Thesis for The Degree of Doctor of Philosophy

On Experimentation in Software-Intensive Systems

David Issa Mattos

Division of Interaction Design and Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden, 2021

On Experimentation in Software-Intensive Systems

David Issa Mattos

ISBN 978-91-7905-465-6
Doktorsavhandlingar vid Chalmers tekniska högskola.
Ny serie nr 4932.
ISSN 0346-718X

Technical Report No 195D.
Department of Computer Science & Engineering
Division of Interaction Design and Software Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden

Printed by Chalmers Reproservice,
Gothenburg, Sweden 2021.

"For a human being, nothing comes naturally.
We have to learn everything we do."
- P.P.

Abstract

Context: Delivering software that has value to customers is a primary concern of every software company. Prevalent in web-facing companies, controlled experiments are used to validate and deliver value in incremental deployments. At the same that web-facing companies are aiming to automate and reduce the cost of each experiment iteration, embedded systems companies are starting to adopt experimentation practices and leverage their activities on the automation developments made in the online domain.
Objective: This thesis has two main objectives. The first objective is to analyze how software companies can run and optimize their systems through automated experiments. This objective is investigated from the perspectives of the software architecture, the algorithms for the experiment execution and the experimentation process. The second objective is to analyze how non web-facing companies can adopt experimentation as part of their development process to validate and deliver value to their customers continuously. This objective is investigated from the perspectives of the software development process and focuses on the experimentation aspects that are distinct from web-facing companies.
Method: To achieve these objectives, we conducted research in close collaboration with industry and used a combination of different empirical research methods: case studies, literature reviews, simulations, and empirical evaluations.
Results: This thesis provides six main results. First, it proposes an architecture framework for automated experimentation that can be used with different types of experimental designs in both embedded systems and web-facing systems. Second, it proposes a new experimentation process to capture the details of a trustworthy experimentation process that can be used as the basis for an automated experimentation process. Third, it identifies the restrictions and pitfalls of different multi-armed bandit algorithms for automating experiments in industry. This thesis also proposes a set of guidelines to help practitioners select a technique that minimizes the occurrence of these pitfalls. Fourth, it proposes statistical models to analyze optimization algorithms that can be used in automated experimentation. Fifth, it identifies the key challenges faced by embedded systems companies when adopting controlled experimentation, and we propose a set of strategies to address these challenges. Sixth, it identifies experimentation techniques and proposes a new continuous experimentation model for mission-critical and business-to-business.
Conclusion: The results presented in this thesis indicate that the trustworthiness in the experimentation process and the selection of algorithms still need to be addressed before automated experimentation can be used at scale in industry. The embedded systems industry faces challenges in adopting experimentation as part of its development process. In part, this is due to the low number of users and devices that can be used in experiments and the diversity of the required experimental designs for each new situation. This limitation increases both the complexity of the experimentation process and the number of techniques used to address this constraint.
Keywords Experimentation, Embedded Systems, Multi-armed bandits, Automated experimentation, Optimization, Experimentation process.

Acknowledgments

First, I would like to express my sincere gratitude to my supervisors. I would like to thank my supervisor, Prof. Jan Bosch, for the patience in guiding me through this research while giving me the freedom to explore the topics that interest me. I would also like to thank my co-supervisor, Prof. Helena Holmström Olsson, for the invaluable support and feedback on my research. I could not have asked for better mentors.

I would like to thank all the companies I have worked with. Working with them was a great motivator to pursue this research and explore new topics. In particular, I would like to thank Anas Dakkak for the fruitful discussions and collaboration. I would also like to thank all my co-authors. I have learned a lot about specific topics and research in general from working with each one of you: Jan Bosch, Helena H. Olsson, Anas Dakkak, Krister Bergh, Pavel Dmitriev, Aleksander Fabijan, Erling Mårtensson, Robin Sveningson, Francisco G. O. Neto, Jennifer Horkoff, Richard Svensson, Alessia Knauss, Nikos Diamantopoulos, Jeffrey Wong, Ilias Gerostathopoulos, Matthew Wardrop, Tobias Mao, Colin McFarland, Aiswarya Munappy, Aita Korshani, Jonn Lantz, Teodor Fredriksson, Érika M. S. Ramos, Cecilia J. Bergstad, Yuchu Liu, Lucas Ruud, and Hongyi Zhang.

Next, I would like to thank all at the Interaction Design and Software Engineering Division for making it a great work. I would like to thank especially Terese Besker and Magnus Ågren for the discussions and friendship.

I am grateful to my parents and my grandfather for all the encouragement to pursue this dream.

I would like to express my deepest gratitude to my life companion Érika, for your love, care and patience. Finally, I would like to thank William for bringing me such joy during this year and a half.

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems, and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation and the Software Center.

List of Publications

Included publications

This thesis includes the following publications:

[A] **Mattos, D. I.**, Bosch, J. and Olsson, H. H. “Your system gets better every day you use it: towards automated continuous experimentation” *43^{rd} Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2017, pp.256-265.*

[B] **Mattos, D. I.**, Bosch, J., Olsson, H. H., Dakkak, A. and Bergh, K. “Automated Optimization of Software Parameters in a Long Term Evolution Radio Base Station” *Annual IEEE Systems Conference (SysCon), 2019, pp. 1-8.*

[C] **Mattos, D. I.**, Dmitriev, P., Fabijan, A. Bosch, J. and Olsson, H. H. “An activity and metric model for online controlled experiments” *International Conference on Product-Focused Software Process Improvement (PROFES), 2018, pp.182-198.*

[D] **Mattos, D. I.**, Bosch, J. and Olsson, H. H. “Multi-armed bandits in the wild: Pitfalls and strategies in online experiments” *Information and Software Technology, 2019, v.113, pp.68-81.*

[E] **Mattos, D. I.**, Bosch, J. and Olsson, H. H. “Statistical Models for the Analysis of Optimization Algorithms with Benchmark Functions” *IEEE Transactions on Evolutionary Computation, 2021.*

[F] **Mattos, D. I.**, Bosch, J. and Olsson, H. H. “Challenges and Strategies for Undertaking Continuous Experimentation to Embedded Systems: Industry and Research Perspectives” *19^{th} International Conference on Agile Software Development (XP), 2018, pp.277-292.*

[G] **Mattos, D. I.**, Dakkak, A., Bosch, J. and Olsson, H. H. “The HURRIER Process for Experimentation in Business-to-Business Mission-Critical Systems” *In submission to the Journal of Software: Evolution and Process, 2020.*

Other publications

The following publications were published during my PhD studies, or are currently in submission/under revision but are not included in this thesis.

[a] **Mattos, D. I.**, Bosch, J. and Olsson, H. H. "More for less: automated experimentation in software-intensive systems"
International Conference on Product-Focused Software Process Improvement (PROFES), 2017, pp.146-161.

[b] **Mattos, D. I.**, Mårtensson, E., Bosch, J. and Olsson, H. H. "Optimization Experiments in the Continuous Space"
International Symposium on Search Based Software Engineering (SSBSE), 2018, pp.264-279.

[c] Sveningson, R., **Mattos, D. I.** and Bosch, J. "Continuous experimentation for software organizations with low control of roadmap and a large distance to users: an exploratory case study"
International Conference on Product-Focused Software Process Improvement (PROFES), 2019, pp.528-544.

[d] **Mattos, D. I.**, Bosch, J. and Olsson, H. H. "Leveraging Business Transformation with Machine Learning Experiments"
International Conference on Software Business (ICSOB), 2019, pp.183-191.

[e] **Mattos, D. I.**, Bosch, J. and Olsson, H. H. "ACE: Easy Deployment of Field Optimization Experiments"
European Conference on Software Architecture (ECSA), 2019, pp.264-279.

[f] Oliveira Neto, F. G, Horkoff, J., Svensson, R., **Mattos, D. I.** and Knauss, A. "Evaluating the Effects of Different Requirements Representations on Writing Test Cases"
International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), 2020, pp.257-274.

[g] Diamantopoulos, N., Wong, J. **Mattos, D. I.**, Gerostathopoulos, I., Wardrop, M., Mao, T. and McFarland, C. "Engineering for a Science-Centric Experimentation Platform"
Proceedings of the ACM/IEEE 42^{nd} International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2020, pp.191-200.

[h] **Mattos, D. I.**, Dakkak, A., Bosch, J. and Olsson, H. H. "Experimentation for Business-to-Business Mission-Critical Systems: A Case Study"
Proceedings of the International Conference on Software and System Processes (ICSSP), 2020, pp.95-104.

[i] Munappy, A. R., **Mattos, D. I.**, Dakkak, A., Bosch, J. and Olsson, H. H. "From Ad-Hoc Data Analytics to DataOps"
Proceedings of the International Conference on Software and System Processes (ICSSP), 2020, pp.165-174.

[j] **Mattos, D. I.**, Bosch, J., Olsson, H. H. Korshani, A. M. and Lantz, J. "Automotive A/B Testing: Challenges and Lessons Learned from Practice"
46^{th} Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2020, pp.101-109.

[k] Fredriksson, T., **Mattos, D. I.**, Bosch, J. and Olsson, H. H. "Data Labelling: An Empirical Investigation into Industrial Challenges and Mitigation Strategies"
International Conference on Product-Focused Software Process Improvement (PROFES), 2020, pp.202-216.

[l] Fredriksson, T., **Mattos, D. I.**, Bosch, J. and Olsson, H. H. "An Empirical Evaluation of Algorithms for Data Labeling"
To appear at the 45^{th} IEEE Computer Society Signature Conference on Computers, Software and Applications (COMPSAC), 2021.

[m] Dakkak, A., **Mattos, D. I.** and Bosch, J. "Perceived benefits of Continuous Deployment in Software-Intensive Embedded Systems"
To appear at the 45^{th} IEEE Computer Society Signature Conference on Computers, Software and Applications (COMPSAC), 2021.

[n] **Mattos, D. I.** and Ramos, E. M. S. "Bayesian Paired-Comparison with the bpcs Package"
In submission to the Journal of Behavior Research Methods, 2021.

[o] Ramos, E. M. S., **Mattos, D. I.** and Bergstad, C. J. "Roundtrip, free-floating and peer-to-peer: A Bayesian behavioral analysis of carsharing"
In submission to the Sustainability, 2021.

[p] Liu, Y., **Mattos, D. I.**, Bosch, J., Olsson, H. H. and Lantz, J. "Size matters? Or not: A/B testing with limited sample in automotive embedded software".
To appear at the 2021 47^{th} Euromicro Conference on Software Engineering and Advanced Applications (SEAA).

[q] Fredriksson, T., **Mattos, D. I.**, Bosch, J. and Olsson, H. H. "Machine Learning Algorithms for Labeling: Where and How They are Used?"
In submission to a conference, 2021.

[r] Fredriksson, T., **Mattos, D. I.**, Bosch, J. and Olsson, H. H. "Assessing the Suitability of Semi-Supervised Learning Datasets with Item Response Theory".
To appear at the 2021 47^{th} Euromicro Conference on Software Engineering and Advanced Applications (SEAA).

[s] **Mattos, D. I.**, Ruud, L., Bosch, J. and Olsson, H. H. "On the Assessment of Benchmark Suites for Algorithm Comparison".
In submission to a journal, 2021.

[t] Dakkak, A., **Mattos, D. I.** and Bosch, J. "Success Factors when Transitioning to Continuous Deployment in Software-Intensive Embedded Systems".

To appear at the 2021 47^{th} Euromicro Conference on Software Engineering and Advanced Applications (SEAA).

[u] Dakkak, A., Zhang, H., **Mattos, D. I.** and Bosch, J. "Dimensions in data collection for embedded software-intensive systems".
In submission to a conference (2021).

[v] Zhang, H., Dakkak, A., **Mattos, D. I.**, Bosch, J. and Olsson, H. H. "Towards Federated Learning: A Case Study in the Telecommunication Domain".
In submission to a conference (2021).

Research Contribution

My contribution to the publications produced in this doctoral research is listed below using the CRediT (Contributor Roles Taxonomy) author statement [1]:

Included publications

- Paper A: Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper B: Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper C: Conceptualization, methodology, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper D: Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper E: Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper F: Conceptualization, methodology, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper G: Conceptualization, methodology, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.

Other publications

- Paper a: Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper b: Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing and, visualization.
- Paper c: Supervision, writing - original draft, writing - review and editing.
- Paper d: Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper e: Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.

- Paper f: Investigation, and writing - review and editing
- Paper g: Conceptualization, methodology, writing - original draft, and writing - reviewing and editing.
- Paper h: Conceptualization, methodology, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper i: Conceptualization, methodology, investigation, writing - review and editing.
- Paper j: Validation, formal analysis, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper k: Writing - original draft, and writing - reviewing and editing.
- Paper l: Conceptualization, methodology, formal analysis, writing - original draft, and writing - reviewing and editing.
- Paper m: Writing - reviewing and editing.
- Paper n: Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper o: Software development, formal analysis, writing - original draft, writing - reviewing and editing, and visualization.
- Paper p: Conceptualization, methodology, writing - original draft, writing - reviewing and editing.
- Paper q: Conceptualization, methodology, formal analysis, writing - original draft, and writing - reviewing and editing.
- Paper r: Conceptualization, methodology, formal analysis, writing - original draft, and writing - reviewing and editing.
- Paper s: Conceptualization, methodology, software development, validation, formal analysis, investigation, data curation, writing - original draft, writing - reviewing and editing, and visualization.
- Paper t: Writing - reviewing and editing.
- Paper u: Conceptualization, methodology, investigation, writing - original draft, writing - reviewing and editing
- Paper v: Conceptualization, methodology, investigation, writing - original draft, writing - reviewing and editing

Contents

Chapter 1

Introduction

Understanding the customer preferences and needs has become of strategic importance for software development companies to survive and grow. Today, software development companies use the collected data to assist their decision-making and gain insights into customer preferences [2,3]. Customer data enables companies to timely adapt their business strategies and gain a competitive edge by delivering better products and services to their customers [4, 5].

From web-facing to embedded systems, software development organizations collect data from multiple sources to help them in their decision-making and development process [6]. Even if technically correct, software functionalities might not be relevant, be implemented in a suboptimal way, fail to deliver value to the customers, or can deteriorate the customer experience [7–9]. To succeed and maintain a competitive advantage, companies need to prioritize the development of software features and functionalities that deliver value and that are relevant to the customers [2, 7, 10].

Previous studies show that, often in the development of systems, the prioritization of a feature is driven by guesswork, previous experiences, beliefs of those involved in the feature selection process, and partial or incomplete knowledge of the customer preferences [2, 7, 11]. This ad-hoc prioritization process might not be aligned with the actual user preferences, behaviors, or even with the company's business goals. This mismatch is leading some software industries to look towards a more systematic way to deliver value in their development approaches [2, 12]. To make more informed decisions, these software companies rely on several qualitative and quantitative customer feedback techniques such as surveys, interviews, participant observations, prototypes, mock-ups, feature usage, product data, and support data to determine and validate a feature value [6].

However, not all of the mentioned customer feedback techniques are adequate for fast-iterative software development, a large and diverse user base, business-to-business applications, or for the development of embedded systems. For example, although interviews and participant observations can generate valuable insights on user behavior, these techniques are expensive, they can take several months, and they are based on a limited number of participants, which means that these methods generate small amounts of data. Even observational studies with a sample of hundreds of people can be biased towards a user segment if the software is used by several thousands of users or the study might not have power enough to detect a small effect size that still has practical significance [13].

Therefore, companies need additional techniques to collect a larger amount of trustworthy and unbiased structured data to help in the development and decision-making of not only larger modifications with a big impact but also incremental changes in smaller functionalities. Companies are investing in the collection and analysis of post-deployment data to improve and optimize their current products, to drive innovation in features, and to generate insights on user preferences [14, 15].

A controlled randomized experiment is an empirical method where an observer can test a hypothesis by manipulating one or more factors on randomized subjects while keeping the other parameters constant and observe effects on the outcome variables [16]. This technique allows the observer to make causal inferences regarding the manipulated factors and the outcome variables. Combining post-deployment data with randomized experiments allows software companies to establish a causal inferences relationship between changes in their software and observed user behavior. Although not the only possible design, A/B testing stands as one of the simplest and key techniques to run experiments in online software systems. A/B testing is widely used in industry and has gained significant research interest in the past decade [2, 17–21].

Web-facing software companies have long verified the benefits of online experiments and utilize it as a standard development practice [2, 18–22]. With the increasing number of simultaneous and overlapping experiments, it is becomes unfeasible for the development organization to grow its size and number of data scientists at the same rate. This situation is leading companies to look for different techniques that can automate the experimentation process and reduce the cost of time for each experimental iteration [18, 23, 24]. In this scenario, machine learning, artificial intelligence and self-adaptation techniques can aid the experiment organization to automate this process and run more experiments at a lower cost [25, 26] and to increase the trustworthiness of the experimentation platform to make the process less prone to human mistakes [27].

At the same time, the embedded systems domain (automotive, telecommunication, and consumer electronics) is still in the early stages of running experiments but they see experimentation and in particular automated experiments as one of the approaches that can increase their competitiveness [28–31].

This thesis analyzes experimentation from these two aspects, automated experimentation, and experimentation in the embedded systems domain. First, we analyze how to support and optimize their software systems through automated experiments. This is investigated through the perspectives of the software architecture, the algorithms for the experiment execution, and the experimentation process. Second, we analyze how embedded systems companies can adopt continuous experimentation to continuously validate and deliver value to their customers. This is investigated from the perspectives of the software development process and focuses on the experimentation aspects that are distinct from web-facing companies.

This thesis was conducted in the context of the WASP (Wallenberg Artificial Intelligence, Autonomous Systems, and Software Program, and in the context of the Software Center. We have focused on conducting research in close collaboration with industry partners and on problems relevant to them. Considering the included and related publications produced during this thesis, we have collaborated with 12 different companies globally. All the included publications originated from discussions with our industrial partners and address specific problems faced by them.

The remainder of this thesis is organized as follows. Chapter 2 contains a background review of the main concepts used throughout this thesis. Chapter 3 presents the objectives of this thesis, the research questions, and an overview of the different research strategies and data analysis methods used in the included publications. Chapter 4 discusses each included providing a summary of its main contributions as well as how they relate to the other publications produced in this doctoral research but not included in this thesis. Chapters 5 to 11 contains the included publications. Chapter 12 discusses the proposed objectives and research questions in light of the included publications. Finally, chapter 13 concludes this thesis and discusses potential research directions for experimentation in software-intensive systems.

Chapter 2

Background

This section introduces the central concepts that are relevant and used throughout this thesis. For topics that are related to one specific paper, we refer to the background and related work section of each individual paper.

2.1 Experiments in software systems

Research and industry have long used the term controlled experiments [17,32] to refer to experimentation in software systems. However, the applied methodology does not control for other parameters and variables that can influence the experiment result, [33, 34] and term randomized experiments (without the word controlled) is more accurate.

Since the experiments are often applied in natural settings (as opposed to contrived settings of a traditionally controlled experiment) a more appropriate term would be *field experiments* [35]. The classification as field experiments also emphasizes better many of the challenges faced by software organizations, which are not seen in properly controlled experiments, such as metric sensitiveness and experiments being sized to samples of hundreds of thousands.

In the subsequent chapters of this thesis, we used the terms *controlled experiments*, *field experiments*, and *randomized experiments* without any distinction, despite all of them being better categorized as randomized field experiments. For other field experimental designs such as *quasi-experiments*, *multi-armed bandits*, or *crossover* designs we refer to them by the appropriated name. The term *experiment* refers to any experimental design (randomized or not).

2.2 Randomized experiments

The two-group design, also called A/B testing, is the simplest randomized experimental design used in software experimentation [2, 16]. In this design, users are randomly assigned to different variants of the product. The control variant represents the current system, without any modifications, and the treatment variant, is the system with a modification X. The modification X can be modifications in existing functionalities of the system (e.g. a different set of parameters for each variation), or the system with a new feature or functionality (e.g. different implementation of a new feature). Both variants are properly instrumented to send data to the research and development organization. The different variations are randomly assigned to different users, and, after a predetermined period of time, the metrics for each variation are statistically compared. If the only consistent difference between the experiments' variants is the modification X, the randomization assumptions hold true, the users of both groups are comparable, the metrics have adequate sensitiveness, and the experiment is properly powered, the research and development organization can establish a causal relationship between the modification in the system and the change in the metrics between the different variations. Kohavi et al. [2] provide an in-depth discussion of common experimental design techniques used in online experiments.

2.3 Other experimental designs

The software industry has also explored a range of other designs such as factorial and fractional designs [2], non-randomized designs such as quasi-experiments [36, 37], where randomization is not applied to the treatments, and multi-armed bandits experiments [38], where the treatments are dynamically allocated based on their past metric performance.

2.3.1 Factorial and fractional experiments

Factorial and fractional experiments are specific types of randomized experiments. In factorial experiments, more than one experimental factor is evaluated at the same time [16]. This design introduces a larger number of experimental groups but it is able to detect an interaction between factors, which can lead to significant learning for the organization [2].

