria:* Each paper resulted from the search process was reviewed by at least one of the authors. We examine the keywords, abstracts, and the body of the paper to identify A/B experiment frameworks and the applicable sector for said frameworks. We selected publications which focus on A/B experiment architecture and/or framework from embedded applications. We did not include publications discussing the benefits or challenges or feasibility of A/B testing. This inclusion criteria resulted in a total of three papers. Since the technique is well established in web-facing applications, we included work on A/B testing framework in the web domain. A total of 11 publications included in this review are [1]–[5], [8], [13], [16]–[19].

B. Case study

Following guidelines from Runeson and Höst [20], we conducted two sets of case studies with three separate automotive companies. In study I, we examine the proposed architecture in practice on a cloud-based A/B experimentation in a vehicle fleet at scale. We study the architecture for A/B testing in a fleet from one of the three companies. The software for case study I was developed in-house in company A. As online experiments are not commonly applied in the industry, to the best of our knowledge, there is a lack of quantitative data to study from. To understand the state-of-practice, we conducted semi-structured interviews with two more OEMs as case study II.

1) *Case study attendees:* The three companies included in the case studies are large OEMs. In each company, we conducted interviews and workshops with at least five different employees from each company, working with varying aspects of software development. Their roles include software engineer, software architect, product owner, data engineer and data scientist.

2) *Data collection:* One of the authors was actively involved in the experimentation design from ground up and supported the entire process. We document the process through meeting notes and design specifications in the project. The questions from case study II were specifically designed to understand the current state-or-practice of A/B experiments in an automotive setting. We also aim to understand the potential of cloud-based A/B testings in each company. During the interviews, we presented our architecture design to the attendees along with questions regarding current practices adopted in their companies. All the interviews were conducted by at least one of the authors. The responses were documented as meeting notes, which were distributed to the interview participants.

We recognise the limitation of our case study approach, as the results of our case studies were obtained from three companies. The outcome can be specific to these companies

		A/B experiment environment	
		Web	Embedded/OS
Variant generation	Software change	[1], [4], [5] [16], [17], [18],	[5], [8], [13], [16], [18], [19]
	Parameter change	[2], [3]	

Fig. 1. Existing A/B experiment framework categorised by environment and variant generation methods.

and without further investigation, we cannot generalise the conclusion to the automotive industry.

IV. EXISTING ARCHITECTURES

In this section, we present the results from our literature review. We included 11 publications [1]–[5], [8], [13], [16]–[19] that focus on describing A/B experiment architectures, in both embedded software and online applications. From our literature review, we have discovered that there is a general gap in the literature on architectures or frameworks designed specifically for automotive software. Based on the topic, we summarise the papers into four overlapping categories. They are grouped firstly by their environment, i.e., embedded or web-facing. Paper [5], [18] are applicable for both groups. We include OS and embedded applications in the same category, as they share many common challenges for instance, the devices can be offline [5]. Second, we identified in these papers how a software variant is shipped to the users. Namely, if a complete software change is required, or variants can be introduced through parameter changes. The categories are presented in Figure 1. As can be seen, variant introduction through parameter change is not a widely explored method within embedded software.

Although the design process is vastly different, there are a numbers of shared components for embedded and web experiment architecture. This includes experiment configuration, data collection, experiment analysis and metrics evaluation. Therefore, some experiment models can be employed in various environments including web, operation systems and embedded [4], [18]. Tang *et al.* [2] and Kohavi *et al.* [3] both report a multi layered experiment configuration system that can handle multiple A/B experiments. Users will be assigned to A or B variant in a consistent manner [4], [19]. In the web environment, this is achieved by assigning unique IDs when users visit the web pages.

Data infrastructure is a major component in any experiment framework. All researchers include data infrastructure as part of their experiment frameworks, particularly focused on trustworthiness [1]–[4], [18], [19]. Such data collection is also required in embedded environments, however, is more difficult

due to hardware limitations [13]. An experiment architecture [16] for automotive software used an on-board data storage before uploading the data through the vehicle's telemetry.

Another key element for A/B experiments is rapid software release. We found that all architectures for rapid experiments in an embedded environment rely on continuous deployment. The "RIGHT Model" discuss that if a function is novel, continuous deployment might not be necessary [18]. However, most frameworks in embedded environments [8], [13], [16] require a well-established continuous deployment process to achieve rapid experimentation loops. Software variant release through Over-the-air(OTA) can increase delivery speed in automotive applications [16]. In the web environment, rapid experimentation can be achieved more flexibly through an array of mechanisms. For example, an offline and online experiment systems in Netflix, as demonstrated by Amatriain [17]. Existing data can be used to train the models before they are introduced to an online experiment, which allows faster and cheaper evaluation of software. Another technique in increasing experiment speed is using parameter updates as mentioned by Tang *et al.* [2]. The A/B variants in target functions are parameterised and configured through data files. These parameters are changed more frequently than code, which enables fast experiments provided the parameters exist.

Furthermore, to fully utilise the benefits of A/B testing, all papers highlighted the importance of the organisational and cultural mindset of making data-driven decisions. Kohavi *et al.* [3] summarise prerequisites which an organisation needs to adapt, highlighting the importance of data-driven decision making mindsets. In the "Experiment Growth Model" introduced by Fabijan *et al.* [1], all components of their A/B experimentation model become more mature as the entire organisation evolves through different stages.

V. ARCHITECTURE AND CASE STUDIES

In this section, we present a software architecture that could enable A/B testing in the automotive domain. A hybrid architecture is presented. The essence of the architecture is to imitate an online environment for an otherwise offline application. In doing so, automotive A/B testing can benefit from the flexibility of online experiments. We present the components of our architecture in Fig.2.

A. System characteristics

We present a hybrid architecture (Fig.2) combining on-board and cloud functionalities. The system is composed of six main components. These are parameterised functions, a release process which most companies have in place. There is a cloud host that writes parameters to the vehicles and collects data from the vehicles. Finally, a centralised data storage and pipeline for distributing the measured data.

The system workflow can be described as follows. First, a function which characteristics can be defined by a list of parameters is delivered. There are two sets of parameters embedded in the function, the local set, which is the default and the cloud set that can receive incoming values

externally. The benefit of parameterisation in A/B testing was also highlighted by [2]. A set of observables which measure function performance is also predetermined. The function and its parameters are delivered to a release process which will integrate with other functions and release the software to vehicles. This release and installation of software can be done through workshop visits or OTA.

Once the software is introduced to the vehicles, users are identified through Vehicle Identification Numbers (VIN), which is unique and comprised of vehicle meta-data. This ensures that although the software is introduced to all cars, no experiments will be conducted unless the users are deemed eligible in advance. Upon key-on of a vehicle, a vehicle will send its VIN to the A/B test cloud. Since the A and B groups are configured in the cloud, the test cloud will match the VIN and then return a status indicator to the vehicle. Ineligible cars will have no match in the cloud and receive no response. For all eligible vehicles, they can be partitioned into A and B groups through remote configuration. The control group will use the functions' local parameters and the treatment group will receive cloud parameters. Since the parameter names are predefined, the vehicle cannot accept any other values, thus increase security. Furthermore, as the cloud parameters are blank values in the vehicles, cloud parameter change can be done remotely. This design enables function behaviour change through parameters provided the parameters exist. Development teams can continuously A/B test and adjust the existing parameters without complete software change and independently from the company-wide release cadence. A complete software update is required only when new parameters need to be added.

Data collection is done through the cloud and it measures a set of predefined observables. The observables are measured and temporarily stored on board, then sent to the cloud at time intervals while driving. This data are collected in a centralised data lake, cleaned, then distributed to development teams. During a trip, time series data is collected for dynamic observables. For stationary observables, only one or a few snapshots are measured. After analysis of the A/B tests, further actions can be taken such as adjusting cloud parameters, re-partitioning A/B groups, or concluding the experiments. When the experiments are concluded, the connection to the cloud will be interrupted and vehicles will invert back to the local parameters automatically. Moreover, the local variant always serves as a safety fallback in critical situations.