Fractional factorial experiments are a reduction of factorial designs to reduce the number of experimental groups by assuming that higher-order interactions are negligible. This design aims to identify the factors that have a higher impact. To achieve that, this design confounds the main low-order interaction effects (depending on the resolution of the design) with the higher-order interactions [16]. Kohavi et al. [2] considers this design a bad practice due to the common presence of interactions and the impact these have in result.

In online experiments, both factorial and fractional experiments are called multi-variate experiments (MVT).

2.3.2 Quasi-experiments

Quasi-experiments are a specific type of experiment that supports causal and counterfactual inference similarly to randomized control experiments, with the key characteristic that it lacks random assignment [33, 36, 37]. The assignment of variations to the subjects occurs by using a cut-off criterion to divide the groups. The criterion can be based on natural conditions such as demographic data or on other criteria or artificial conditions such as clustering methods based on different characteristics. Since quasi-experiments do not use randomization to minimize selection bias this can decrease internal validity, since additional confounding factors can be introduced during the assignment. However, well-planned transparent designs can minimize internal validity threats. One of the key motivating factors to use quasi-experimental designs compared to randomized designs is in situations where its randomization is impractical or unethical. In the analysis, matching techniques can be used to reduce the variation between the experimental groups [36, 37].

2.3.3 Crossover experiments

Crossover experimental design is a particular type of design that the same subjects receive a series of treatments over time [39]. In this design, each subject serves as its own control since we measure and evaluate within-subjects variance. One of the challenges of crossover designs, is potential the presence of carryover or learning effects is identified. In this case, if not controlled for that the variance can significantly increase (or decrease) invalidating the design, because the subjects change as they are exposed to different treatments. However, if the presence of carryover is known, different groups with different treatment sequences can be created to estimate the carryover effects [39]. One of the advantages of crossover designs is when there is a limited number of subjects that do not present carryover effects.

2.3.4 Multi-armed bandit experiments

A multi-armed bandit experiment is a type of experiment design that that aims to minimize the cumulative regret by allocating fewer users to under-performing variants. As an example, if A is the current system and B is the system with a modification. Initially, both A and B are allocated with 50% of the users. If A is under-performing B, the design shifts the user allocation to B, and be would have more than 50% of the users. This type of design is aimed at minimizing the average number of users that are exposed to worse variations. In Chapter 9, we provide an overview and comparisons of multi-armed bandit designs and controlled experiments.

2.4 Continuous experimentation.

Continuous experimentation has often been referred to as the process of running experiments systematically and continuously in the software organization [32]. However, research does not provide a systematic definition or scope to what constitutes continuous experimentation. Despite the reference to experiments, continuous experimentation has been used to refer to a broad range of other techniques that are not experimental design but rather field evaluations such as canary releases, dark launches, or online optimization.

Yaman et al. [40] defined continuous experimentation as "as an experiment-driven development approach that may reduce such development risks by iteratively testing product and service assumptions that are critical to the success of the software". The definition reinforces the term experiment-driven and the authors refer to "experiment-driven development as a means of testing critical product assumptions in the software development process". However, the proposed definition does not emphasize the use of Design of Experiments (DoE) [16] in the experimentation-driven process.

Schermann et al. [41] state that "Continuous experimentation guides development activities based on data collected on a subset of online users on a new experimental version of the software. It includes practices such as canary releases, gradual rollouts, dark launches, or A/B testing". This statement expands continuous experimentation from the original design of experiments or experiment-driven to the evaluation of experimental software, the evaluation process can be through experiments or not. This change in scope moves experimentation from the methodology aspect to the state

Giaimo et al. [42] refer to continuous experimentation as a way to "systematically deploy and run instrumented software variants during the development phase in order to collect data from the field of application."

In this thesis, we consider continuous experimentation as the definition of "techniques that are used to systematically and continuously evaluate, with an experiment methodology, the software deployed in the field". This definition differs from the others in different aspects. While this definition broadens the scope of experimentation it emphasizes that the techniques should be evaluated with an experimental methodology.

2.5 Experimentation models

Despite the simplicity of the A/B testing experimental design, running trustworthy experiments systematically in a software development organization presents many challenges. These challenges can be grouped from the business, organizational, and from the technical perspective [32]. From the business perspective, an experiment can have multiple metrics that might move in opposite directions. The analysis of these experiments is a challenging process, as they can involve multiple stakeholders. It also requires a clear view on how these metrics connect to the business goals of the company [27, 43, 44]. From the organizational perspective, a vertical organizational structure might not be willing to accept experimentation as part of the development process [2]. From the technical perspective, even companies with a large user base have difficulties in sizing their experiments for high confidence levels and power. Collecting such an amount of data can take between weeks and months [23, 45] and might require specialize statistical models to improve metric sensitiveness [43] or domain-specific modifications in the experimentation process [23].

Challenges in these three perspectives have led researchers to investigate and develop multiple experimentation-driven models. These models are used for both introducing, refining, and scaling experimentation inside companies.

2.5.1 The Build-Measure-Learn model

The Lean Startup methodology [46] proposes an approach for companies to continuously and systematically innovate from a startup perspective. The methodology employs a Build-Measure-Learn cycle to ensure that the software development is aligned with the customer's wishes. One of the key aspects of this Build-Measure-Learn cycle is running scientific experiments to validate customer needs and ensure that the product is aligned with these needs.

The build phase reinforces the use of a minimum viable product to steer the product roadmap's direction in a startup environment. The measure phase emphasizes instrumentation needs in the products to measure users' and systems' behavior. The learn phase uses collected post-deployment data to understand and learn movements in hypothesis metrics. This methodology describes a general experimentation process similar to experiments for learning and innovation.

2.5.2 The ESSSDM model

The Early Stage Software Startup Development Model (ESSSDM) [47] proposes an operational support, based on lean principles, for practitioners to investigate multiple ideas in parallel and scale their decision-making process. The model consists of three steps. The first is the generation, in which the startup (or the existing company) generates ideas to expand their product portfolio. The second is the prioritization of the potential ideas in a backlog. The third is the systematic validation funnel using a Build-Measure-Learn loop similar to the Lean Startup methodology. In this step, multiple ideas can be investigated and validated in parallel. The funnel is divided into four stages: the validate problem, the validate solution, the validate the minimum viable product on a small scale, and the validate the minimum viable product on a large scale. This model supports the use of experiments for learning and innovation in a similar manner as the Build-Measure-Learn model.

2.5.3 The QCD model

The QCD model (Quantitative/qualitative Customer-driven Development) [48] explores the continuous validation of customer value instead of relying on up-front requirement specification. The QCD model treats requirements as hypotheses that need customer feedback for validation at the beginning of the development process. All hypotheses are gathered in a hypotheses backlog, where they are prioritized and selected for evaluation. In the validation cycle, the selected hypothesis is evaluated through both quantitative and qualitative feedback. If the hypothesis is not confirmed through the evaluation techniques, it can be refined in another hypothesis for a future iteration or abandoned. This model provides a higher-level experimentation process abstraction. It considers both qualitative and quantitative data analysis methods.

2.5.4 HYPEX model

The HYPEX (Hypothesis Experiment Data-Driven Development) model [5] is a development for companies aiming to shorten the feedback loop to customers. Instead of spending engineering efforts on large pieces of non-validated functionality, the HYPEX model reinforces the need for an iterative and incremental approach. The model divides the development process into six steps:

1. Generation of a feature backlog.

2. Feature prioritization and gap specification.

3. Implementation of a minimum viable feature (MVF).

4. Analysis and comparison of the actual behavior with the expected one.
5. Generation of hypotheses to explain the actual behavior of the MVF.
6. Deciding if the feature should be abandoned, iterated once more, or if it should be considered completed.

2.5.5 The RIGHT model

The RIGHT (Rapid Iterative value creation Gained through High-frequency Testing) [7,8] is a model for driving experiments in a startup environment. The RIGHT process model uses the Build-Measure-Learn loop to help a startup company to achieve the company's vision through continuous experiments. Hypotheses that implement business strategies are generated and prioritized, minimum viable features or products are implemented and instrumented, and data are collected. The analysis of the collected data helps the decision-making process in a similar manner to the HYPEX model [5], where decisions to continue iterating, abandoning, or moving on to the next cycle are made. The RIGHT model describes a high-level experimentation process that can be used in innovation and learning experiments.

2.6 Automated experiments

While experimentation can be automated in different levels of abstraction, from complete iterations with higher-level hypotheses, hypotheses generation to compilation results, it has been mostly discussed in software development in terms of sequential optimization and in experiment execution.

Bosch and Olsson [26] first proposed the concept of automated experimentation in software systems with aim of letting the system own and control the experiments as opposed to the R&D organization.

Sequential optimization experiments refer to the use of sequential experiments to optimize a particular system behavior. In these experiments, a hypothesis about a feature change is manually made by developers. This hypothesis is commonly a change in parameter values and an optimization algorithm searches for the optimal value for this parameter based on live user response. The simplest optimization algorithm is a grid search with the use of sequential A/B tests or A/B/n tests.

The architecture framework proposed in Chapter 5 provides a general architecture framework for automated experimentation, that is not restricted only to sequential optimization. However, its instantiation as well as most of the work on automated experimentation focus on sequential optimization [29, 30, 49–51].

A different perspective is on automating the deployment and execution of A/B testing. This has been the standard approach taken by online companies to reduce the cost of each experiment iteration. In this perspective, both hypotheses generation (from parameter modifications to large functionalities) and the learning of the experiments are out of the scope of the experimentation system and is conducted mainly by experiment owners and development teams. This perspective on automating the experiment execution process is discussed in chapter 7.

2.6.1 Algorithms for online optimization

A range of algorithms has been proposed and used in both research and industry. While it is not of interest to this thesis to overview all possible algorithms that can be used for sequential optimization experiments, we provide below a brief description of four classes of algorithms that can be used.

In the context of combinatorial or discrete optimization, multi-armed bandit algorithms have been used for optimization combined with regret minimization [38]. These algorithms assume that there is a finite (and often small) number of variations to choose from. Based on the user response, new users are allocated more to best-performing variants.

For optimization of continuous (linear) variables with regret minimization, χ-bandits algorithms can be used [29, 52]. This class of algorithms divides the search space into a tree of possibilities. The growth direction of the tree depends on the user response.

For optimization of continuous variables without considering regret minimization, a range of black-box optimization algorithms can be used. Examples of these algorithms are Bayesian Optimization [53], Evolutionary and Nature-inspired Algorithms [54, 55], the Nelder-Mead [56] among others.

Finally, automated hyperparameter tuning in machine learning has provided a range of algorithms that can be used to run automated experiments with mixed discrete and continuous variables. Examples of these algorithms are the Tree-Parzen Estimator [57], BOHB [58], HPBandster [59], and the SMAC [60]. These algorithms provide more flexibility in the specification of the search space, better performance in the case of a low number of data points at the expense of higher computational time for each iteration which can limit its usage in live settings.

The choice and comparison of these algorithms are often conducted utilizing a group of benchmark functions [61, 62] (benchmark suites) and utilizing frequentist statistical methods [63]. However, these statistical methods are often misused, in particular in the case of evolutionary algorithm comparisons, and do not take into account correlation due to repeated measures, interpretation of effect size, family-wise error correction, verification of the model assumptions or simply utilize non-parametric tests for individual benchmark function ranking. The lack of systematic comparison between these algorithms increases the difficulty to make an informed decision regarding the selection of an appropriate optimization algorithm.

2.7 Experimentation in the embedded systems.

The first research discussing the experiments in embedded systems appeared in 2012 [64]. This paper discusses the possibility of utilizing experimentation to drive innovation in embedded systems and identifies general challenges, such as experimentation in safety systems, managing multiple stakeholders, and hardware limitations. It also presents an initial infra-structure to run experiments in embedded systems with a case study in infotainment systems in the automotive industry.

The case study presented in Chapter 5 and in [65, 66] instantiated an experimentation framework in the Robot Operating System (ROS) in a research mobile autonomous vehicle in a proxemics distance problem.

Giaimo et al. [42] investigated the broader range of continuous experimentation techniques in a systematic literature review. The study concludes that there are more conceptual analysis and challenges identification than proposed solutions. Continuous experimentation has started to gain visibility and be applied in the automotive domain. Giaimo and Berger [67] discuss continuous experimentation in the context of self-driving vehicles. The paper presents functional (such as instrumentation, logging, data feedback to a remote server) and non-functional (separation of concerns, safety, short cycle to deployment) requirements to achieve continuous software evolution. Giaimo et al. [68] investigated the perception of practitioners on the automotive domain on the use of experimentation in the automotive domain. While the perception is positive practitioners see safety and organizational structure as major challenges. Mattos et al. [31], discuss challenges and lessons from the automotive industry when starting to run the first A/B experiments. While some challenges are visible in other domains, such as the number of variants, suppliers, or the low number of users to run, others are specific to the automotive domain and do not generalize to other domains, such as restrictions imposed by the AUTOSAR architecture.

However, experimentation in the automotive industry has only recently started and both practice and research in this domain still does not have sufficient evidence and validated processes for running in other automotive companies or generalization to other embedded systems.

In the context of embedded telecommunication software systems, we have investigated the use of continuous experimentation techniques in collaboration with Ericsson and Sony Mobile [29, 30, 69]. The continuous experimentation perspective in the embedded telecommunication domain is discussed in detail in Chapter 11.

2.8 Experimentation in the B2B domain.

One important aspect of CE in the business-to-business (B2B) domain is the difference between customers and users. Customers acquire or subscribe to a product or service for the users [70, 71]. In the business-to-customer domain, the customers are also the users, and generally acquire or subscribe to the product for themselves. Therefore, in the B2B domain, vendors usually sell products and services to other companies that sell products or services to users. A distinctive factor is that user data, product usage, and user feedback are not readily or easily available for the vendors without prior agreements. This can restrict the data collection, user feedback, and even new deployments aimed at product improvement.

Yaman et al. [40] describe the process of introducing continuous experimentation in companies with an established development process using two company cases with pure software products, Ericsson and a digital business consulting company. The study investigates the introduction of experimentation in a cloud service platform, describing relevant decision points taken (such as the target of the experiment, how to update the experiment design, etc), benefits from the experiment (new insights, reduced development effort, etc) and challenges (access to end-users, inexperience with experimentation, length of the process, etc). Rissanen and Münch [71] investigate challenges, benefits, and organizational aspects when introducing CE in the B2B domain. They identified that customers play a major role when designing and deploying an experiment.

We have further investigated the use of experimentation in the business-to-business domain in Chapter 11.

Chapter 3

Research approach

In this chapter, we present the research objectives of this thesis, the specific research questions for each objective, and the research strategies and methods used in the included publications.

3.1 Objectives

This thesis has two main objectives that are investigated and discussed in detail in the seven included papers (chapters 5 to 11). These objectives are divided in multiple research and sub research questions. The mapping between the research questions and the included publications can be visualized in figure 4.1.

3.1.1 Objective 1

The first objective of this thesis is the analysis of how software companies can support and optimize their systems with automated experiments.

Web-facing companies recurrently report the benefits of conducting experiments as part of their product development [4, 13, 18, 19, 45, 72, 73]. While, in large-scale companies, experimentation has scaled to several thousands of experiments a year most of these experiments are created, developed, and conducted by humans. This thesis investigates how software companies automate part of their experimentation processes as well as how research developments can increase the level of automation in experiments. This thesis study the automated experiments from the perspective of software architectures for automated experimentation on chapters 5 and 6, the algorithms for the experiment execution on chapters 8 and 9 and the experimentation process on chapter 7.

For this objective, the following research questions and sub-questions are discussed in the included publications:

- **RQ1:** What are the characteristics of an architecture for automated experimentation?
  - **RQ1a:** What architectural software qualities support automated experimentation?
  - **RQ1b:** What are the existing software architectures that support these qualities?

- **RQ2:** How can we utilize automated experimentation to optimize an existing software-intensive systems?
- **RQ3:** What are the main components to run trustworthy online controlled experiments?
  - **RQ3a:** What are the set of activities that are conducted in each experiment iteration?
  - **RQ3b:** What is the role and lifecycle of metrics in the evolution of experiments?
- **RQ4:** How are multi-armed bandit (MAB) algorithms used in online field experiments?
  - **RQ4a:** What are the restrictions and pitfalls associated with MAB algorithms applied to software online experiments?
  - **RQ4b:** What are the decisions involved in the design of MAB-based online experiment?
- **RQ5:** How can we improve the conclusion validity on the analysis of optimization algorithms with benchmark functions in different domain specific research questions?

3.1.2 Objective 2

The second objective of this thesis is the analysis of how non web-facing companies can adopt continuous experimentation as part of their development process.

Research in continuous experimentation has mainly focused on driving experimentation in web-facing business-to-customer companies that have high-speed deployment cycles, constant connectivity, and user data collection. However, continuous experimentation can have a significant impact in software-intensive companies developing embedded, telecommunication, mission-critical and business-to-business systems. These companies face many different challenges compared to web-facing companies, including safety-regulated environments, larger development and deployment cycles, non-constant connectivity, higher distance to users in terms of data collection and ownership, service level agreements among others. All these differences impact how experimentation is planned and conducted. This thesis investigates the challenges related to conducting experimentation in these companies on chapter 10 and the different types of experiments, techniques, and processes in business-to-business mission-critical systems on chapter 11.

For this objective, the following research questions and sub-questions are discussed in the included publications:

- **RQ6:** How can the embedded systems industry adopt continuous experimentation in their development process?
  - **RQ6a:** What are the recognized challenges towards continuous experimentation faced by the embedded systems industry?

  - **RQ6b:** What are the recommended strategies to facilitate the use of continuous experimentation in the embedded systems domain?

- **RQ7:** How experimentation can be conducted in mission-critical business-to-business systems?

  - **RQ7a:** What are the types of experiments that can be conducted and that are relevant in mission-critical B2B systems?
  - **RQ7b:** What are the current continuous experimentation practices used in mission-critical B2B systems?
  - **RQ7c:** What processes can be used to drive CE in mission-critical B2B systems?
  - **RQ7d:** What are the current CE challenges and opportunities in mission-critical B2B systems?

To achieve these objectives discussed in the previous section, this thesis utilizes a range of different research methods, such as literature reviews, experimental simulations, case studies, and empirical evaluations in collaboration with multiple companies. In the next sections, we provide an overview of these methods and the collaborations with the industry.

3.2 Research context

This research was conducted in the context of two initiatives, the WASP and the Software Center.

The Wallenberg AI, Autonomous Systems and Sofware Program (WASP) [1] is a research initiative that focus on the development of artificial intelligence and autonomous systems acting in collaboration with humans, adapting to and learning from their environment through sensors information and knowledge, forming intelligent systems-of-systems. Software is seen as the main enabler of these systems. Automated experiments allows software to be optimized through the interaction with humans and the environment. The first objective of this thesis is placed in the WASP context. Both Sony Mobile and Ericsson, which were part of studies in the context of automated experiments and online optimization are affiliated to WASP initiative.

The Software Center [2] is a initiative that runs research projects in active, close and long-term collaboration with industrial and academic partners. Both objectives of this thesis are placed in the context of the Software Center. Many of the publications had collaboration with Software Center companies, such as papers *B*, *D*, *e* for objective 1, and papers *F*, *h*, *G*, *j*, and *p* for objective 2.

3.2.1 Company collaborations

In the context of these two initiatives, we have collaborated with multiple industrial partners. Below, we provide a brief description of each company collaboration that was part of the included publications and their experience and

[1] https://wasp-sweden.org/research/
[2] https://www.software-center.se/about/mission/

relation to experimentation. Since both Ericsson and Microsoft are explicitly mentioned in the included publications (chapters 6, 7 and 11). The other companies, referred as Company I-VIII, remain anonymous as requested by them when conducting the study.

Ericsson Ericsson AB is a multinational networking and telecommunications company that develops, produces, and sells telecommunication equipment, services, software, and infrastructure to telecommunication operators in both mobile and fixed broadband. Ericsson employs over 95,000 people in around 180 countries. Over the last 10 years, Ericsson started the transition from traditional development to agile and towards DevOps. In the last 5 years, CE started to get attention and promotion inside Ericsson, and although continuous experimentation is not a well-defined process throughout the company, several teams independently conduct over a thousand field experiments a year, in different products and parts of the system. Experiments in Ericsson are used in a large number of use cases ranging from innovation and new feature development to legacy assurance and performance optimization. We have collaborated with multiple teams, areas, and products spread over seven locations in five countries.

Microsoft Microsoft Corporation is a multinational technology company that develops, manufactures, licenses supports, and sells computer software, personal computers, consumer electronics, and services. The Analysis and Experimentation group at Microsoft is one of the leading groups in online experiments running over 20 000 experiments a year [45] in multiple types of systems such as web, personal computer, mobile, embedded systems, and cloud infrastructure.

Company I Company I is a travel fare aggregator and travel engine provider. It develops booking and travel solutions used by both individuals and the travel industry. A/B testing methodologies are an integral part of the development process of the company.

Company II Company II is a multinational company that provides telecommunication and networking systems. The company is adopting continuous development practices and is looking for new strategies to deliver more value to their customers by optimizing their products.

Company III Company III is a global automotive manufacturer and supplier of transport solutions. As the company's products are continuously growing in complexity and software size, the company is looking for strategies to prioritize its R&D effort and deliver more value to its customers. As many employees have experience in web software development, experimentation is getting traction in some development teams.

Company IV Company IV is a global software company that develops and provides embedded systems software solutions related to autonomous driving technology for the automotive industry. Autonomous driving is an emerging and fast-moving technology and the company is looking to deliver competitive solutions faster by adopting continuous development practices.

Company V Company V is a global software company that develops both software and hardware solutions for home consumers. The company already has experience running continuous experimentation in their web systems and is starting to run experiments in their hardware solutions.

Company VI Company VI is a multinational company that manufactures embedded systems and consumer electronics. In recent years, the company started to adopt experimentation in their software solutions and is looking for data-driven strategies in their embedded systems products.

Company VII Company VII is a company that develops experimentation solutions for its customers. The company offers A/B/n, MVT, and other experimentation tools for websites along with frameworks for experimentation in mobile platforms. The company developed its own statistics engine and offers solutions using MAB algorithms to customers. The company's customers include several multinational companies from different domains, from entertainment to large news agencies.

Company VIII Company VIII is a software company focused on website optimization and offering experimentation tools and solutions for A/B testing and MABs. The company's customers include several multinational companies in North America, Asia, and Europe.

3.3 Research strategies

In this section, we provide an overview of the research strategies and the research methods utilized in the appended publications of this thesis. We utilize the ABC classification framework provided by Stol and Fitzgerald [35] to describe each research strategy and additional publications for each research method.

3.3.1 Field study

Field studies refer to any research conducted in a real-world setting [35]. In this kind of study, researchers do not actively control or change any parameters or variables of the context. The main goal is to understand the phenomena of interest in a concrete and realistic scenario at the expense of a lower precision of the measurements and lower generalizability of the findings, reducing both the internal and external validity.

3.3.1.1 Case studies

An exploratory case study is a common research field study strategy in software engineering [74–77]. While there are many definitions of what constitutes a case study [78], we utilize the broader and widely used definition of Runeson and

Höst [77]: *"Case study in software engineering is an empirical inquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between the phenomenon and context cannot be clearly specified"* . In contrast with other types of research, this definition emphasizes the need for a real-life context and that of a contemporary phenomenon [78, 79]. In this thesis, we conduct case studies following the five major process steps guidelines [77, 79].

The first step consists of the case study design. In this step, the objectives are defined in terms of the object of study, the context, theoretical frame of reference, research questions, methods to collect the data, and selection strategy for the data collection [80].

The second step consists of the preparation of the procedures and protocols for data collection. The protocols refer to the field procedure to guide the data collection with the aim of preventing the researchers to miss collecting data that was planned or not attaining ethical considerations (such as informed consent, handling sensitive results, etc.). The protocol is a dynamic procedure that is updated when the plans of the study change.

The third step consists of collecting data from multiple sources. These sources can be of three different degrees [81]: The first degree consists of data collected from direct contact with the subjects or objects of study (such as interviews). The second degree consists of data collected indirectly from observations without interaction. The third degree consists of data from the analysis of artifacts that were already produced and compiled (such documents). In this thesis, we mainly collected first (interviews) and third degree (documentation) data. Due to the exploratory and explanatory nature of our interview-based case studies, these were designed to be semi-structured with open questions [80]. The number of interviews was decided based on saturation (when no new information or viewpoint is gained) and availability of subjects.

The fourth step consists of the analysis of the collected data. While in case studies we can conduct both qualitative and quantitative data analysis, the case studies presented in this thesis focus only on qualitative data analysis based on thematic coding [82]. We describe the thematic process used in more detail in section 3.4.1.

Finally, the fifth step consists of reporting the results. This thesis reports the results of four case studies in chapters 7, 8, 10 and 11.

While the described process is similar to other empirical studies, a case study allows a flexible approach with multiple iterations over the steps give the constraints of the specific objectives and protocols of the study [77].

3.3.2 Sample studies

A sample study is a strategy that aims to achieve the generalizability of the findings [35]. This kind of research is unobtrusive and does not manipulate any variables. While there are many types of sample studies (e.g. data mining, surveys, literature reviews) we focus, on this thesis, on the literature review.

3.3.2.1 Literature review

A literature review is a type of secondary study that aims to identify evidence concerning a specific technology, current gaps and suggest areas of investigation, and provide a framework for positioning new research activities [83].

In this thesis, we have performed literature reviews to identify existing software architectures and architectural qualities for an automated experimentation system in chapter 5 and to identify existing challenges and solutions for conducting experiments in the embedded systems in chapter 10.

We have performed these literature reviews in accordance with the procedures summarized by Kitchenham [83], to minimize publication bias and increase the completeness of the search.

3.3.3 Laboratory experiments

A laboratory experiment is a research strategy that aims to observe and measure the behavior of a sample of a population (e.g. algorithms or software systems) in an artificial setting recreated to represent a concrete application or set of applications [35]. In a laboratory experiment, we have high control of the measurements and of external factors at the expense of the realism of the application. The artificial setting differentiates this research strategy from the commonly used field experiments in A/B testing.

Chapter 5 conducts a laboratory experiment evaluation of a mobile robot. The environment, as well as the task, was controlled without randomization of the experimental subjects to the different variation instances as it was an online optimization application. Similarly, chapter 6 also utilizes a laboratory experiment without randomization for the online optimization application of a radio base station.

3.3.3.1 Controlled experiments

A controlled experiment is a vital part of the scientific and engineering method. This method consists of elucidating information about why and how a system works based on observation measurements. The process of understanding how and why of a system is supported by the creation of empirical models [16]. In an experiment, we have one or more input variables (the changes we are applying, also known as independent variables), response variables (the observation of interest, also known as dependent variables), controllable and uncontrollable variables that can influence the response variable.

As discussed in the chapter 2, if the subjects are randomized into the experimental groups we call it randomized controlled experiments.

In controlled experiments, we can fix the values of controllable variables in order to minimize the effects of uncontrollable variables in the response. We have utilized controlled experiments in Chapter 8 to observe the differences between A/B testing and multi-armed bandits under different circumstances.

3.3.3.2 Benchmarking studies

Benchmarking studies are another type of laboratory experiment where researchers set up a contrived environment to analyze and measure the difference between different techniques, such as algorithms. The environment is controlled and consists of a number of benchmarks where the multiple techniques are evaluated and their behavior is measured.

In the specific context of benchmarking for the comparison of optimization algorithms, researchers have discussed many aspects of what consists of good benchmarks. The survey by Bartz-Beielstein et al. [63] provides an extensive survey that discusses different topics for promoting good benchmark practices, from objective statement, selection, and characteristics of benchmarks, to the analysis and presentation of results. However, from the analysis perspective they focus solely on the usage of frequentist statistics and null hypothesis testing, while their well-known limitations and pitfalls of frequentist statistics are not considered and alternative analyses such as Bayesian Data Analysis (BDA) and item response theory are not mentioned.

Chapter 9 addresses the specific conclusion validity problem that is common in the analysis of optimization algorithms utilizing benchmark studies.

3.3.4 Summary

Table 3.1 shows an overview of the research strategies and methods used in each appended paper.

Paper	Research strategy	Comments
A	Sample study and laboratory experiment	Literature review combined with a laboratory experiment evaluation with a mobile robot.
B	Laboratory experiment	Experimental evaluation in a controlled environment testbed with Ericsson.
C	Field study	Interview-based case study with Microsof.t
D	Field study and laboratory experiments	Interview-based case study with 5 companies and simulation-based controlled experiment.
E	Laboratory experiments	Benchmark simulations to illustrate the proposed statistical models.
F	Sample study and field study	Literature review and a multiple case study with 5 companies.
G	Field study	Interview-based study with Ericsson.

Table 3.1: Summary of the research strategy used in the appended publications.

3.4 Data analysis

In this thesis, we have used primarily two data analysis methods, thematic coding, and statistical analysis. Below, we describe the theoretical foundations of these methods and how they were used in each appended publication.

3.4.1 Thematic coding

The data collected from our case studies, such as including interview transcripts, documentation, meeting notes, workshop material, etc, were analyzed utilizing the thematic coding process described by Braun and Clarke [82]. Thematic analysis is a qualitative research method for identifying, analyzing, and reporting patterns observed in the collected data. This process can be taken further to interpret different aspects of a research topic and provide a rich set of details. Different from other qualitative thematic decomposition analyses, such as grounded theory and interpretative phenomenological analysis (IPA) which are theoretically bounded, thematic analysis is not research methodology itself but a data analysis procedure. And as in such, thematic analysis allows themes to emerge [82].

It is worth noting that within thematic analysis the themes are not a property of the data but they emerge from the links and understanding we make from them [84].

Thematic analysis can be divided into inductive and theoretical thematic analyses [82]. An inductive thematic analysis allows themes to be strongly linked to the data without a pre-existing coding frame or analytic pre-conceptions. Theoretical thematic analysis tends to be driven by a specific theoretical area, construct, or coding frame.

Braun and Clarke [82] present a six-phase process. The first phase consists of familiarizing the data. This is done by the researchers in several ways, such as participating in the interviews, transcription, reading, initial ideas, etc. In the second phase, the researchers generate the initial codes by highlighting interesting features of the data in a systematic way. In the third phase, researchers search for potential themes from the identified codes in phase two. In the fourth phase, the researchers review the potential themes in relation to the extracted codes and with the entire data. This phase generates a thematic map of the analysis. In the fifth phase, the researchers define and name the themes in relation to the overall story of the analysis and contextualizing it with existing research or theoretical constructs. The last phase consists of producing a report including a discussion of the proposed research questions that motivated the research and selection of compelling data extracts as examples of the collected data.

Chapter 5 has conducted an inductive thematic analysis to find the relevant architectural qualities in the data collected from the literature review. Chapter 7 has conducted an inductive thematic analysis from interviews, diagrams of the process, and the platform architecture to identify the presented findings and the proposed experimentation framework. Chapter 8 has conducted an inductive thematic analysis of the interviews to identify and group the presented restrictions, pitfalls, and strategies. Chapter 10 has conducted a theoretical thematic analysis on the interviews, workshop material, and literature review for categorizing the observed challenges and an inductive thematic analysis to categorize the strategies. Finally, chapter 11 has utilized an inductive thematic analysis on the interviews and in the additional collected data to derive the HURRIER.

3.4.2 Statistical analysis

Statistical analysis is the process of analyzing, summarizing, interpreting, and presenting the collected quantitative data. While statistical analysis can be used for describing tendencies and summarizing the data, in this thesis we focus on inferential statistical analysis, which aims to draw conclusions from the data considering observational or measurement errors and sampling variation. Statistical inference relates to creating statistical models to draw inferences on the population from data collected from a sample.

In this thesis, we utilize the two paradigms of statistical inference, frequentist, and Bayesian that are explained next.

3.4.2.1 The frequentist paradigm

The frequentist paradigm refers to obtaining point estimation for statistical model parameters under the frequency view of probability. Under this paradigm, parameters and hypotheses are seen as unknown fixed quantities that we want to estimate. For inference, frequentist statistics rely on procedures such as hypothetical repeated sampling of the data (frequency view) [85]. Frequentist statistics is the standard approach for evaluating experimental results in online experimentation, where big data and a large number of metrics and hypotheses are conducted simultaneously. Frequentist estimation is usually based on the maximum likelihood estimator (MLE) or in variations of it, such as the quasi or penalized maximum likelihood estimator.

Unfortunately, frequentist methods for null hypothesis testing have often been misused by scientists and practitioners looking for a dichotomy tool to assess a particular problem without evaluating the size of the observed effect or with a discussion with complementary analysis [86]. By utilizing statistical tests as black-box tools, many pitfalls and misuses have been observed in different fields of science. We list some of the observed pitfalls.

- [a] Lack of separation between the effect size and sample size in the p-value [87].
- [b] Lack of information regarding the null hypothesis [87–89].
- [c] Misinterpretation of the actual meaning of the p-value (including by instructors in statistics) [89–91].
- [d] Misinterpretation of the meaning of confidence intervals [92, 93];
- [e] Lack of transparency in the reporting of the statistical procedures (such as providing the value of test statistics, the actual value of the p-value, confidence intervals) [92];
- [f] Common problems related to the misuse of the statistical tests such as not verifying the statistical test assumptions, not controlling for correlated samples, not controlling for family-wise errors in multiple group comparisons [94].

In this context, the Bayesian paradigm has gained attention from researchers and practitioners as it naturally solves many of the problems listed above.

Chapter 8 analyzes the results of the simulation experiments with multi-armed bandits and A/B testing utilizing the frequentist paradigm since this is the most common paradigm in A/B testing.

3.4.2.2 The Bayesian paradigm

The Bayesian paradigm, also called Bayesian Data Analysis, treats all unknown quantities in the statistical model as random variables, contrasting with the fixed constants from the frequentist approach. With the use of appropriated conjugated priors, Bayesian inference can lead to analytical and tractable solutions for the posterior distribution of the parameters. However, most practical problems require a Markov Chain Monte Carlo (MCMC) sampler to find numerical solutions for the posterior distribution of the parameters.

The main idea behind Bayesian data analysis is the reallocation of credibility across possibilities [88]. In practical terms, we start with a prior explanation of the results before seeing any data and a model on how the data is generated. As we collect new data, our beliefs about the system are reallocated. The probability of candidate explanations that do not fit the data well is therefore reduced. In this updating process, we get a probability distribution of each possible explanation of the data. This allows us to obtain the credible (or uncertain) intervals [93, 95].

The process of allocating explanations into probability distributions happens through the principles of conditional probabilities and the Bayes theorem:

$$\mathcal{P}(h|d) = \frac{\mathcal{P}(d|h) \cdot \mathcal{P}(h)}{\mathcal{P}(d)}, \tag{3.1}$$

where d represents the data, h the explanation (or hypothesis), $\mathcal{P}(h|d)$ is the conditional probability of the hypothesis given the observed data. Below are common names for the factors in the Bayes theorem:

- $\mathcal{P}(d|h)$ is called the likelihood of the data d under the hypothesis h.
- $\mathcal{P}(h)$ is called the prior.
- $\mathcal{P}(h|d)$ is called the posterior. The posterior represents the probability distribution of each parameter estimate (our hypothesis h) given our observed data
- $\mathcal{P}(d)$ is called the marginal likelihood and it is a constant, that is often impossible to compute analytically.

In chapter 9, we provide different Bayesian statistical models that can be used to evaluate optimization algorithms, which are used in online optimization problems. These models are used to answer different research questions and incorporate random-effect terms to model the repeated measures that are common in benchmarking applications [96].

3.5 Validity considerations

In this section, we present the main validity threats and considerations of the results discussed in this thesis.

3.5.1 Construct validity

Construct validity refers to what extent the operational measures represent what is under investigation according to the research questions [77]. In case studies, a threat to the construct validity could be if the interview questions (or concepts) are not interpreted in the same way by the researchers and the interviewees.

Another definition in closer context to experimentation is: "(construct validity) involves accepting a set of operations as an adequate definition of whatever is to be measured" [97]. In online field experiments, a construct validity threat is the misuse of a non-validated metric (such as click-through rate) as a latent measurement to a more complex construct (such as customer satisfaction).

In our presented case studies in chapters 7, 8, 10 and 11, we specifically address threats to construct validity by selecting interviewees and companies relevant to the research questions. In those cases, if any concept was not properly understood or if the interviewee's answer indicated misunderstanding, these concepts were explained and given examples to illustrate.

The laboratory experiments did not present construct threats as the interpretation of the measurements was directly connected to the definition. None of the measurements were based on latent variables, that required further validation.

3.5.2 Internal validity

Internal validity refers to whether the unaccounted factors could impact the results of the investigated factors when causal relations are examined [77, 98].

In the context of experiments, Campbell et al. [98] describe internal validity as the minimum requirement of which for which the results of any experiment are interpretable. Campbell et al. also point eight primary classes of variables that can produce confounded effects if not controlled in the design. These variables are:

[a] History of the events. The order of the events between measurements and the introduction of the experimental variable matters.

[b] Maturation process. The experimental subjects can change over time (e.g. getting tired).

[c] Testing. Taking the effects of a test on the scores of a second testing. E.g. a pretest can influence the result of the test.

[d] Instrumentation. Instruments should be calibrated to measure the produced changes in the measurement.

[e] Statistical regression. If the subjects (one or more) have been selected from an extreme measurement group, it is more likely that an effect will be observed.

[f] Selection of respondents. A bias in the selection process can lead to non-comparable groups before introducing the treatment.

[g] Experimentation mortality. A differential loss of respondents (not at random) impacts the validity. E.g. a treatment leads more people to give up on the experiment.

[h] Selection-maturation interaction. The selection criteria can interact with other variables and be mistaken with the effect of the treatment. E.g. the Simpson paradox discussed in chapter 8.

In the context of the laboratory experiments discussed in chapters 5, 6, 8 and 9 we have addressed every item of this list. All the measurements occur after the introduction of the treatments. Maturation and testing are not considered as possible confounding variables since we are investigating computational artifacts as subjects. The instruments are based on previous research and data collections were validated in simulations prior to the experiments. Statistical regression, this effect was minimized with repeated measures in benchmarking (chapter 9) and the experimental evaluations (chapters 5 and 6), and with larger samples in the multi-armed bandit simulations (chapter 8). The selection of the respondents does not apply to the laboratory experiments from chapter 5 and 6 as they are single respondent repeated measures, for the simulations and the benchmarks the respondents (algorithms) were selected based on previous research, and potential impacts of different seeds were also addressed with repeated measures. Mortality and selection interaction do not apply to our experiments as we are dealing with computational artifacts and simulations.

However, while internal validity has been a common priority in research [99], it is not always possible to provide high level of internal validity in applied research. Victoria et al. [100] argues that while randomized controlled experiments and internal validity are essential for evaluating the efficiency of treatments and interventions, the complex causal chains in large organizations (and in public health), prevent simple results to be safely extrapolated to other settings. In these cases, non-causal research and plausibility designs might be the only alternatives to investigate the potential impact of interventions.

In this thesis, the conducted case studies in chapters 7, 8 and 10. Given the complex context of the organizations of these case studies and their exploratory natures, they can be potentially affected by internal validity threats since they cannot establish causal relationships.

3.5.3 External validity.

External validity refers to what extent the results of the study can be generalized to other situations outside of the investigated case [77, 99, 101]. In applied research, external validity and generalization of the results are emphasized and strengthened [100, 101].

This thesis considers the external validity from two aspects. First, results can achieve higher generalization if artifacts prove to have effectiveness in less controlled environments in comparison with the proved efficacy (high internal validity) in controlled environments. An example of this is the software architecture produced in a laboratory environment in chapter 5 being used in a less controlled environment in a company context in chapter 6.

Second, external validity can be achieved if artifacts developed or identified in a particular context (e.g. in one organization) can also be used and identified in additional organizations. In the context of the conducted case studies, we do not have a representative sample to generalize the findings and the results are specific to the context of the case study (e.g. the company in that specific point in time). However, the results are intended to enable to generalize to other cases that have common characteristics [77]. In chapters 8 and 10, we aim to increase the external validity of identified challenges, restriction and potential solutions by increasing the number of case companies and triangulating the results with existing literature and simulations. However, in chapters 7 and 11, the artifacts produced are specific to the case companies and cannot be generalized. Nevertheless, we aim at providing maximum context so additional cases that have common characteristics with the ones we provide can benefit from some level of generalization.

3.5.4 Conclusion validity

Conclusion validity, or statistical conclusion validity, refers to the particular reasons, methods, and procedures we use to draw conclusions about a possible covariation between variables [33, 102]. Statistical validity requires a close examination of the statistical procedures and assumptions used. In contrast with construct validity, a correctly statistical conclusion can be drawn on the wrong construct. In addition, conclusion validity does not assess if the covariations obtained are causal (internal validity) and can be generalized to new settings, persons, and treatments (external validity) [33].

In the laboratory experiments from chapters 5, 6, 8 and 9, we have carefully investigated if the statistical conclusions we have made are in agreement with the assumptions and procedure taken. The algorithm used in chapter 6, was developed and empirically validated in [29]. In 8, we address some of the conclusion validity threats observed by practitioners when adopting MAB algorithms. In these situations, we conducted simulations to illustrate each case and the statistical analysis presented follow the required assumptions of each statistical test. In chapter 9, we present a number of statistical models to address the problems in the statistical conclusion observed in optimization algorithms development. These models are applied in benchmarking data and take into account known assumptions (such as repeated measures and correlation between benchmark functions). All the statistical procedures are reproducible in an online appendix.

We do not discuss the statistical conclusion validity of the case studies from chapter 7, 8, 10 and 11 since they do not perform statistical analysis. However, the conclusions presented in those chapters are in the context of the thematic coding analysis method presented earlier. While it is possible to assess the procedures conducted in thematic analysis, the themes and results obtained are not a property of the data but they emerge from the links and understanding we make from them and cannot be separated from the researchers [82, 84].