B. Case study

The first case study was performed in company A on an energy management (EM) function that was developed internally. The function Energy Management has a local and a cloud set of parameters which determine the local and cloud energy management strategy, respectively. By default, the vehicle will always run the local strategy unless a connection to the cloud is established and the vehicle is eligible. The development team delivers the software through the company's existing release process. There are 50 vehicles in this fleet of company cars

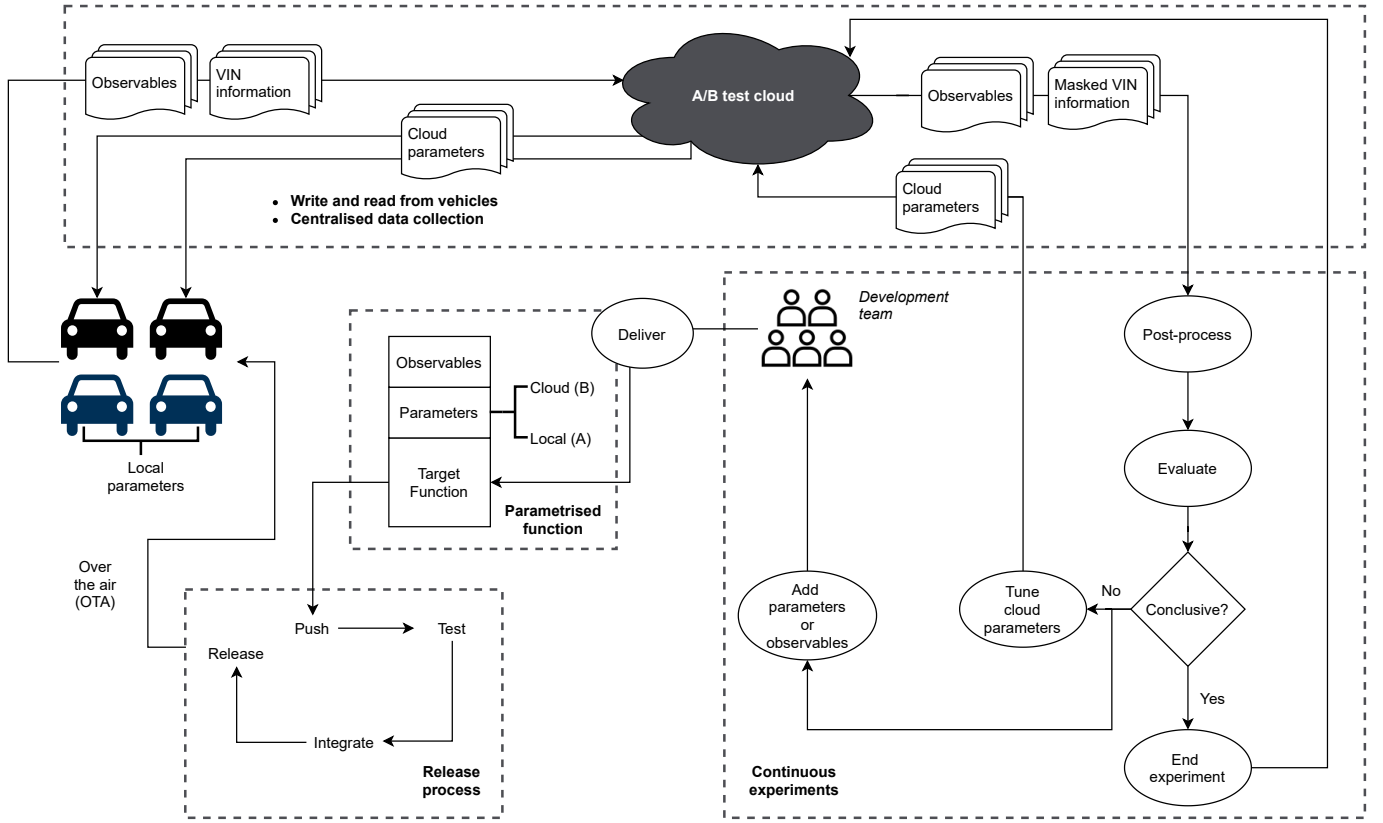


Fig. 2. Process of the cloud-based A/B test architecture, illustrating the general work flow of conducting an A/B experiment with parametrised functions.

driven for a total 18 month period, during which, there were 58 observables measured. The experimenters also monitor how frequently the users manually interrupt the cloud connection.