Chapter 4

Contributions of this thesis

This chapter describes how the included and the related publications are connected and contribute to a broader understanding of experimentation. First, we provide a general overview of the research projects conducted in this doctoral study (that are connected to the objectives of this thesis). Second, we provide a summary of the study, the research method, and the main results of each included publication. Finally, we provide a summary of the related publications that are not included in this thesis.

4.1 General overview

This thesis started with projects related to each of the objectives of this thesis, paper *A* and paper *F*. We describe below how each publication connects in each objective and related themes.

Objective 1 The first objective was initially investigated from the academic perspective with literature reviews and evaluations in contrived scenarios (papers *A* and *a*). These first results led us to a case study in collaboration with Sony Mobile in paper **b** and with Microsoft in paper *C*. In paper *b*, we investigated the use of different algorithms to drive online optimization and proposed our own χ-armed bandit algorithm (the LG-HOO). The initial results from our work with Sony Mobile led to a new collaboration with Ericsson (paper *B*) and the initial design of study *D*.

During the data collection of paper *C*, informal discussions also indicated that while practitioners often saw multi-armed bandit as a logical step towards automating experiments, its use was often associated with pitfalls. Based on the experiences in studies *b* and *C*, we designed a multiple case study with multi-armed bandit experts.

Iterations over the ACE architecture framework (discussed in paper *A*) and its specialization towards online optimization in collaboration with Sony Mobile and Ericsson led to the design of the architecture discussed in paper *e*. This architecture contained several optimization algorithms such as Bayesian optimization, multi-armed bandits, and evolutionary algorithms. While these algorithms were often designed, evaluated, and reviewed in their own communities, they lacked a common empirical statistical evaluation process. This led us to conduct the study presented in paper *E*. In this paper, we provided an overview of statistical models that can help researchers evaluate research questions beyond statistical significance. We take this evaluation a step further in paper *s*, where we utilize item response theory to assess the adequacy of the benchmarks in benchmark experiments.

While most of the statistical models discussed in paper *E* have an equivalent Bayesian software package to facilitate the analysis, the Bradley-Terry model did not. In paper *n*, we introduce the `bpcs` package to facilitate the analysis of paired comparisons. This package contains several extensions to the Bradley-Terry model and illustrates its use in behavioral sciences. In paper *s*, we propose the use of item response theory (IRT) to analyze benchmark data as well as to assess suitability of the benchmark suites in terms of difficulty and discrimination.

Objective 2 While the first objective consisted of a conceptual system not yet available commercially, the second objective represented a clear need faced by many of the companies in the Software Center, which are mainly in the embedded systems domain. This led to the design of the multiple case study and literature review presented in paper *F*. In this paper, we explore specific challenges faced by embedded systems companies and discuss potential solutions.

During study *B*, we observed that while A/B testing and experimentation were terms not commonly used at Ericsson, many teams conducted experiments. This led us to design study *h*, which was extended to study *G*. In this study, we explore the practices and processes used to drive experimentation in a mission-critical B2B system.

Paper *c* was a result of a master thesis supervision. In this paper, we investigate the challenges of designing and running A/B tests in software organizations with low control of the roadmap and large distance of the users.

At the same time we were conducting study *h*, we had the opportunity to collaborate with Netflix in study *g*. In this study, we investigate the Netflix Experimentation Platform and its vision to allow flexible design for scientists of diverse backgrounds. This departs from the traditional way companies run experiments, where software engineers are trained to conduct and interpret a small number of fixed designs. While more evidence is needed, an approach similar to Netflix allows more flexibility for embedded systems companies to start and scale an experimentation culture.

The results of the study *h* led to a multiple case study in collaboration with two automotive companies described in paper *j*. This paper investigates the actual challenges faced by automotive domain companies when introducing A/B testing. Paper *p* continues to explore experimentation in the automotive domain. In this paper, we investigate the use of a matching technique to minimize random imbalance and improve the sensibility of experiments in small samples.

Figure 4.1 represents the relationship between the included, related publications and the research questions addressed in this thesis.

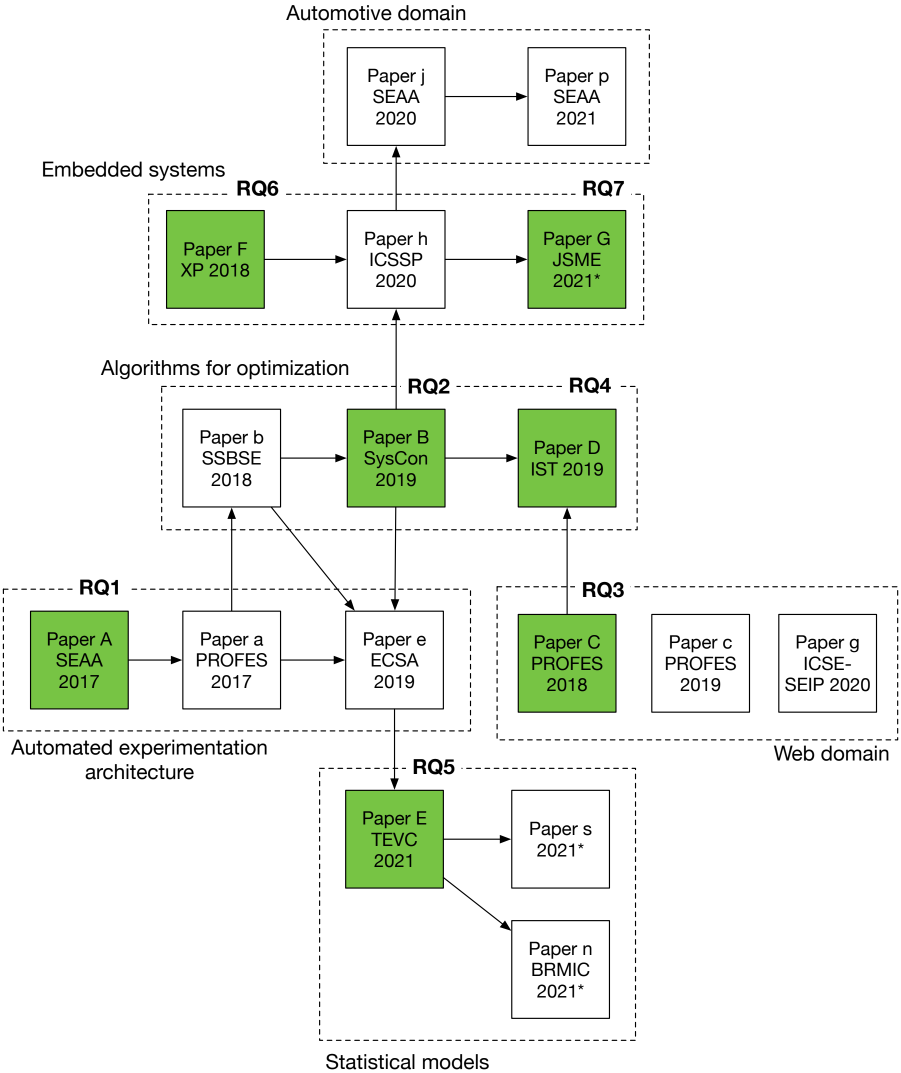

Figure 4.1: Relationship between the papers and the research questions. The included publications are highlighted in green. Papers with a star are currently under submission. The dashed lines group papers with a related theme. The arrows indicates the and how the different papers connect to each other.

4.2 Included publications

4.2.1 Paper A: *"Your system gets better every day you use it: towards automated continuous experimentation"*

4.2.1.1 Summary of the study

This study has two main goals. The first goal is to review relevant software architectures from the perspective of self-adaptive systems that have some desired architectural qualities for automating experimentation. These architectural qualities are external adaptation control, data collection as an integral part of the architecture, performance reflection, explicit representation of the learning component, decentralized adaptation, and knowledge exchange. The second goal is to develop an architecture framework inspired by existing architectures and based on the desired architectural qualities that can serve as the basis for automated experimentation systems.

4.2.1.2 Research method

The research method of this publication was conducted in three phases.

The first phase consisted of a literature review. We queried the indexing libraries Scopus and ScienceDirect and identified 34 relevant publications. Cross-references indicated an additional 18 papers that described relevant concepts for experimentation and for automating experiments through adaptation. In this phase, we also identified the relevant software architecture qualities that can support automated experiments.

The second phase reviewed the previously identified architecture for the desired architectural qualities.

The third phase consisted of developing an initial framework to support automated experiments inspired by the FUSION [103] framework, which satisfied most of the desired software architecture qualities. This framework is instantiated in a service robot for the human-robot proxemics distance interaction.

4.2.1.3 Main results

The main result of this paper was the first development of a software architecture for automating experiments (ACE) in software systems. It is worth noting that the proposed architecture is high-level and can be instantiated in different problem domains. For instance, in the service robot example, it was instantiated in the specific context of the Robot Operating System[1]. However, in the case studies discussed in [29] and in Chapter 6, the framework instantiated to provide an external experimentation layer for other systems. The system can provide A/B testing, MAB, single-objective multi-dimensional optimization with space constraints with or without regret minimization.

Figure 4.2 shows the proposed architecture framework.

[1] `http://www.ros.org/`

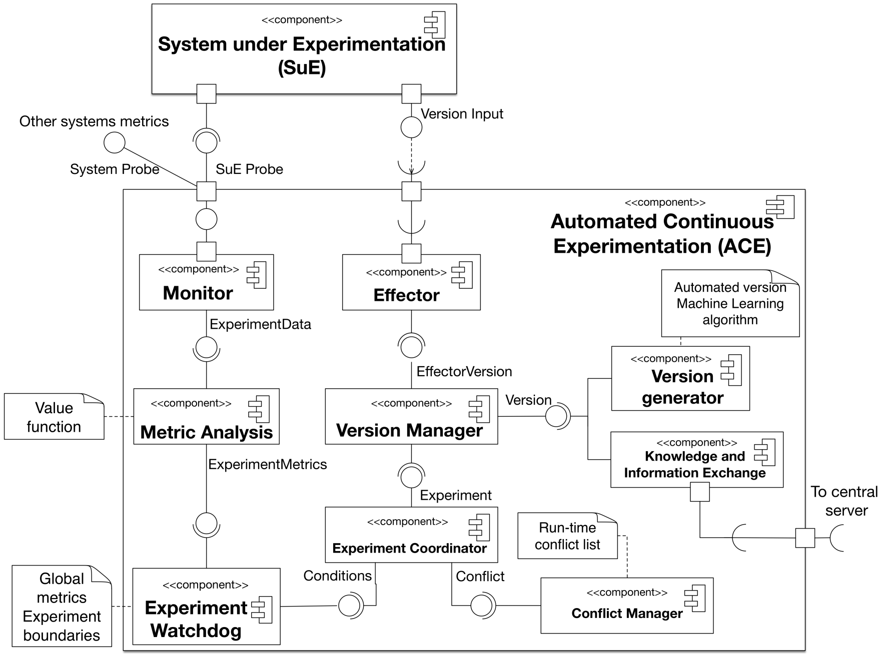

Figure 4.2: Automated Continuous Experimentation (ACE) architecture framework from Chapter 5.

4.2.2 Paper B: *"Automated Optimization of Software Parameters in a Long Term Evolution Radio Base Station"*

4.2.2.1 Summary of the study

This study is an empirical evaluation of the ACE architecture framework developed in Chapter 5. In this study, we instantiate the ACE framework in collaboration with Ericsson to address automated optimization of software parameters in a Long Term Evolution (LTE) radio base station. A radio base station has a high number of calibration parameters, that due to the complexity of the system and the deployed environment can interact with each other. Additionally, each mobile operator can set their own key performance indicator (KPI) metrics of interest to be optimized. Finally, optimization procedures conducted in live networks should consider regret minimization to minimize the impact on both the mobile operators and the final users.

4.2.2.2 Research method

This study conducts an empirical evaluation in a test bed at Ericsson. The test bed has a radio base station and multiple user equipment (e.g. cell phones). The user equipment utilizes pre-registered traffic profiles to interact with the mobile radio base station. This empirical evaluation optimizes the metric random access success rate (RASR) with two parameters maximum transmission power and cell range.

The ACE framework utilizes existing APIs from the radio base station and therefore does not modify the existing mobile radio station (and already validated) software. This is an important aspect that refers to the external adaptation control aspect of the architecture framework since many of the optimization algorithms and specifically MAB-based algorithms, are associated with technical debt [104]. The algorithm used to perform the optimization is an improved modification of the algorithm proposed in paper *b* [29], to address multi-dimensional optimization spaces.

4.2.2.3 Main results

We have shown that the ACE framework can be used in online optimization procedures in complex software-intensive systems from the empirical evaluation perspective. Its general approach allows the same experimentation system and interfaces to be used in collaboration with different industries, A/B and optimization experiments at Sony Mobile [29] and multi-dimensional experiments at Ericsson [30].

From the specific RASR case, the ACE system was able to find a configuration parameter that is 46.3% better than the default parameters of the radio base station. This result is illustrated in figure 4.3

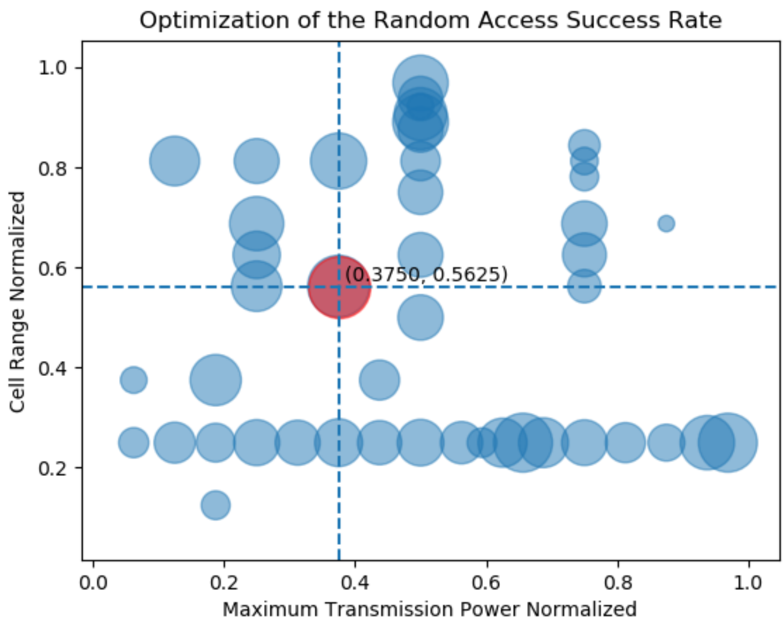

Figure 4.3: Optimization of random access success rate RASR based on maximum transmission power and cell range as discussed in Chapter 6. The central point in (0.5, 0.5) represents the default value and the red marker represents the optimized value.

4.2.3 Paper C: *"An activity and metric model for online controlled experiments"*

4.2.3.1 Summary of the study

Although previous research has presented many models for conducting and introducing experimentation in software organizations, these models do not provide enough details for organizations to implement a trustworthy experimentation process, scale, and run different types of experimentation. This paper analyzes the experimentation process used by the Analysis and Experimentation team at Microsoft, which runs over 20,000 experiments annually. This paper aims to analyze at a more granular level how to conduct trustworthy experimentation at scale and what parts of this process can be automated, semi-supervised, or should be manually conducted.

4.2.3.2 Research method

This paper utilizes an interview-based case study method [77]. The data was collected from nine semi-structure interviews. The interviewees were selected by one of the authors that worked on the team. All interviewees had a large experience in running experiments and developing the experimentation platform. The data was analyzed with thematic coding [82].

4.2.3.3 Main results

This paper provided three main results. First, we describe an activity model for conducting experiments. Figure 4.4 illustrates this model, which is described in more detail in chapter 7. Second, we describe how four types of metrics evolve, from creation to phase out. Third, we provide an overview of three qualitative findings that impact how experiments are planned and evolve, such as the role of customers and competition as a source of hypotheses, how metrics evolve with the business and how they capture unstated assumptions of the experiment. These three findings emphasize that a trustworthy experimentation process not only requires automation of certain parts of the process but also requires continuous manual intervention so the product evolves aligned with the business.

4.2.4 Paper D: *"Multi-armed bandits in the wild: Pitfalls and strategies in online experiments"*

4.2.4.1 Summary of the study

When discussing with practitioners about automating experiments in software systems, the topic often revolves around the use of multi-armed bandits, since an A/B experiment can also be formulated as a MAB problem. However, multi-armed bandits solve an essentially different problem and are not a substitute for scientific experimentation. This paper investigates how different companies have used MAB-based experiments and contextualizes these problems and solutions with A/B experiments. We discuss the problems associated with MAB experiments and what strategies are suitable to overcome these problems.

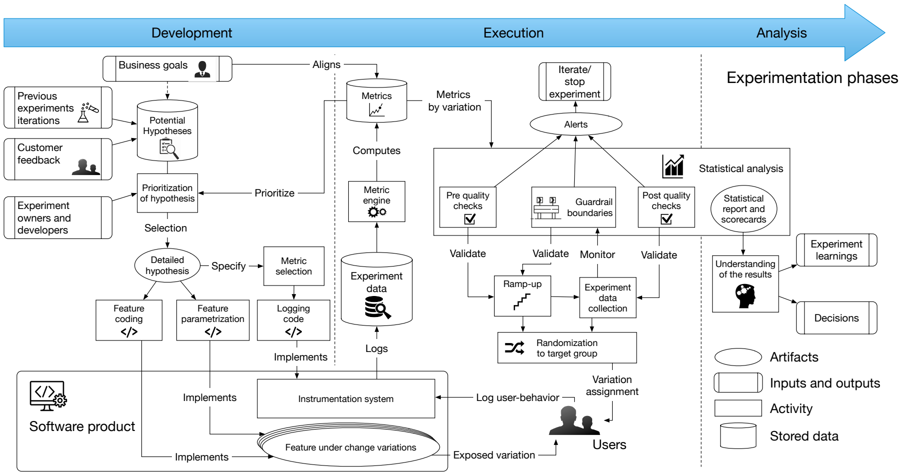

Figure 4.4: The experimentation activity model.

4.2.4.2 Research method

This research utilizes a multiple case study method [77] with eleven experts across five companies and simulations to triangulate and illustrate some of the identified problems.

4.2.4.3 Main results

In this paper, we identified three main restrictions and pitfalls, which are divided into nine reasons. For instance, one of the problems often observed in MAB-based experiments is the increased type I error which is often due to naïve implementations, violation of assumptions, and using MAB in essentially exploration problems. These problems can be addressed in different ways, such as using a traditional design of experiments (A/B testing or full factorial experiments), adding contextual information to be MAB algorithm, and implementing rigorous statistical analysis on top of the MAB experiment. We also provide guidelines for when practitioners should consider MAB-based experiments or A/B experiments.

4.2.5 Paper E: *"Statistical Models for the Analysis of Optimization Algorithms with Benchmark Functions"*

4.2.5.1 Summary of the study

Given the complexity of the software systems and the high number of interactions of the users with the system, the online optimization problem can be formulated as a black-box optimization problem. Decades of research have provided a high number of black-box optimization algorithms used in several different problems. During the development phase of these algorithms, researchers and practitioners traditionally test these algorithms against benchmark functions [63]. These functions have known topology and properties that can be used to evaluate how the new optimization algorithm works and compare different algorithms.

These algorithms are commonly compared with simple frequentist statistical tests in empirical evaluations to identify a statistical difference between these algorithms. However, these statistical tests are often misused, for instance, they do not take into account the problem structure such as repeated measures or correlation in the data, decisions are made solely on statistical significance without effect size considerations, investigations are made for each benchmark function individually, and the results are not properly reported.

This paper address this problem by proposing a series of Bayesian statistical models that allows researchers to make an integrated analysis of the benchmark suites, taking into account correlation in the data and proposing transparent practices for running and reporting the results.

4.2.5.2 Research method

This paper utilizes empirical evaluations of different algorithms in a benchmark suite (a collection of benchmark functions) to illustrate the use of the statistical models.

4.2.5.3 Main results

We provide three main contributions in this paper. First, we motivate the need for utilizing Bayesian data analysis (BDA) and provide an overview of this topic. Second, we discuss the practical aspects of BDA to ensure that our models are valid and the results transparent. Finally, we provide five statistical models that can be used to answer multiple research questions. These models are used to evaluate the probability of solving a problem (a binomial model), to evaluate the relative improvement (a linear regression model), to rank algorithms (a Bradley-Terry model), to estimate the number of evaluations to converge to a solution (a Cox's regression model) and to compare multiple algorithms for CPU time (robust regression model). All models include random effects term to model the intra-correlation introduced by repeated measures of the benchmark functions.

4.2.6 Paper F: *"Challenges and Strategies for Undertaking Continuous Experimentation to Embedded Systems: Industry and Research Perspectives"*

4.2.6.1 Summary of the study

This paper studies the challenges of introducing and adopting experiments as part of the development process in embedded systems.

4.2.6.2 Research method

This research was conducted in two parts. The first part is a literature review to analyze the challenges in adopting experimentation from the research perspective. In this literature review, we identified a total of 42 papers. We utilized thematic coding to classify the challenges observed in the literature.