A number of automated mechanisms were put in place in the cloud to ensure data quality. The experimenters have access to the data collected in real-time. The data collected was post-processed in an automated manner in a database, while the team can also choose to export the raw data. The file size of data collected per week averages at 1.7 gigabytes when exported in CSV format. The EM function software has dependencies on six ECUs that are mostly supplier parts. Traditionally, changing the software means rebuilding of all these ECUs completely through suppliers and downloading the software to the vehicles physically. The usual lead time for such changes is anywhere from three month to one year. This system caters an environment where continuous experiments are independently from release processes that could be lengthy at times. In average, the total distance travelled by all eligible users is over 18,000 kilometres per week and over 80% of the vehicles are being driven daily. Comparing to any test fleet, they are generating measurements at a much larger scale.

The second case study is conducted to understand the state-of-practice in company B and C. Although neither company has experience with large scale A/B experiments, but there are commonalities in the components. Through our interviews, it was apparent that company B and C have adopted some level of capabilities, specifically the data collection capabil-

ities. Company B has invested intensively in an online data collection system for their vehicles. A set of observables are measured, their data collected and distributed to the corresponding development teams through a centralised database. Each functional team within the company can also request for more observables to be measured from the fleet. A similar approach was reported by company C. A centralised database was built to distribute high quality data in a fast manner. The teams have the freedom to determine the sampling frequency accordingly to their measurement requirements.

VI. DISCUSSION

In this paper, we presented a hybrid architecture that enables continuous A/B experiments in automotive embedded software. Comparing to the existing A/B experiment architecture, our architecture offers the flexibility of being independent from continuous deployment processes. By allowing parameter changes, functional changes can be experimented continuously without a complete software change.

However, we foresee some potential weakness in the design and they are discussed here. Firstly, the threshold of functional behaviour change through parameters is low comparing to a complete software change. The system enables A/B experiments for fine tuning of functions but not complete concept changes. Secondly, many parameter changes are not independent from each other in an automotive setting. When multiple experiments are running simultaneously, the

configuration of experiments becomes critical as suggested by [2] and [3] from their experience in online businesses. Similar to web-facing applications, we need to consider contradicting and hierarchical functions and their parameters. Performance of contradicting or hierarchical software variants cannot be determined individually. Therefore, some centrally well-established and understood performance metrics need to be put in place before parallel/multiple experiments can be conducted. Thirdly, the teams shall coordinate their experiment design when parameters or observables are shared between different functions. Such coordination requires organisational support [1]. As many automotive companies are going through agile transformation [9], the data-driven development mindsets and support structure are gradually improving. The speed of the transformation will influence how quickly an A/B experiment framework can be implemented at scale.

Finally, receiving cloud parameters requires an active internet connection. Although functions can be safeguarded by using local parameters as fallback, functions which require millisecond response time cannot rely on cloud connection. A possible setup for time critical functions could be, one embeds the A and B versions of parameters in the software itself, and use the cloud to trigger the switch in between them. As a trade-off, one will lose the freedom of tuning cloud parameters without complete software updates.

VII. CONCLUSION

In recent years, some research effort was put in the adoption of A/B experiments in the automotive domain [8]–[10]. In this paper, we raised a research question on how to enable continuous experiments in an automotive, and presented an architecture that demonstrated such capabilities. Through a literature review, we found that embedded experiment architectures share many components with web-facing ones, however, lack the capability of rapid changes. The architecture design is a hybrid A/B testing model that address many challenges in the industry. Comparing to existing frameworks, our hybrid architecture enable rapid software changes without compromising the high safety and security standards. Similar framework for automotive software A/B testing is not previously discussed in literature. We shared case studies of cloud-based A/B experiments at scale, which shows high potential of the parameterised hybrid architecture. The components of our architecture were compared with the state-of-practice of two other large automotive manufacturers. We found that the case study companies have applied many components, thus paving the way to an A/B experiment capable architecture.