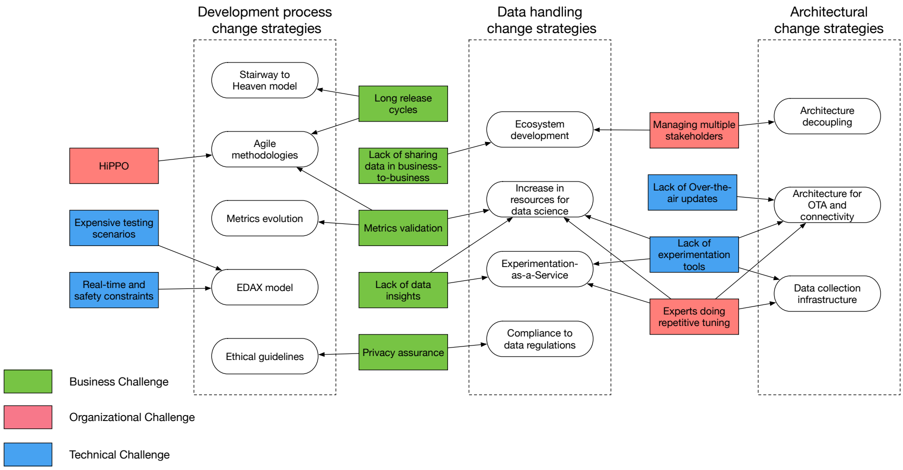

Figure 4.5: Summary of the challenges and the strategies faced by embedded systems companies adopting continuous experimentation.

The themes were then categorized with the perspectives presented first in [32], the business, the organizational, and the technical perspectives.

The second part is a multiple case study based on interviews and workshop sessions with five companies to understand the challenges from the industry perspective and how they are working to overcome them. The interviews were analyzed with thematic coding, and the results were compared with the challenges observed in the literature.

4.2.6.3 Main results

The main result of this paper is the identification of twelve challenges categorized in the business, organizational and technical perspectives. The solutions and potential strategies are categorized in terms of development process changes, data handling changes, and architectural changes.

These challenges and strategies are summarized in figure ,4.5 extracted from chapter 10.

4.2.7 Paper G: *"The HURRIER Process for Experimentation in Business-to-Business Mission-Critical Systems"*

4.2.7.1 Summary of the study

The development of mission-critical B2B systems differ in many aspects from the development of web-facing applications, including stricter validation procedures, service level agreements, for instance, ownership of the product and data, control of deployments, to name a few. This study investigates the use of the broader range of continuous experimentation practices in developing a telecommunications mission-critical business-to-business application in collaboration with Ericsson.

4.2.7.2 Research method

This study was based on a qualitative case study design following the guidelines proposed by Runesson and Höst [77]. The collected data consists of semi-structured individual and group interviews with 25 practitioners in six different locations in four countries and data collected from over 30 documents, including project documentation, feature development plans, solutions, and product presentations for both internal employees and external customers. The interviews lasted approximately one hour, and at least two authors were present in all interviews. The data analysis method followed the six-phase thematic coding process described by Braun and Clark [82].

4.2.7.3 Main results

The main contributions of this paper are the identification of four types of experiments, several experimentation practices, and the development of the HURRIER Continuous Experimentation process that combines existing constraints in B2B mission-critical systems with continuous experimentation practices.

The four identified types of experiments are business-driven experiments, regression-driven experiments, optimization and tuning experiments, and customer support experiments. Business-driven experiments are widely discussed in research and often associated with A/B testing. These experiments are aimed at validating and assessing business ideas and quantifying change. Regression-driven experiments are seen as a quality assurance technique designed to observe a negative impact. Although optimization and tuning experiments have been discussed previously in research, it is often mixed with business-driven experiments. We emphasize that in optimization experiments, often there is no new deployment, but rather just adjusting parameters of the software. Customer support experiments are a new type of experiment not previously discussed in research. These experiments aim at identifying the cause of a failure and negative impact when it is not possible to easily trace back to a particular change due to the complexity of the system and the deployed environment. All these types of experiments are discussed with concrete case studies.

In terms of the experimentation practices and techniques, the paper introduces a taxonomy for the different practices, for example, *experiment design and analysis*, *variation assignment*, *implementation techniques* and *release techniques*. In this taxonomy, we contextualized techniques already identified in research with new techniques and practices identified at Ericsson.

Finally, we introduce the HURRIER process that allows CE to be used to develop mission-critical B2B applications. The HURRIER process can be seen in Figure 4.6, extracted from chapter 11.

4.3 Related publications

This thesis includes seven appended publications. During this doctoral research, we have published other related publications relevant to this thesis but not appended. This section briefly discusses these publications and how they relate to the appended publications.

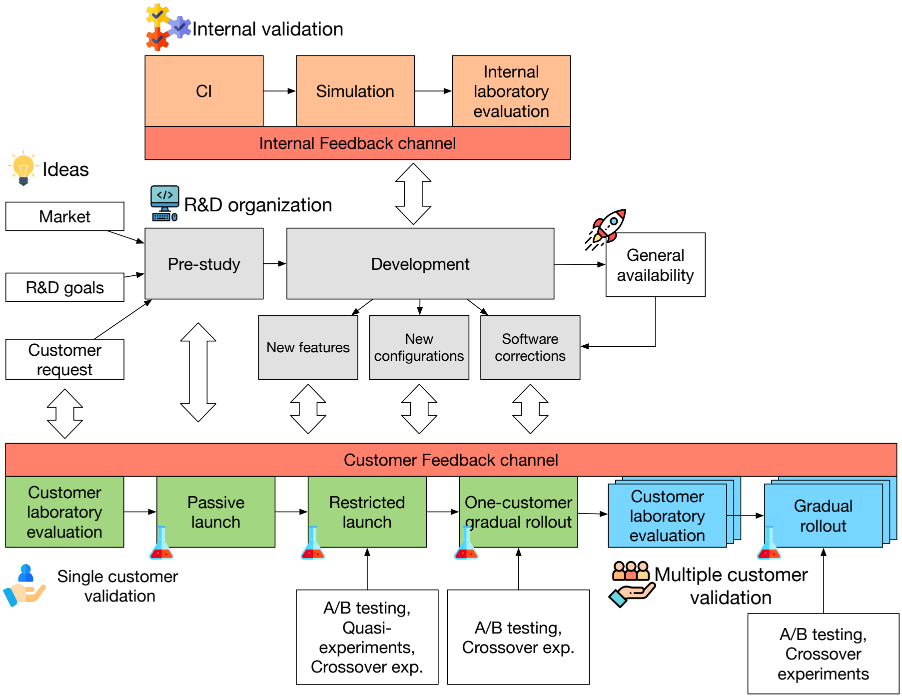

Figure 4.6: The HURRIER process presented in chapter 11

4.3.1 Paper a: *"More for less: automated experimentation in software-intensive systems"*

This paper details the architecture framework developed in paper *A* in terms of architectural design decisions [105]. In this analysis, we evaluate the design rules, the design constraints, the consequences, the pros and cons of each alternative to justify our decision in the architecture framework of paper *A*.

4.3.2 Paper b: *"Optimization Experiments in the Continuous Space"*

In collaboration with Sony Mobile, we instantiated the architecture framework from paper *A*, to use in conjunction with a mobile and office installation applications. The framework was instantiated in a cloud environment and supported traditional A/B testing experiments and online optimization. One of the online optimization constraints was to minimize the regret to lower the negative impact for the user.

In this paper, we developed a modification of an existing χ-bandit algorithm (HOO) [52] that provided better performance in the industrial context of Sony Mobile. The new algorithm, the limited growth hierarchical optimistic optimization (LG-HOO), was used to optimize the parameters of another backend algorithm that directly impacted the product's main features. A multi-dimensional modification of this algorithm was used in paper *B* to optimize mobile radio base stations.

4.3.3 Paper c: *"Continuous experimentation for software organizations with low control of roadmap and a large distance to users: an exploratory case study"*

This paper is based on a Master thesis conducted by the first author. In this paper, we investigate the specific problem of designing and running A/B tests in software organizations with low control of the roadmap and large distance of the users.

Low control of the roadmap refers to companies that develop products for another company and has to follow requirements and roadmap imposed by the hiring company. This has a direct impact on how experiments are planned and conducted. As discussed in paper C, experiment hypotheses often come from the development organization. In companies with low control of the roadmap, these hypotheses need to be approved by the hiring company.

Distance to users refers to data availability from the users (such as user behavior) and user feedback. As the case study discussed in the paper, companies with large distance to users do not have direct access to user behavior and feedback. The distance to the users and the low control of the roadmap are seen as blocking issues for running experiments.

4.3.4 Paper e: *"ACE: Easy Deployment of Field Optimization Experiments"*

This paper discusses modifications of the ACE architecture framework proposed in A to facilitate the adoption of optimization experiments in different stages of development, such as simulations, test beds, and live experiments. This architecture introduces domain-agnostic interfaces to allow for optimization procedures with minimal invasiveness and optimization expertise.

The system implements several optimization algorithms, including MAB-based, χ-bandits, and Bayesian optimization. These algorithms can be accessed using a simple API interface. Four steps are required to run an optimization experiment. First, a developer implements the optimization interfaces in the system under experiment (SuE). This step consists of parametrizing the part that is going to be optimized. The second step consists of configuring the experiment, specifying which algorithm, the constraint metrics and the objective metrics, and further restrictions on the search space. The third and fourth step consists of the actual optimization loop where the SuE requests new trial values to the ACE system and logs the trial results by updating the optimization model.

We discuss this architecture and simulation case study, testbed optimization with Ericsson and in a live experiment with Sony Mobile.

4.3.5 Paper g: *"Engineering for a Science-Centric Experimentation Platform"*

In this paper, we discuss the Netflix Experimentation Platform and the need for a flexible experimentation platform to allow scientists from diverse backgrounds to plan and conduct experiments.

Many software organizations aim to utilize and train software engineers to conduct experiments in the functionality they develop. While this strategy allows companies to increase the number of experiments, it has the drawback of limiting what type of design and analysis the development team is allowed to do. Netflix has taken a different approach for its experimentation platform. Instead of relying on a small number of designs conducted by software engineers, Netflix focuses on allowing scientists from diverse backgrounds to plan and conduct experiments with teams they are embedded into. This has greatly increased the pace of innovation and experimentation in Netflix and allowed for deeper strategy discussions and richer analyses.

The platform supports this flexibility by simplifying the experimentation process for new types of analysis by adopting a non distributed architecture and scientific languages such as R and Python. Scientists can create their designs and analysis in those languages and contribute to the experimentation platform which scales these reproducible Jupyter Notebooks as new analysis flows.

4.3.6 Paper h: *"Experimentation for Business-to-Business Mission-Critical Systems: A Case Study"*

Paper *G* is an extension of paper *h* and contains several additional contributions. In paper *G*, (1) we provide a classification of the different types of experiments, practices, and techniques used in B2B mission-critical systems; (2) we provide a revised version of the HURRIER process to include relevant information to complement the different types of experimentation; (3) We provide, in addition to the original case study, we added three new case studies for the other types of experimentation; (4) We include new relevant discussion for the new research questions.

4.3.7 Paper j: *"Automotive A/B Testing: Challenges and Lessons Learned from Practice"*

This paper investigates the use of A/B testing in the automotive domain. Unlike previous research, which focuses on hypothesized or toy scenarios, this paper investigates the challenges of adopting A/B testing in real experimentation with two large-scale automotive companies. This paper utilizes a case-study method [77] with two companies. The data collection utilizes three main sources. The first consists of 12 semi-structured interviews with 14 employees. The second consists of notes from weekly meetings of a working group in an A/B testing iteration. The third source of data consisted of material produced in a workshop with the development team of the feature being experimented with. The main results of this paper are the identification of the challenges faced in practice, such as the high number of vehicle variants, restricted number of test vehicles, or lack of support for data-driven development in the AUTOSAR architecture.

4.3.8 Paper n: *"Bayesian Paired-Comparison with the bpcs Package"*

In paper *E*, we have utilized a Bayesian version of the Bradley-Terry model to rank the different algorithms. However, there were no software packages or research that implemented these models at the time of the writing. This paper introduces an R package for the analysis of paired comparison data, the bpcs package. This package implements the Bayesian Bradley-Terry model and many of its extensions, such as for order-effect, ties (Davidson model), random effects, generalized models, and subject-specific predictors. The Bayesian inference is performed using the Stan probabilistic programming language and the Stan No U-Turn sampler. The examples provided in the paper are focused on behavior research. Nevertheless, the vignettes on the package also show a replication of one of the results of paper *E* and the use of these models for sports research.

4.3.9 Paper p: *"Size matters? Or not: A/B testing with limited sample in automotive embedded software"*

This paper explores a minimization technique to address the random imbalance of A/B testing experimental groups with small samples in the automotive domain. We utilize the Balance Match Weighted method [106] to create experimental balanced groups in a fleet with 28 vehicles for an experiment with vehicle energy management.

The Balance Match Weighted generated more balanced groups compared to random sampling. The results of using this method were validated with an A/A test. These results indicate that minimization techniques can be used in the context of A/B testing in embedded systems if there are prior information about the experimental subjects (pre-experiment data) and domain-specific knowledge about the potential influence of different covariates.

4.3.10 Paper s: *"On the assessment of benchmark suites for algorithm comparison"*

In this paper, we analyze the suitability of the benchmark functions for the analysis of algorithms. While in paper *E* we proposed different statistical models, here we propose the use of item response theory (IRT) to evaluate not only the algorithms but the suitability of the benchmark functions in terms of difficulty and discrimination. We show that common suites used for algorithm comparison have poor discrimination factors and are either too difficult or too easy. We conclude the paper highlighting potential uses of IRT for improving the design of benchmark suites and analysis of benchmarking data.

Chapter 5

Paper A

Your system gets better every day you use it: towards automated continuous experimentation

Mattos, D. I., Bosch, J., Olsson, H. H.

43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2017, pp.256-265.

Abstract

Innovation and optimization in software systems can occur from pre-development to post-deployment stages. Companies are increasingly reporting the use of experiments with customers in their systems in the post-deployment stage. Experiments with customers and users are can lead to a significant learning and return-on-investment. Experiments are used for both validation of manual hypothesis testing and feature optimization, linked to business goals. Automated experimentation refers to having the system controlling and running the experiments, opposed to having the R&D organization in control. Currently, there are no systematic approaches that combine manual hypothesis validation and optimization in automated experiments. This paper presents concepts related to automated experimentation, as controlled experiments, machine learning and software architectures for adaptation. However, this paper focuses on how architectural aspects that can contribute to support automated experimentation. A case study using an autonomous system is used to demonstrate the developed initial architecture framework. The contributions of this paper are threefold. First, it identifies software architecture qualities to support automated experimentation. Second, it develops an initial architecture framework that supports automated experiments and validates the framework with an autonomous mobile robot. Third, it identifies key research challenges that need to be addressed to support further development of automated experimentation.

5.1 Introduction

Autonomous systems already play a big part in several areas, from financial markets, industrial robots, airplane navigation systems to the development of autonomous vehicles. These systems are deployed in uncertain environments and contexts with an ever-increasing demand to work more autonomously. As these systems become more and more complex, it is not clear how the developed features could be contributing to the system and whether they are delivering the expected value [26].

Previous studies show that in the development of systems, traditionally, the prioritization of a feature is usually driven by earlier experiences and beliefs of the people involved in the selection process [11]. The development of the new features should be, ideally, data-driven and done systematically [107]. The development of a full feature from conception to user deployment can result in inefficiency and opportunity cost if it does not have a confirmed value for the customer. Early customer feedback is important to determine and validate the feature value [3]. If a gap between the expected and the actual value of a feature is identified, the feature under development can be refined or abandoned. Additionally, companies continuously collect data from deployed software [14, 15]. Although not widely used in industry, post-deployment data can be used for both continuous optimization and improvements and to drive innovation in features of the existing products [15]. Testing and experimenting with customers and users are used as problem-solving processes and are critical to organizational innovation [12].

Frequently, complex systems and user behavior in development of software-intensive systems lead to opinion-based decisions captured in traditional requirements [2]. This mismatch between user behavior and opinion-based requirements is leading industry to change from requirements-based to experiment-based development [12].

Systematic experiments provide a level of understanding about what really works and leaving opinion-based decisions behind [108]. Web-facing companies report the use of experiments with customers and that these experiments can lead to a significant learning and return-on-investment [2]. Several companies report the use of A/B experiments for confirming feature value through hypothesis testing and for feature optimization [2, 32] (in short A/B experiment consists of comparing a variant A, the control, against a variant B, the treatment). In [12] online experiments are identified as a technique to drive innovation during development and post-deployment of a system.

However, traditional manual controlled experiments can be an expensive way to optimize a system. Validation runs online and can last for several months [2], the metrics used by different teams can lead to conflicting business goals, and it can be hard to reason on the system when dealing with hundreds of different observable variables [36]. In this scenario machine learning and artificial intelligence techniques can aid the research and development team to run optimization procedures to existing features more efficiently.

In customization and recommender systems there is an increase use of machine learning (ML) and data-driven approaches. Techniques such as deep learning, reinforcement learning (RL), deep reinforcement learning, multi-armed bandits are getting large attention as companies (Google, IBM, Microsoft, Spotify, Facebook among others) successfully report the use of those techniques in their systems. Machine learning communities continuously report solutions to a vast range of difficult problems. However, ML also raises several software engineering problems, such as testing, system validation, supporting infrastructure and increase of technical debt in the system [104]. Many ML algorithms have mathematical proofs but with the ever-changing landscape that those algorithms interact, system validation becomes a hard problem. Controlled experiments are used to evaluate both the ML value and system behavior [109].

As companies start to build autonomous systems with an increasing level of autonomy, experiments are seen as scientific way to learn and adapt the system behavior. The uncertainty raised by the environment, the interaction with humans and other agents all impact in the system behavior in unknown ways. It is unfeasible to grow the size of the R&D team with the increasing demand for experiments . This calls for the use of automated experiments, where the R&D team can build part of the functionality and guardrails where the system can experiment and learn from this process continuously and autonomously [26].

As companies are moving towards experiment-based development, they face several challenges such as: experiments in live field are time and cost expensive, experiments from different teams can lead to conflicting goals, and the lack of a systematic approach to run experiments in different domains leads to several experiment platforms. [2, 18, 26, 36]. To address some of these challenges the communities studying experiments in software can be viewed as moving towards automated experiments. Important research deals with automating data collection and data analysis [36], facilitating the deployment of new experiments in web systems [19], dealing with overlapping experiments [18], leveraging experiments with log data [110] and automated experiments from a algorithm perspective [25].

Automating tasks and allowing systems to perform adaptation online play a role part in the development of autonomous systems. Autonomous and adaptive systems are able to automatically adjust their behavior at runtime in response to changes in the operating environment [111]. Over the past fifteen years, different domains such as the self-adaptive systems community studied and developed several software engineering approaches and techniques for adaptation. While the ML is mostly concerned with algorithms and the theoretical basis of learning, these communities study the software engineering challenges.

Running continuously automated experiments in the already deployed systems, with both the aim of improving the system behavior and confirming the delivered value, requires novel software architectures and engineering approches [26]. Techniques from these different perspectives can serve as a basis for bridging from manual hypothesis experimentation and experimentation for optimization to automated experiments.

Although we recognize the important role ML and other artificial intelligence techniques have in the development of automated experiments, this work focuses on architectural aspects to support different techniques in automated experiments for both validation as well as optimization of features. Continuous optimization through automated experiments leads to systems that get better every day we use them.

The contribution of this paper is threefold. First, it identifies a set of software architecture qualities to support automated experimentation and analyzes several architectures from these qualities perspectives. Second, a framework to support automated experiments is developed based on the presented analysis. This framework presents a novel way towards automated experimentation, where manual hypothesis experimentation and fully automated experiments in optimization can be combined. This framework is validated in the context of an autonomous system in a human-robot interaction problem. Third, it identifies the key research challenges that need to be addressed to support the development of automated experiments.

The rest of the paper is organized as follows. Section 5.2 provides a background on some of the related areas, controlled experiments, machine learning applications for experiments and software architecture for adaptation of systems. Section 5.3 describes the research method. Section 5.4 provides a list of desired architecture qualities to support automated experimentation. Section 5.5 analyzes several software architectures in light of the discussed qualities. Section 5.6 shows an initial architecture framework that satisfies the discussed qualities for automated experimentation. Section 5.7 provides an implemented version of the architecture framework in an autonomous mobile robot on the proxemics distance problem. Section 5.8 concludes and discusses future research challenges and future works.

5.2 Background

This section reviews some concepts on controlled experiments, reinforcement learning in experiments and software architectures for adaptation. These concepts are complementary to each other and form the basis of this work.

5.2.1 Controlled experiments

In controlled experiments, researchers manipulate independent variables in an experiment and observe the effect on the behavior by measuring the dependent variable.

In software development, experiments with customers can lead to a significant learning and return-on-investment [2]. The web provides an opportunity to test and evaluate development ideas using controlled experiments. Kohavi [2] provides an overview on A/B experiments controlled, techniques, study cases in a web environment and common pitfalls. Controlled experiments provide a reliable experimental setup for understanding the system, however it can be an expensive method to evaluate new ideas.

When the sample cannot be considered to be independent (the case of some experiments social networks), networked A/B testing techniques [20] are used. When running several experiments at the same time, overlapping of experiments and confounding factors can influence the results. In [18], some strategies such as dividing the experiments in layers are discussed.

5.2.2 Reinforcement Learning in experiments

Reinforcement learning can be seen as a technique for solving sequential decision making problems [112] and largely developed by Sutton and Barto [113]. The method is based on agent that repeatedly interacts with the environment and is receiving feedback from it. The perception of the each state allows the system to continuously improve its state into a more optimal selection of actions. It is modeled as a Markov decision process. Reinforcement learning focuses on online improvement and presents the trade-off between exploration and exploitation. The exploration/exploitation problem consist of finding a balance on exploring different solutions to achieve an optimal solution and exploiting the best solution.

The explore/exploit trade-off is well studied with the multi-armed bandit problem. This problem consist of deciding which arm of a K-slot machine would maximize the total reward in a limited series of trials. Several algorithms for this problem were proposed and evaluated [114]. Experiments with a finite number of treatments can be formulated as bandit problems and A/B testing is mathematically equivalent to the ϵ-first strategy with equal division between the arms (ϵ-first strategy explore all arms choices before it starts to exploit). Complex strategies can take into account context information (contextual multi-armed bandits). Contextual bandits are wideley used in personalized/recommender systems [115], ad placement [110] and in search engines applications [116].

However, bandit problems differ from controlled experiments in the conceptual level. Controlled experiments focus on hypothesis testing and obtaining an understanding of the system with a statistical confidence level. Bandit problems focus on optimization and do not balance the experiments to accurately estimate the inferior treatment effects [117] .

5.2.3 Software architectures for adaptation

Adaptation refers to the ability of the system to automatically adjust their own behavior in response to changes in the operating environment [118]. Adaptation is seen as a key enabler for the development of autonomous systems and autonomic computing.

In 2003 Kephart, introduced the MAPE-K model with the IBM Autonomic Computing Initiative [119]. The MAPE-K model is a feedback loop and it stands for Monitoring, Analyzing, Planning and Executing over a Knowledge base. The MAPE-K loop forms the basis of self-adaptive systems feedback loops and has become the reference model in self-adaptation. Self-adaptive systems can be seem as an umbrella term to cover different areas involved in adaptation and are studied from different perspectives: software architecture, requirements engineering, middleware, component-based development, control systems theory among others [120]. These different perspectives solve their specific domain problems with different architecture configurations. However, most self-adaptive systems approaches can be mapped into one or more MAPE-K loop parts [120].

5.3 Research Method

The research method in this paper was conducted in three phases.

Phase I: In the first phase, we reviewed literature to identify relevant software architectures for experimentation and adaptation and how these architectures solve domain-specific problems.

For the literature search, we looked for software architectures in the different domains and applications related to experimentation:

```
(TITLE-ABS-KEY("software architecture") OR
TITLE-ABS-KEY("software framework")) AND
(TITLE-ABS-KEY("experimentation") OR
TITLE-ABS-KEY("A/B tests") OR
TITLE-ABS-KEY("split tests") OR
TITLE-ABS-KEY("self-adaptive systems") OR
TITLE-ABS-KEY("controlled experiments")).
```

This query was used in the indexing libraries SCOPUS and Science Direct, that cover the large research libraries. As a result of the literature search, we identified 410 papers with relevance to this search.

From these papers, we identified 178 based on reading the abstract and identifying the relevance in relation to our research topic. From this we selected 34 papers that describe relevant concepts and software architectures for experimentation, adaptation and controlled experiments. In addition to the literature search based on the identified key search terms, through cross-reference we identified a set of 18 additional papers with relevance for our research.

The analysis of the selected works provided us two outcomes: (1) the identification of relevant architectures for experimentation, adaptation and controlled experiments and (2) identification of software architecture qualities that can support automated experimentation. The qualities are described in detail in Section 5.4.

Phase II: Based on the outcomes of phase I, we scoped the second phase to understand how each identified architecture implements the software qualities. In this phase, we revisited the identified architectures to identify how they implement or solve a problem related to each of the qualities. Our analysis, described in Section 5.5, indicated that none of the architectures could be used as is with the automated experimentation problem but they can be used to derive a new framework that could support automated experimentation.

Phase III: Based on the analysis performed in phase II, phase III focused on developing an initial framework that could support automated experiments. An initial version of this framework was instantiated in a proxemics distance problem in human-robot interaction. This human-robot interaction problem is currently being studied using manual experiments and therefore is a good candidate to try the developed automated experimentation framework.

5.4 Architecture qualities

This section discusses the identified architectural qualities that support the idea of automated experimentation.

The identified qualities are outcomes of phase I of the research method, described in Section 5.3. In our research, we identified six qualities that would constitute our approach towards automated experimentation. However, this list of qualities is not static. Further research on the area might extend this list to include different qualities. Additionally, the identified qualities are seen as desired qualities rather than required qualities. Excluding one or more of the qualities might be needed to implement a domain specific setup for automated experiments. Therefore, this list is ordered in terms of importance.

5.4.1 External adaptation control

Adaptation can be divided in two types of adaptation control: internal and external [121]. Internal adaptation refers to interlace application logic with adaptation logic. External adaptation refers to the use of an adaptation manager that coordinates adaptation with the use of sensors and effectors in the managed system (application logic). The use of an external manager increases maintainability of the system. However, it introduces an overhead to the system, penalizing performance.

Traditional controlled experiments are analyzed offline. All the data is extracted, analyzed to support a decision. Depending on the decision the system is manually modified to incorporate this decision. Systems running multi-armed bandits and other ML algorithms are usually incorporated in the feature application code. The external adaptation control was identified as an important quality for automated experiments, because it allows separating the application logic from the experiment logic. This helps both automated experiments in controlled settings and in optimization settings, by facilitating to add automated experiments to existing features and removing it from features that already reached the static system loop [26].

5.4.2 Data collection as an integral part of the architecture

Software-intensive systems can gather large amounts of data in real-time, from the context, from the internal system as well as from users of the systems. Collecting data for online and offline analysis by both the system and the R&D should be an integral part of the architecture. An emphasis is given in this step as the development of software should be data-driven [11].

As most adaptive systems are based on the MAPE-K reference framework, data collection is usually seen as part of the monitor phase on the MAPE-K loop. The managed system exposes several observable internal states through the sensors touch points [122]. Data is collected continuously and filtered and later analyzed by the Analyze component to detect changes and decide whether adaptation is needed or not.

Machine learning systems are easy to develop but hard to maintain [104] due to hidden technical debt. One of the causes is the high diversity data and unstable data dependencies that rises by the extensive use of *glue code*. Strategies to create common API's allow a more reusable infrastructure supporting integral data collection and the external adaptation.

5.4.3 Performance reflection

As part of the learning process, the system should have performance reflection as an important part of the architecture. Performance reflection consists of evaluating the current behavior according to an expected value function.

The experimental methods also use the performance reflection to validate the observed behavior.

In the conceptual solution presented in [26], evidence-based engineering refers to validate the deployed feature based on experiments and the value it delivers. Also, it was identified three levels of evidence-driven development feedback loop: the R&D loop, the dynamic system loop and the static system loop. Performance reflection occurs in the dynamic system loop in which the system can compare its delivered value with the initially expected value.

If the system does not keep track of its adaptation performance it is not possible to have an evidence that the experimentation and the learning process is increasing value to the system.

5.4.4 Explicit representation of the learning component

Most of the current approaches in self-adaptive systems recognize the importance of learning, but few of them support an online learning process [120]. Most architectures use predefined and reactive adaptation plans.

The field of ML provides several algorithms and techniques tailored to domain problems. Experimenting features in different domains requires modification of the learning algorithm. An explicit representation of the learning component facilitates introducing and maintaining (mitigating entanglement effects) learning algorithms in the optimization process. We believe that an explicit representation of the learning component is necessary because it reinforces the learning characteristic of the systems and facilitate the reuse of the learning component and learning algorithms in other features.

5.4.5 Decentralized adaptation

Centralized control refers to using a single adaptation manager to control the adaptation. Decentralization refers to each adaptation where each sub-system has a full adaptation manager. The use of decentralized adaptation control improves the performance of the system, splitting responsibility between subsystems. Hybrid approaches might use different patterns, such as the IBM hierarchical structures to allow adaptation [122].

Different systems running automated experiments will require different levels of decentralization. Centralized systems can grow quickly in complexity handling several features under experimentation. Traditional A/B experiments relly on centralized coordination. However, as the infrastructure grows decentralized patterns start to emerge. Tang et al. [18] describe an hybrid approach, dividing the system in layers with different levels of overlapping crossing multiple domains. We believe that automated experiments will require decentralized solutions to support the increasing number of concurrent experiments.

5.4.6 Knowledge exchange

Knowledge exchange can help systems to share learned and experimented solutions [123]. Collaborative learning is an increasing topic of interest in ML and in experimentation. Systems instantiated or users experimenting might not be completely independent or randomized. Algorithms for solving this sort

Table 5.1: Analyzed architectures

Approach	**Architecture**
Architecture-based	Rainbow framework [125], 3-layered approach [126], Archstudio [127]
Reflection-based	DynamicTAO [128] CARISMA [129]
Control-based Architectures	Model Predictive Control, MIAC, MRCA, Gain scheduling, Cascaded control [130]
Service oriented	SASSY [131] MetaSelf [132] MOSES [133] MUSIC [134]
Agent-based	CRL [123], Unity [135], MOCAS [136]
Learning systems	FUSION [103], Controller-Observer [137]
Requirement Engineering methods	LoREM [138] Zanshin [139]

of problem are studied within networked A/B tests [36] and counterfactual reasoning [110]. As the feature being experimented might also be evaluated by the development team in terms of the value it gives, the sharing component should also allow communication with the development team. Knowledge exchange can be seen in the work by [124] and in the area of collaborative feedback and Collaborative Reinforcement Learning [123].

5.5 Architecture analysis

Due to page limits constraint, this section discusses briefly the analyzed architectures and how they can contribute to the development of automated experiments. We analyzed the existing architectures for adaptation and experimentation in relation to the identified qualities as stated in the research method.

Architectures from different domains were selected in order to minimize the bias in selecting only a few. However, this list does not aim to be complete in respect of all existing architectures for adaptive systems. A description of the different approaches can be found in [120]. Table 5.1 provides an overview of architectures analyzed in this work.

The architectures were analyzed, classified and ordered according to the six qualities listed in section 5.4. Figure 5.1 shows the obtained classification

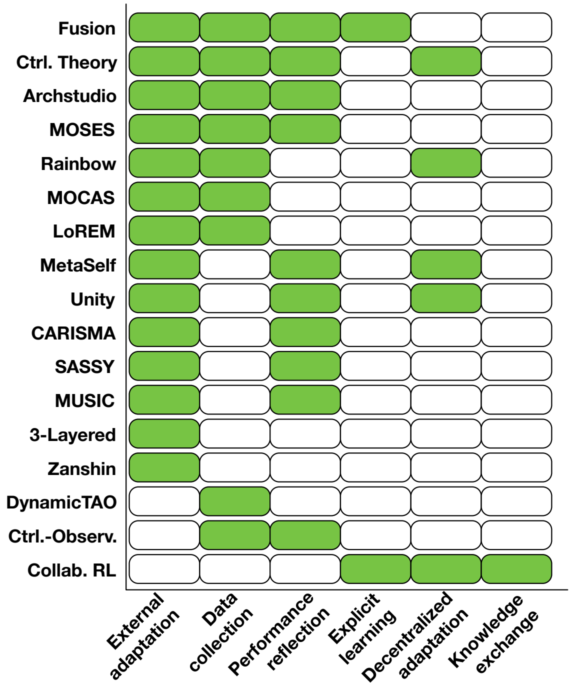

Figure 5.1: Summary of the classification of the architectures with respect to the architecture qualities. In green, are the qualities satisfied by the architecture. The architectures were ordered according to the relative importance of the quality.

of the analyzed architectures according to the listed qualities. Each of the six qualities is connected to the architectures that fulfill these qualities.

The summary table provides an overview of how the different architectures relate to the desired qualities. However, it is possible to see that most of the approaches deal with only a few of the identified qualities. The architecture that satisfies most of the qualities is the FUSION framework [103] and the developed architecture framework in Section 5.7 is inspired by this framework.

The FUSION framework can be seem as a variation of the MAPE-K loop for architecture optimization. It uses a learning approach to drive adaptation of features. It is focused in optimization in the architectural level and it is independent of the learning algorithm. The FUSION framework uses a centralized external approach. The adaptation manager is divided into two cycles, the adaptation and the learning cycle. The learning process is based on measurements of the system. After the collection, the learning cycle identifies any emerging pattern and refine the induced model in a knowledge base. Then, the knowledge base is used in the adaptation decision. Objective functions are used as metrics for the learning process and for the decision making. This framework brings together many of the identified qualities, except for the decentralized adaptation and the knowledge exchange. However, it focuses on optimizing from learning from existing features in the knowledge base in

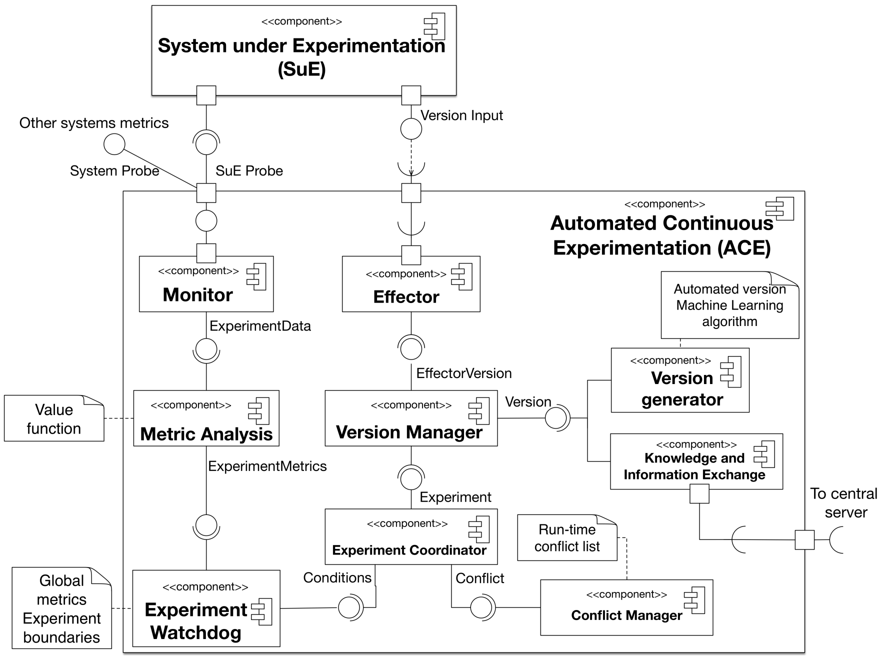

Figure 5.2: Automated Continuous Experimentation architecture framework

the architecture level. Automated experimentation is focused on learning and optimizing business goals to drive post-deployment innovation.

5.6 Architecture framework

In this section, we describe an initial architecture framework that satisfies the identified qualities to support automated experimentation.

5.6.1 The architecture framework

The presented architecture framework is represented in Figure 5.2. This architecture is the result of the design decisions over several iteration processes and inspired by the analyzed architectures described in Section 5.5.

The intention of this architecture framework is to provide for an existing system the capability of doing automated experiments decentralized in feature level, rather than providing an architecture for the whole system. Manual hypothesis testing is integrated as the hypothesis are still formulated by the R&D team.

The presented architecture modules are described next:

Monitor. This module is responsible for the data collection. The data collected come from the probes available in the system and in the SuE (system under experimentation), therefore both local and global behavior. This module is directly related to the data collection in the discussed qualities. Access to all the necessary data for experimenting requires proper instrumentation of the code. This module does not represent only a stream of raw data into the automated experimentation architecture framework. It represents data processed that add information to the system.

Effector. This module is responsible for interfacing with the managed system. Besides the monitor, it is the only point of contact with the rest of the SuE. This module requires that the managed system exposes interfaces for interaction with the system. This concept of not intermixing the experimentation code and the managed system code is directly related to the external adaptation quality. The same observations made to the monitor module are valid for the effector.

Experiment coordinator. This module is responsible for running the experiment and coordinating with the version manager. This module controls only the specific SuE, other experiments have their own experiment coordinator modules. The experiment coordinator can control experiments such as A/A (control variant A against the same variant for sanity checks), A/B/n (controlled experiment with more than one treatment), explore/exploitation and crossover experiments. This module keeps track on when to experiment, the number of experiments that should be run, which solution is more significant. This module receives inputs from the conflict-list manager if it is allowed to run an experiment or not. It also receives inputs from the experiment watchdog module, if the system is deteriorating any global metrics, if it went out of boundaries or if it still needs to perform more experiments.

Version Manager. This module is responsible for managing and generating different versions to experiment. This can be acting in parameters or replacing whole sub- component models. The version generator keeps a list of the versions used and accepts versions inputs from the Knowledge Exchange module and the Version Generator. This allows the system to experiment both automatically generated versions, as well as manual versions crafted by the R&D team. Although this module is not directly connected to a one of the design decision listed, this module is linked to both the functional and quality requirements

Version Generator. This module can accommodate different artificial intelligence algorithms that we might want to test. The generation algorithm is not specified, but it could include machine learning algorithms, such as reinforcement learning algorithms, genetic algorithms, parameter scheduling or randomized versions. This module is directly connected to the learning quality.

Experiment Watchdog. This module checks the conditions that the system can run the experiments, such as when the system should continue experimenting and when it should stop. If the system goes out the predefined boundaries or if there is deterioration in global metrics this module can stop the experiment and return the system to the "safe" version. Having a stop condition for global metrics prevents the system improving a local metric, but degrading a global metric. If any of the stop conditions is reached this module signals to the experiment coordinator to stop the experimentation process or to roll back to a safe version.

Conflict-list manager. This module keeps track in run- time of components that are being experimented with and which factors it affects. This is an important component in a decentralized experiment environment. Several other systems of the robot can be experimenting. This manager keeps track of those systems in order to avoid confounding variables in the experiment. This is directly related to the decentralized adaptation quality. In a generic implementation, the conflict manager advertises its current state (experimenting or not) and listen to other conflict manager's states.

Metric Analysis. This module is responsible for keeping track of the managed system behavior and the value function. This module serves as a trigger to drive the adaptation through optimization or through keeping track of the validation process. In this module, we insert the value function and we run our statistical analysis. This module is directly related to the performance reflection quality.

Knowledge and Information Exchange. This module communicates with the optimization and experiment validation module and with the external world. This module is responsible for sharing discovered solutions in the experimentation process and also for sharing and learning the validated solutions from the experiment through a central server infrastructure. This also represents a way in which the R&D can interact with the system, either helping in the analysis step or proposing different versions not generating by the version manager, for example, testing different algorithms. This module is directly related to the Knowledge Exchange quality.

5.7 Automated experiments in a human-robot interaction problem

In this section, we present an experimental implementation and evaluation of the automated continuous experimentation architecture framework. In this experimental scenario, we first validate the correctness of the architecture behavior. Second, our architecture for automated experimentation indicates a more cost-effective solution for running experiments compared to manual experimentation.

5.7.1 Proxemics distance in Human-Robot Interaction

Human interaction is based on several unwritten and subjective rules. One example is respecting other people's personal space. In human-human relations, several social factors play an important role in this interaction. Not conforming to these rules may cause miscommunication and discomfort. To have a good human-robot interaction, the robots must follow similar rules [140]. The large body of work in Human-Robot Interaction (HRI) identified several factors that come into play in proxemics distance, such as gender, age, personal preferences, technology involvement, crowdedness, the direction of approach, form factor and size [140].

Different works recognize some base distances and how they are influenced depending on a change of factor. However, this is still an open problem. The development of new robots and the deployment of these robots in very different contexts (e.g. different countries) require new manual experiments to validate and optimize the proxemics distance.

The presented automated continuous experimentation framework can be used to allow the robot to try different distances and learn an optimal distance on the context that it is inserted. This would allow robots to adapt its proxemics distance and validate it without the need of designing costly experiments for each different context.

The developed architecture framework is instantiated in the open source mobile research platform Turtlebot 2[1]. This platform is similar in size and form factor with several commercial mobile companion robots.

The robot runs the Robot Operating System (ROS)[2] middleware. ROS is a widely used framework for writing robot software. It is a collection of tools, libraries and conventions that aim to allow creating complex and robust robot behavior across different robotic platforms.

The instantiated architecture consists of six ROS nodes that can be mapped to the architecture framework modules. Each node was implemented as a separate process, communicating through a mix of publish-subscribe and client-server methods as defined by ROS. Figure 5.3 shows the instantiated framework. In this initial step of the research we are focusing on implementing the framework in one system initially, therefore the knowledge exchange quality was the only one not contemplated in this experimental validation. Full implementation of the automated experimentation architecture can be seen in `https://github.com/davidissamattos/david_ws`.