REFERENCES

- [1] A. Fabijan, P. Dmitriev, C. McFarland, L. Vermeer, H. H. Olsson, and J. Bosch, "Experimentation growth: Evolving trustworthy A/B testing capabilities in online software companies," *Journal of Software: Evolution and Process*, vol. 30, no. 12, p. e2113, nov 2018.
- [2] D. Tang, A. Agarwal, D. O'Brien, and M. Meyer, "Overlapping experiment infrastructure: More, better, faster experimentation," in *Proceedings 16th Conference on Knowledge Discovery and Data Mining*, Washington, DC, 2010, pp. 17–26.
- [3] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann, "Online controlled experiments at large scale," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*. ACM Press, 2013.
- [4] S. Gupta, L. Ulanova, S. Bhardwaj, P. Dmitriev, P. Raff, and A. Fabijan, "The anatomy of a large-scale experimentation platform," in *2018 IEEE International Conference on Software Architecture (ICSA)*. IEEE, apr 2018.
- [5] P. L. Li, P. Dmitriev, H. M. Hu, X. Chai, Z. Dimov, B. Paddock, Y. Li, A. Kirshenbaum, I. Niculescu, and T. Thoresen, "Experimentation in the operating system: The windows experimentation platform," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, may 2019.
- [6] D. I. Mattos, J. Bosch, and H. H. Olsson, "Challenges and strategies for undertaking continuous experimentation to embedded systems: Industry and research perspectives," in *Lecture Notes in Business Information Processing*. Springer International Publishing, 2018, pp. 277–292.
- [7] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The benefits of controlled experimentation at scale," in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, aug 2017.
- [8] F. Giaimo and C. Berger, "Design criteria to architect continuous experimentation for self-driving vehicles," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, apr 2017.
- [9] D. I. Mattos, J. Bosch, H. H. Olsson, A. M. Korshani, and J. Lantz, "Automotive A/B testing: Challenges and lessons learned from practice," in *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, aug 2020.
- [10] F. Giaimo, H. Andrade, and C. Berger, "The automotive take on continuous experimentation: A multiple case study," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, aug 2019.
- [11] P. Dmitriev, S. Gupta, D. W. Kim, and G. Vaz, "A dirty dozen: Twelve common metric interpretation pitfalls in online controlled experiments," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1427–1436. [Online]. Available: <https://doi.org/10.1145/3097983.3098024>
- [12] K. Forsberg and H. Mooz, "The relationship of systems engineering to the project cycle," *Engineering Management Journal*, vol. 4, no. 3, pp. 36–43, sep 1992.
- [13] J. Bosch and U. Eklund, "Eternal embedded software: Towards innovation experiment systems," in *Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change*. Springer Berlin Heidelberg, 2012, pp. 19–31.
- [14] A. Fabijan, H. H. Olsson, and J. Bosch, "The lack of sharing of customer data in large software organizations: Challenges and implications," in *Agile Processes, in Software Engineering, and Extreme Programming*. Springer International Publishing, 2016, pp. 39–52.
- [15] B. Kitchenham, "Procedures for performing systematic reviews," Department of Computer Science, Keele University, UK, Keele University. Technical Report TR/SE-0401, 2004.
- [16] U. Eklund and J. Bosch, "Architecture for large-scale innovation experiment systems," in *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*. IEEE, aug 2012.
- [17] X. Amatriain, "Beyond data: from user information to business value through personalized recommendations and consumer science," *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013.
- [18] F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch, "The RIGHT model for continuous experimentation," *Journal of Systems and Software*, vol. 123, pp. 292–305, jan 2017.
- [19] D. K. Vasthimal, P. K. Srirama, and A. K. Akkinapalli, "Scalable data reporting platform for a/b tests," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE, may 2019.
- [20] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, dec 2008.