Feedback monitor: this node can be mapped directly to the monitor module. It is responsible for capturing the human feedback. In this case, we receive input from both verbal feedback (e.g. "Too far") and non-verbal feedback (e.g. if it is too close the person steps back).

Metric Analysis: this node receives as input the value function of the system and analyzes input data. The value function we are using is the user satisfaction. We expect our user to be satisfied at least 70% of the approaches in the long run (after a minimum number of experiments).

Experiment Watchdog: this node verifies the current state of the system and the boundaries of our problem. For this system, we defined some boundaries such as, do not get closer than 20 cm from the human, do not experiment if the battery is low and if the experiment is performing poorly (e.g. less than 30% of the cases) we rollback to another version. This module works even if the version manager generates values outside the constraints of the experiment (by implementation/runtime errors).

Experiment coordinator: this node is responsible for keeping track of which experiment is going on, optimization, validation or running in static loop mode.

Version manager: the version manager node receives input from the experiment coordinator regarding which experiment is running. It the feature is running in safe-mode it uses a safe predefined distance. If the learning process has converged the experiment coordinator requests a static learned version. The version manager generates these versions either by static input or by calling a learning module to generate it (e.g. calling the machine learning module).

Machine learning: the version manager requests new versions to the machine learning module. This module uses the K-means clustering algorithm

[1] `http://www.turtlebot.com/`
[2] `http://www.ros.org/`

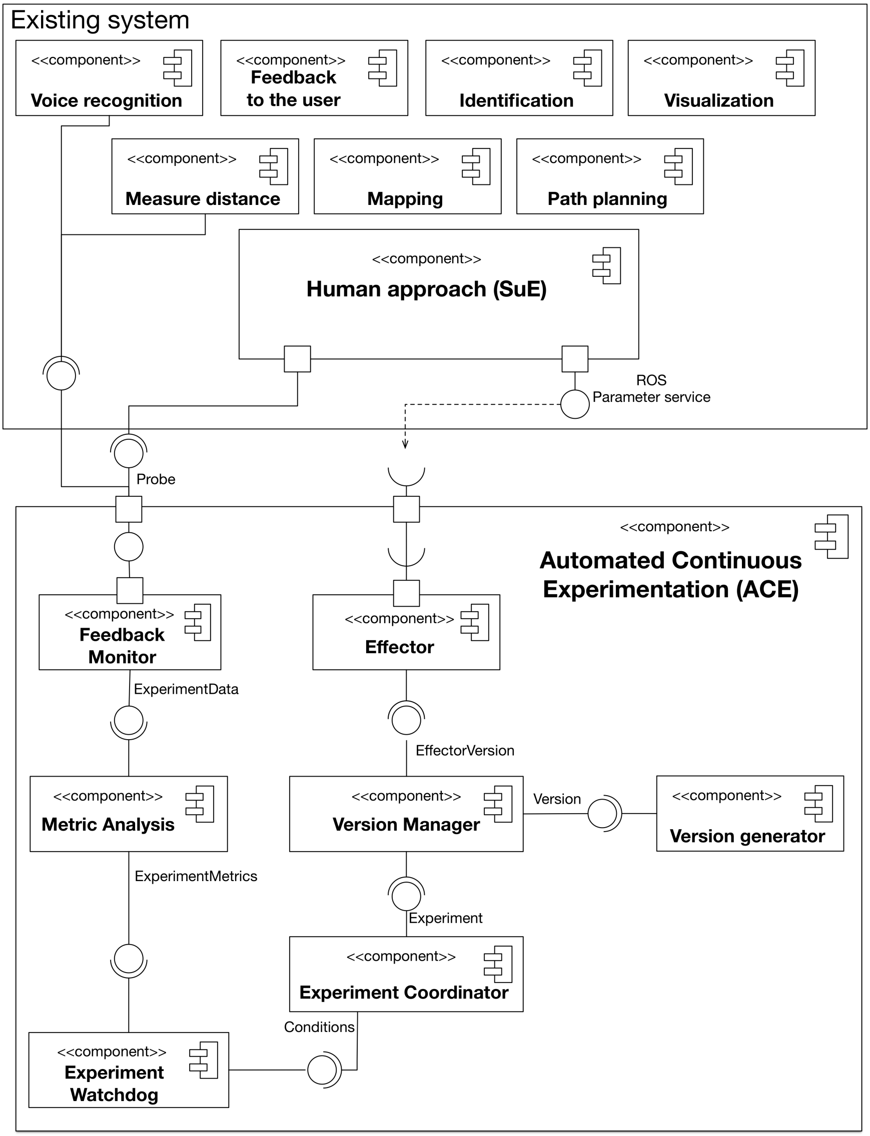

Figure 5.3: Instatiated version of the Automated Continuous Experimentation architectural framework in the case of a human-robot proxemics distance problem.

with the k-means++ initialization algorithm implemented in the scikit-learn [3] Python library. This learning algorithm is a particular solution of the k-armed bandit optimization problem.

Effector: this node is responsible for interacting with the human approach feature and modify in runtime its internal value for approaching. In the ROS context, this means changing parameters in the parameter server. The

[3] `http://scikit-learn.org/`

implemented architecture framework is completely external to the system. The system runs normally without the framework and even with if the instantiated framework crashes.

5.7.2 Experimental results

In this subsection, we describe the experimental conditions which we tested in this system.

The robot was placed in an office environment. The participant was instructed to give verbal (speak), non-verbal feedback (stepping forward or backward), or both, to the robot if they think the robot is too far or too close (feedback value -1), and not to give any feedback if they think the robot is at a comfortable distance (feedback value 1). The robot always moves to a different location before approaching again. The participant did not have any previous experience with autonomous robots or knowledge of the internal system of the robot. The participant also didn't know that the robot was learning from the given responses.

Situations, as when the system goes out of the defined boundaries, were tested manually during the experiment (by explicitly sending the system to a new state) without the knowledge of the participant. All cases were handled by the stop experiment module without any problems.

Fig. 5.4 shows three graphs representing the learning process of the system experimenting with different uniform random approaching distances. The first graphic shows the system exploiting the solution space trying different distances. The second graphic shows the system learning and refining the lower and upper distance boundaries using the clustering algorithm. The third graphic shows the system learning a static distance after several robot approaches. The static learned distance is the centroid of the cluster located between the boundaries. This graphic shows an online optimization through experiments on the proxemics distance. Changes in the user behavior are reflected in the learning process. Although this experiment shows only one experiment in an office environment, this could be extended to incorporate different factors that influences the robot behavior, such as incorporating the type of room in the experiment, or identifying different use scenarios. This would allow the robot to continuously experiment and learn new distances in different contexts, therefore increasing the value it delivers. This experiment suggests a more effective way of experimenting compared to manual experimentation. Each experiment requires time to run and set up and is valid in restrictive conditions. Although this experimenting method does not guarantee optimally, manually experimenting a matrix of solutions can be expensive in both terms of set-up cost and time.

5.8 Conclusion

Experiments in the field are used in a problem-solving process to drive both innovation and optimization of post-deployed systems. Companies are moving towards to experiment-based development, where experiments support the decision-making process. Several challenges, such as resources, the experiment architecture and novel engineering approaches arise when running experiments

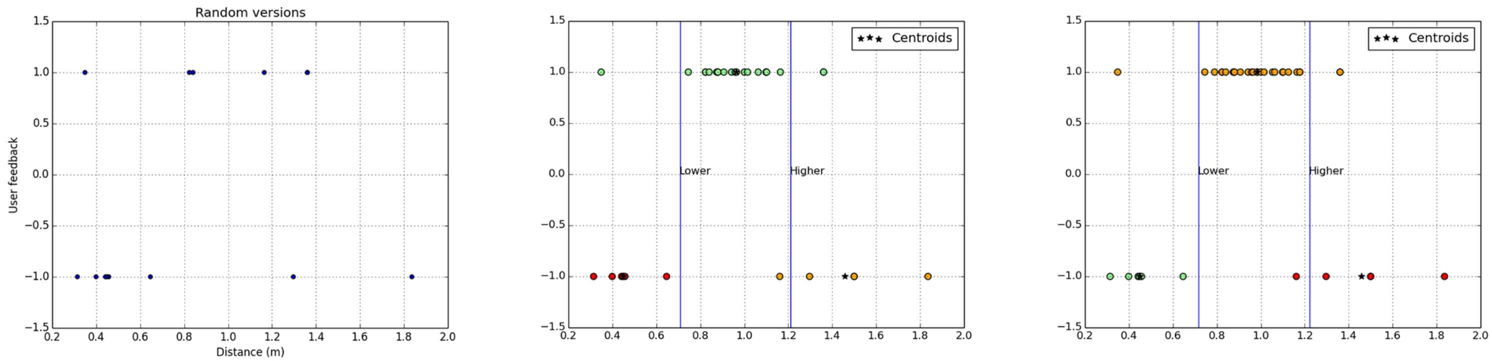

Figure 5.4: Three graphics representing the learning of the system. The first graphic shows the robot exploring the feedback-distance space. The second shows the robot learning an interval and refining it. The third shows the system after achieving a validated static value for the distance. This distance is represented by the centroid in the orange section.

in a large scale. In this paper, we study automated experimentation to address these challenges.

This paper takes the software architecture perspective to understand and develop an initial architecture framework that supports both hypothesis validation and for optimization through experiments. Different architecture qualities are identified and it is proposed an initial architecture framework to support automated experiments.

The architecture framework is validated using an autonomous mobile robot establishing the optimal human-robot proxemics distance. The robot implements an optimization solution based on the explore/exploitation problem running automated experiments. The experimental validation not only assesses the correctness of the framework behavior but also suggests that this is a cost-effective way to run experiments.

5.8.1 Research Challenges

Automated experiment systems still have a long way to go before they are matured. Different domains develop solutions and algorithms that continuously push systems in the conceptual solution proposed in [26]. However, these different domains solve their experiment-specific problem without an unified view over the experimentation process itself. This work brings together different facets of automated experimentation and proposes a framework to support automated continuous experimentation. The framework was validated experimentally in the context of an autonomous system and is currently under validation in the mobile domain and in web-systems. One research challenge is to evaluate this and other architecture frameworks in different contexts. This will bring into perspective the challenges from the different domains that might reflect in both the desired architecture qualities and in the architecture framework.

A second research challenge is to combine manual experimentation with automated experimentation in R&D teams as part of the development process for the product. As much as the culture for experimentation changes the organization perspective [36], automated experimentation can change how the systems are developed.

A third research challenge in the process of fully automate experiments is the ability to formulate hypothesis automatically from the data. Manual experimentation requires significant effort for the hypothesis formulation from analyzed data. The proposed framework automates how to run and evaluate experiments, but it still does not support hypothesis generation. Creating hypothesis automatically from recorded data can leverage the amount of experiments the system can run and generate deeper insights on how the system works in the uncertain environment. This would allow the experiment innovation perspective to be an integral part of the system.

Concluding, although there are several challenges ahead, we are moving to a future where systems get better every day we use them through automated experimentation.

Acknowledgments

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Chapter 6

Paper B

Automated Optimization of Software Parameters in a Long Term Evolution Radio Base Station

Mattos, D. I., Bosch, J., Olsson, H. H., Dakkak, A., Bergh, K.

Annual IEEE Systems Conference (SysCon), 2019, pp. 1-8

Abstract

Radio network optimization is concerned with the configuration of radio base station parameters in order to achieve the desired level of service quality in addition to many other differentiating technical factors. Mobile network operators have different physical locations, levels of traffic profiles, number of connected devices, and the desired quality of service. All of these conditions make the problem of optimizing the parameters of a radio base station specific to the operator's business goals. The high number of calibration parameters and the complex interaction between them make the system behave as a black-box model for any practical purpose. The computation of relevant operator metrics is often stochastic, and it can take several minutes to compute the effect of changing a single, making it impractical to optimize systems with approaches that require a large number of iterations. Operators want to optimize their already deployed system in online scenarios while minimizing the exposure of the system to a negative set of parameters during the optimization procedure.

This paper presents a novel approach to the optimization of a Long Term Evolution (LTE) radio base station in a large search space with an expensive stochastic objective and a limited regret bounds scenario. We show the feasibility of this approach by implementing it in an industrial testing bed radio base station connected to real User Equipment (UE) in collaboration with Ericsson. Two optimization processes in this experimental setup are executed to show the feasibility of the approach in real-world scenarios.

6.1 Introduction

Delivering software that adds value to customers is an important aspect for the success of every company. Software calibration parameters can have a direct impact on how customers use the system and the direct value features deliver to customers [48, 141]. As software systems grow in size and complexity they often have an increased number of available software calibration parameters [142]. As an example, the number of calibration parameters that can be optimized in radio base station (RBS) software and in the mobile network is in the range of thousands of parameters and it is proportional to the number of cells in the network.

The interaction between a subset of these parameters and the response of the system towards the customers' needs is often too complex to model and hence the system behaves like a black-box model. The responses of the customers and users are stochastic regarding changes of parameters and often the software system needs several minutes to compute a response to a new set of parameters. At the same time, the customers expect the software system to perform as well as it did when it was acquired initially; in other terms the system should minimize the regret associated with the exploration of parameter space [143].

At the same time as the optimization of the parameter space can deliver value to the customers, the design of such an optimization system needs to assure that the optimization software integrates and is able to optimize the system prior to customer deployment and during operation.

One of the critical challenges facing mobile network operators is how to optimize the parameters of a deployed radio base station to deliver better services to customers, by providing quality of experience (QoE) for a range of different applications, such as voice over LTE, uplink signal, video traffic, web browsing and online gaming. Therefore, the optimization procedure for the mobile network should consider how well the business goals of the operators are achieved.

Together with other parameters such as physical location and environment, levels of traffic, and number of connected devices, the optimization of a mobile network should address the individual optimization of each RBS. This type of optimization procedure cannot be conducted realistically in simulations or in laboratory settings.

The current state of practice of optimization network parameters consists of manually tuning the network based on pre-determined patterns and the expected changes from mathematical models. Zhang [144] describes several different procedures for optimizing an LTE network. However, as mentioned above, the current available tools focus on a small number of parameters and require a lot of manual effort from experts. For example, Awada et al. [145] describe the Taguchi method that requires creating grids of orthogonal arrays that need to be manually created and executed before serving as input for the next iteration.

Research on network optimization is evolving and introducing new methods and approaches to facilitating the optimization of a network after deployment and managing changes in the network, such as the expansion of the network with new RBS and the future integration of 5G technologies. However, the state-of-the-art research in optimization is based on specific radio parameters that are pre-optimized in simulations [145–150]. These approaches cannot accommodate customized business metrics of the operators and are not suitable for using in live RBS.

The contribution of this paper is three-fold. First, it describes the online optimization of an RBS as the optimization of an expensive stochastic objective and limited regret bounds. To the knowledge of the authors, this is the first paper to propose an approach for mobile network optimization given this set of restrictions. Second, this paper implements the proposed approach in a testing bed configuration with a deployed radio base station connected to real user equipment (UE). Two empirical experiments are shown, demonstrating the feasibility of the approach. Third, the paper provides guidelines of how the presented approach can be incorporated and be used in both a testing bed and in customer deployment real live networks.

The rest of this paper is organized as follows. Section II presents an overview of an LTE radio base station, LTE optimization, parameters and metrics of interest and an overview of the online optimization problem. Section III describes the experimental setup, the optimization system and the integration of this system with a radio base station in a testing bed configuration. Section IV presents the empirical results of the two optimization procedures, discusses how the optimization system can be integrated in a deployed radio base station, before considering the limitations of this approach. Section V presents and discusses related work in radio base station optimization. Finally, Section VI concludes the paper.

6.2 Background

6.2.1 Overview of a Radio Base Station

A cellular mobile network consists of a core network and a radio access network. The radio access network is responsible for connecting the UE (e.g. a mobile telephone) to the core network, which is connected with the internet through a gateway. The radio access network is composed of a collection of radio base stations (RBS).

An RBS is an installation that provides the uplink and downlink communication with the UE. Each RBS divides its coverage region space into cells. The separation region between each cell in the same RBS or between cells of different RBS is called the handoff zone. The collection of cells defines the finite area of coverage of the radio access network. The RBS implements the different radio standards in order to connect to UE, such as 2G (GSM), 3G (WCDMA), 4G (LTE). Although the RBS can be studied and viewed from multiple perspectives, this work focuses on the software aspect. In this work, we describe an approach to the optimization of the calibration parameters of a single 4G/LTE RBS.

6.2.2 LTE Optimization Overview

The optimization of a mobile network is a necessary process for ensuring that key performance indicators (KPIs) established by the operator are satisfied or improved. The optimization of an LTE mobile network consists of two stages: the first is the pre-launch optimization of the LTE procedure and the second is the continuous optimization of the network after deployment [144].

The pre-launch optimization changes are mainly physical, such as the antenna tilt and azimuths, but can also include some software parameters [144]. In the pre-launch, the operator does from coarse-tuning operator independent optimization to fine-tuning optimization based on counters and KPIs tailored to the operator needs. Post-launch optimization procedures also include fine tuning to accommodate the expansion of the network with both existing technologies as well as incorporating and optimizing the network for new ones such as 5G.

6.2.3 Radio Base Station Parameters and Metrics

Although mobile network operators can define their customized KPIs and metrics tailored to their business objectives, this subsection presents a small subset of counter-based RBS metrics that can be used in the optimization process, and presents the parameters that can be optimized.

6.2.3.1 Metrics

Random-Access Success Rate (RASR) When a UE tries to connect with an RBS it needs to get synchronized in the uplink. This is done by having the UE send a random-access channel preamble message (Msg1) to the RBS. The RBS responds with a random-access response message (Msg2) [151]. The random-access success rate refers to the rate of successful connections compared to the number of connection attempts. The random access has an impact on the average setup call time and can have a larger impact in networks with a high number of devices, such as internet-of-things and machine-to-machine communication. Some parameters that can influence the RASR are preamble initial received target power, maximum random-access transmission power and power increment steps.

6.2.3.2 Parameters

The parameters investigated in this work are related to power control. Proper power configuration reduces the interference between cells, the power consumption of UE in the uplink, and uplink throughput, among others.

Maximum Transmission Power This parameter controls the maximum transmission power. To modify this parameter, the cell that is being transmitted should be stopped, preventing transmission while changing the parameter, therefore affecting the traffic. This represents a case where the optimization is expensive not only in terms of time to collect data but also how it can affect the users' traffic in the network.

Cell Range The cell range parameter defines how far the cell coverage will extend. This is an important setting that defines the boundaries between different cells and the geographical coverage extend of an RBS. The transmission does not need to be halted for the change of this parameter. However, the optimization of this parameter is also expensive as it directly affects traffic in the hand-off procedures.

6.2.4 The Online Optimization Problem

The online optimization of software parameters problem is concerned with selecting the best set of software parameters that maximize an objective while it is running. Compared to the offline scenario, where optimization is carried out on a model of the system, online optimization poses several restrictions:

The system must have interfaces that allow the change of software parameters without stopping the execution of the system. Alternatively, the system should have predetermined time windows where it can stop its execution and is allowed to change the parameters.

The metrics and the objective of an online system take time to be calculated and modifications of the parameters cannot be evaluated instantly. This can introduce delayed feedback if the parameters are changed in a high frequency.

The optimization algorithm should balance the exploration-exploitation of the parameter space, with limited regret bounds on the exploration. As the system serves real users every change that serves suboptimal parameters to the users and degrades the objective metrics has associated regret [143].

The objective function is often not easy to model in terms of parameter functions. It represents complex interactions with users and can capture complex and abstract business goals [43]. Its measurement can be stochastic with a high unknown variance.

The first and second restrictions impose practical limitations not observed in offline optimization methods and in the relatively simple models discussed in most research [145, 147]. The second restriction makes the optimization of a system an optimization problem with an expensive objective function, as the system cannot run multiple parallel versions and instantly evaluate the optimization objective. This restricts the type of algorithms that can be used in the optimization process, as it is unfeasible to run several thousands of iterations for convergence if the objective takes several minutes to be updated.

The third and fourth restrictions limit the test to algorithms that are able to converge under the stochasticity of the objective function and that balance the exploration-exploitation trade-off of the parameter space [52, 53, 143]. This scenario corresponds to an optimization problem in a large search space with an expensive stochastic objective and limited regret bounds.

The problem can be mathematically expressed as finding the global maximum of an unknown objective function with stochastic response $f(\boldsymbol{X})$ in the compact subset space $\chi \in \mathbb{R}^d$, where d is the number of parameters being optimized [53].

$$f(\boldsymbol{X}) = L(\boldsymbol{X}) + \epsilon(\boldsymbol{X}) \tag{6.1}$$

$L(\boldsymbol{X})$ is the unknown true value of the metric we are optimizing for, that that only observable through the stochastic function $f(\boldsymbol{X})$ and $\epsilon(\boldsymbol{X})$ is a stochastic variable.

$$\boldsymbol{X}^* = \arg\min \max_{\boldsymbol{X} \in \chi} f(\boldsymbol{X}) \tag{6.2}$$

Based on the observations output $y \in \mathbb{R}$, where:

$$\mathbb{E}[y|f(\boldsymbol{X})] = f(\boldsymbol{X}) \tag{6.3}$$

Cumulative regret refers to the sum of all simple regret observed during the optimization procedure, where the simple regret corresponds to the difference in response between the unknown theoretical maximum and the actual observation. The cumulative regret for this problem is represented by:

$$R = n \cdot f(\boldsymbol{X^*}) - \sum_{t=1}^{n} y_i \tag{6.4}$$

It is assumed that f(X) can be computed for all values of $\boldsymbol{X} \in \chi$ in a limited amount of time, but not necessarily computed instantly. The variable n represents the number of observations of y, where n is a small number of observations (compared to the problem constraints and the space χ).

The algorithm is responsible for finding the optimal set of parameters within n observations while minimizing the expected cumulative regret. This ensures a balance of the exploration-exploitation trade-off.

Research on expensive optimization [53] suggests classes of algorithms derived from Bayesian optimization and hierarchical tree search. Bayesian optimization methods have the disadvantage that they require setting up hyper-parameters and selecting an appropriated acquisition function for the specific problem. However, Bayesian methods still perform better than hierarchical methods when prior knowledge of the problem exists and the optimization function helps to determine good acquisition functions, kernels and hyperparameters.

For large complex systems with limited or no prior knowledge, the chosen approach is the usage of hierarchical optimistic optimization algorithms that can handle stochastic values with regret bound guarantees such as the authors' Limited-Growth Hierarchical Optimistic Optimization (LG-HOO) algorithm [29], a modified version of the Hierarchical Optimistic Optimization (HOO) [52].

These hierarchical algorithms are proposed and extensively discussed in [152]. They have mathematical guarantees on the cumulative regret R and depends only on the observations output y. They are designed to balance the trade-off between exploring the solution space and exploiting the best-known solution by building a space-partitioning binary tree by splitting leaves that have a higher upper confidence bound. The root of the tree is usually the current parameter and the exploration of the space starts from this root. The leaves that have a higher confidence bound are expanded and the binary tree grows in the direction of the highest confidence bound. The algorithm balances the exploration-exploitation by selecting parameters that belong to the highest confidence bound branch.

In higher dimensional space, the algorithm expands the leaves by randomly selecting a dimension to split. In the worst-case scenario, where the parameters do not have any influence on the measured objective, the tree grows uniformly in all directions. Even in worst-case scenarios, the algorithm response time provides fast enough results for the usage in expensive scenarios, with response time in the range of milliseconds.

6.3 Experimental setup

This research was conducted in collaboration Ericsson. Ericsson is a Swedish multinational network and telecommunications company. The company provides services in software and infrastructure in information and communications technology, including mobile network infrastructure.

6.3.1 The ACE system

The optimization system called ACE (Automated Continuous Experimentation) was developed in prior work by the authors at Chalmers University of Technology and aims to provide a common application interface for companies and research projects running online and offline optimization procedures via experiments [65, 66]. The systems that are being optimized are called systems under experiment (SuE) and the optimization system will be referred to only as ACE. The ACE system was used in different black-box optimization such as in a mobile application together with Sony Mobile [29] and in a mobile robot [65,66].

ACE consists of multiple services: an optimization backend, a visualization and configuration frontend, non-relational databases and a webserver to handle communication with interfacing systems. The ACE optimization backend implements several algorithms for black-box optimization, including A/B/n experiments [2], multi-armed bandit and contextual bandit algorithms (such as UCB1, UCB2, Softmax, Epsilon-Greedy and LinUCB) [143], χ-armed bandits for optimistic optimization (DOO, HOO and the LG-HOO) [65, 143, 152], and Bayesian Optimization [53]. Based on the restrictions presented in Section II, we utilized the LG-HOO algorithm for the experiments.

ACE is designed to handle and scale multiple simultaneous optimization jobs, allowing multiple simultaneous systems under experiments and user connections, maintaining and tracking data storage and the evolution of the optimization models to allow tracing, and debugging the optimization process at any point. ACE can be instantiated either in cloud environments or local deployment through Docker containers (https://www.docker.com/) and can communicate with the SuE through HTTP requests with JSON payloads. This ensures that ACE and the SuE can evolve and be deployed separately. One of the reasons for this design decision is that incorporating optimization code in existing and legacy systems can significantly increase the cost of the system by increasing the cost of maintaining new optimization software for every new product, introducing new costs with re-verification, re-testing, re-validation and re-certification of existing systems. Therefore, ACE only modifies parameters within a predetermined and validated range.

6.3.2 Testing bed setup

Testing bed optimization of software parameters is a common method in the development of control systems in industry. Often the tuning of these parameters becomes a trial-and-error process that is both tedious and time-consuming, but still requires expert knowledge [153].

This method helps to bridge the gap between the optimal parameters of the simulation model and those of the final product. We utilize an RBS testing

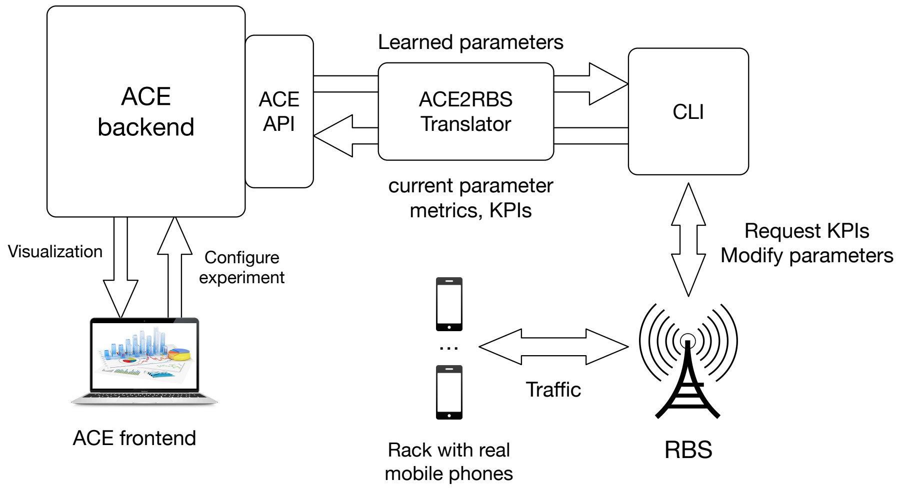

Figure 6.1: Schematic representation of the testing bed experimental setup. The ACE frontend systems communicates with the ACE backend to allow remote configuration of the optimization procedure and visualization of the experiment results. The ACE backend is deployed in a separate computer or cloud infrastructure utilizing Docker containers and exposing tcp ports in an internal network to the equivalent of operator system that controls the RBS. In the operator computer the ACE2RBS translator communicates between the ACE system and the command line interface software that controls the RBS. The CLI system and the communication between the CLI and the RBS are the same as the deployed in live RBS. The utilization of the commercial interfaces used by the operators allow the system to be integrated in production environment without modifications in the deployed software of the RBS.

bed setup as both a validation process of our optimization approach, and as an automatic way to fine tune some general parameters before deployment.

The RBS used in the optimization procedure is part of the Ericsson mobile LTE network test bed. The RBS is connected to a rack of multiple mobile phones. Although these mobile phones are not driven by real users the test bed generates relevant traffic scenarios for these mobile phones that can stimulate and test the RBS in realistic scenarios. Although the traffic is generated in a predicted way at every test run, it can be considered stochastic inside each run, as it would be with traffic generated by users.

The RBS has different interfaces where the operator can modify and customize the behavior of the RBS. One of the interfaces that operators use is a command line interpreter software (CLI) that allows operators to easily access statistics and KPIs and modify parameters of multiple managed objects.

The interface between the ACE and the RBS is done through translator software called ACE2RBS, which converts ACE responses into CLI commands. The ACE2RBS is also responsible for polling the KPIs of interest together with the set of parameters that generated the KPIs. Fig. 6.1 shows a schematic of the experimental testing bed setup.

The metrics generated by the RBS can be collected at any time but are more stable in fifteen minutes cycles. The metrics are stochastic regarding the parameters as the connected mobile equipment perform different tasks and follow different traffic routines. Due to the costs of carrying out this expensive optimization scenario under uncertainties $n = 100 \cdot m$ samples, where m is the number of parameters being optimized, were chosen per optimization procedure

The ACE2RBS translator software can request the metrics from the RBS and can modify the parameters described in Section II.C.

6.4 Results

This section describes the two different experimental runs in the testing bed setup, how the system can be integrated in a deployed RBS and the limitations of this approach. The first experimental run consists of an optimization procedure for one parameter while the second case shows the optimization procedure for two parameters simultaneously.

6.4.1 Experimental runs

The results presented in this section were normalized to the range $[0, 1]$ and therefore corresponds to a different unit system in to hide the real values used inside Ericsson test beds.

Optimization of the Random-Access Success Rate (RASR) based on Maximum Transmission Power

In this experiment, the χ space corresponds to the maximum transmission power in the range of 0 to 1 units of power, and the objective functions are measurements of the RASR. The default value of the test bed RBS is 0.5. The aim is to maximize the RASR by observing how the transmission power affects it within the predetermined range.

The result presented in Fig. 6.2 indicates that the tree of the LG-HOO algorithm is growing uniformly in the space. A total of measurements of 95 samples were taken, and as by the design of the hierarchical optimistic optimization algorithms, the same parameters set could be sampled multiple times. It can be concluded that the transmission power in the range of 0 and 1 units of power has a small influence in the RASR variance in the used testing bed setup. However, a maximum transmission power of 0.8125 had a higher RASR. The normalized effect size of this maximum transmission power compared to the default settings (maximum transmission power of 0.5) is 24.2%, i.e. the optimization procedure increased the RASR metric by 24.2% in this optimization scenario.

Note that the obtained result is not generalizable for other cases, as it is a tuning procedure specific to the experimental conditions of the deployed RBS and testing procedure.

6.4.2 Optimization of multiple parameters

This experiment consists of changing two factors in the χ space, the maximum transmission power and the cell range. The default value for the testing bed RBS is 0.5 for the maximum transmission power and 0.5 for the cell range.

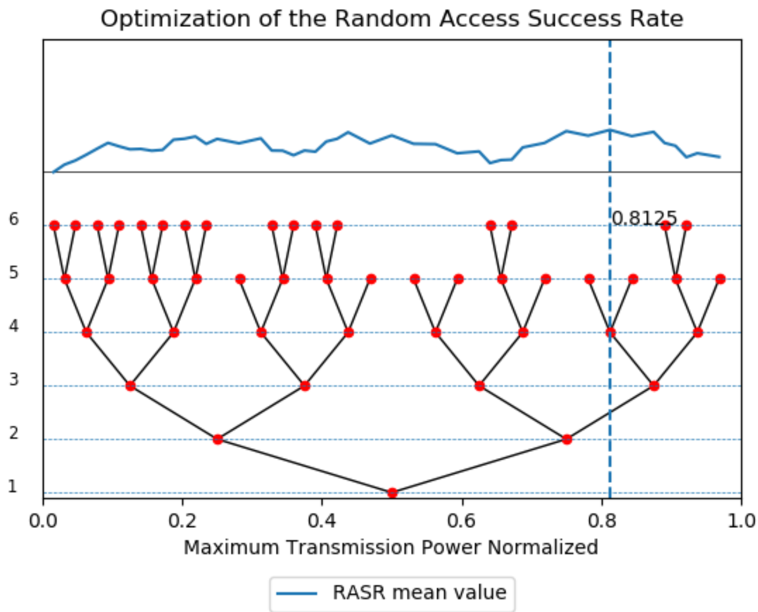

Figure 6.2: This figure shows the tree of the maximum transmission power with the Random Access Success Rate metric. The line plot shows the normalized RASR mean value for each node.

This experiment measures the effect of the RASR and aims to maximize it changing these parameters.

The results of this experiment are showed in the three pictures of Fig. 6.3. A total of measurements of 200 samples were taken. The first graphic shows how the tree grew in function of the RASR measurements looking only the maximum transmission power, but the measurements consider both factors. This graphic shows that the tree is growing in a similar manner as the first experiment. The second graphic shows the tree for the cell range. The tree grows towards higher cell range. However, the best value for the whole optimization procedure is not in the same places as if it was considering independent optimization procedures. This result shows that both parameters have interaction effects in the RASR for this experimental scenario. The vertical lines represent the best parameter set considering the whole optimization procedure. The third graphic shows how the space was sampled in the optimization procedure, the size of the circles represent the mean effect of the parameter set and red circle represents the best parameter set obtained parameter obtained from the optimization procedure. The normalized effect size of the best parameter set (maximum transmission power = 0.3750 and cell range = 0.5625) compared to the default settings (maximum transmission power of 0.5 and cell range = 0.5) is 46.31%, i.e. the optimization procedure increased the RASR metric by 46.31% in this optimization scenario.

It is worth noting that this was the best set of parameters that were obtained in the limited and constrained sample size. However, other sets of parameters have a mean RASR close to the best set. If the experiment was allowed to run for a longer period and increased the sample size the best set of parameters could change as more confidence on the upper bounds are built.

6.4.3 Integration in a deployed RBS

The presented testing bed approach can be used as demonstrated in the integration with a deployed RBS, for a simple online optimization procedure. However, a few considerations might be required for a more robust implementation.

The optimization procedure depends on the operators' objectives and business goals. For a correct optimization procedure, the metrics should be validated and aligned with the business goals; otherwise the optimization procedure might lead to a suboptimal business while having optimal parameters for a different construct.

The translator software utilizes one of the interfaces available to the operators; the CLI software. The decision to utilize this interface with the RBS was made in order to have lower implementation effort for this research project. For the deployment of a more stable and robust solution other RBS interfaces might be more appropriate.

6.4.4 Limitations

The proposed approach represents a small subset of the possible optimization cases in an RBS and presents a few limitations.

First, we considered an RBS as an isolated system that we want to optimize. The deployment of this system in an access network will require an optimization

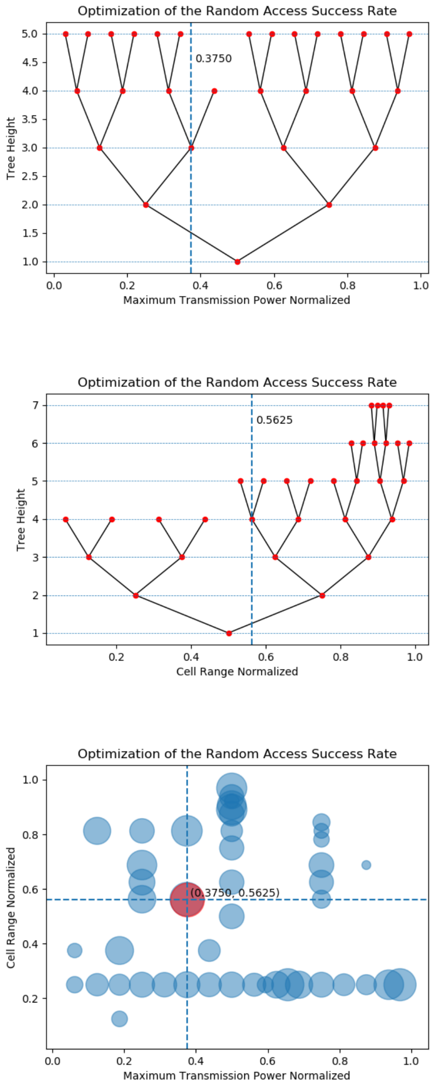

Figure 6.3: Results of the second experiment, where the Random Access Success Rate is observed by changing the cell range and the maximum transmission power. The first graphic shows how the tree grew in function of the RASR measurements looking only the maximum transmission power. The second graphic shows the tree for the cell range. The third graphic shows how the space was sampled in the optimization procedure, the size of the circles represent the mean effect of the parameter set and the red circle represents the best parameter set obtained parameter obtained from the optimization procedure. The fact that the best set does not correspond to the highest node in each tree indicates the presence of interaction effects between the cell range and the maximum transmission power in the Random Access Success Rate for this experimental setup.

of metrics of the whole access network. Even with the individual optimization of every RBS, due to interaction effects in the network, network metrics can still not be performing optimally. One way to allow individual optimization of multiple RBS with a common objective is to use contextual expensive optimization, where each RBS is represented through contextual information [154].

Second, testing analysis and optimization of an access network might prove an expensive process for most operators. In this situation, alternative methods such as simulations and logged traffic data analysis datasets, and alternative algorithms that do not consider regret, might have a competitive advantage over the presented method, if the time constraints of the simulations and analyses are feasible, and the simulation precision is realistic enough.

Third, the existing and the proposed approaches do not incorporate information regarding the operators' traffic profile and requirements. For most operators any degradation of traffic during regular operation can lead to significant negative impact in the whole network. This restriction reduces the operational window that an experiment can run. In such situations, the experiment needs to be divided in different steps. During periods of reduced traffic or maintenance, it is performed an exploration step for exploring new sets of parameters and for building confidence over them. During regular traffic it is performed the exploitation of the sets of parameters that have confidence to perform better than the current.

6.5 Related work

Awada et al. [145] propose an approach to optimize LTE-A (Long Term Evolution-Advance) radio network parameters using the Taguchi method. The Taguchi method is a robust design method, that has been used in different fields to promote improvements in product development. The advantages of the Taguchi method are its scientifically disciplined method used to explore the search space and the fact that procedure is hyperparameter free. The number of experiments without replication depends on the number of factors (parameters) and the number of levels (discretization of the parameter) per factor. However, the Taguchi method has several disadvantages [155]. First, the presence of a stochastic response requires multiple replications for each iteration, resulting in an inefficient method. The method does not place upper bounds on the regret and working well only with a small number of parameters, as the number of experiments without replication increases exponentially with the number of factors. Third, interactions are confounded in the Taguchi's methods, making it hard to utilize in higher order experiments [155]. Although the paper optimizes jointly multiple cells, the optimization procedure is done independently (one parameter at a time) and the proposed approach is only discussed and used with simulation models.

Dastoor et al. [147] compare different algorithms for the problem of parameter optimization in LTE-A networks. They compare four algorithms with a simulated model for mobile communication. The compared algorithms are Genetic Algorithms, Particle Swarm Optimization, Simulated Annealing and the Taguchi method. With the exception of the Taguchi method, which was discussed above, the other algorithms require a high number of iterations together with the population size, making these methods impractical to use in expensive optimization scenarios such as the deployed and testing bed scenarios. Additionally, some of these algorithms also require good tuning of hyperparameters for good optimization and convergence.

Huang et al. [156] propose a framework called DINO (data-driven network optimization), to optimize a large group of RBS. The proposed method is a combination of an offline artificial neural network prediction model, an online clustering, and a network optimization method to distribute the load. Although the problem solved with the DINO framework is different from the problem addressed in our work, the DINO framework suggests that alternative methods to ours can be used if enough quality data traffic records are available to train the neural network. Our approach does not require previous data to start the tuning process, or an optimization model of the system, but relies on a reinforcement learning process in a black-box model.

Döttling and Viering [146] propose the concept of self-optimizing networks. The presented concept is similar to the continuous application of the method presented in our work in deployed systems. However, their approach suggests that the self-optimization feature should be part of the RBS software instead of the external approach we used with the ACE system. In terms of algorithms, the work suggests classes of algorithms such as genetic-algorithms and fuzzy systems that could be used but does not provide further justification for the selection of a particular class of optimization algorithm.

Gerostathopoulos et al. [49, 50] discuss the problem of field and self- optimization using tunable parameters in a semi-formal notation. They model the system under experimentation SuE as a black-box system. However, they utilize a gaussian process followed by a factorial ANOVA is used to analyze and compare the optimized results. The system is evaluated in a simulated traffic routing system and in just in time compiler in a Java virtual machine. For the reasons previously discussed we adopted hierarchical methods instead of the Bayesian optimization procedure.

6.6 Conclusion

This work describes a novel approach to the optimization of software parameters of an RBS. The paper shows how to optimize an RBS online, with regret minimization and a low number of iterations in the presence of uncertainties due to the stochastic response of KPI metrics. This approach allows operators to optimize their custom metrics and KPIs that are aligned with their business goals without the need to model the complex interaction between the parameters and KPIs and to restrict their optimization efforts to only a small subset of optimization scenarios. An automated optimization system can be integrated as part of the deployment functionality and let the system performs as a self-optimizing network in the pre-determined scenarios of interests for the mobile operators.

In this research project, we utilized the ACE system, a system that runs black-box optimization methods for expensive stochastic objectives, integrated with an RBS in a testing bed scenario. The RBS is connected to a rack with several real UEs generating traffic according to test case patterns. Three different optimization experimental runs are presented and discussed.

The proposed approach can be launched and used to optimize an RBS with small manual effort and without the expenses of a mobile network optimization expert. This approach could significantly reduce the effort needed to optimize a mobile network when expanding the network and introducing new technologies such as 5G, where optimization expert knowledge and guidelines will be limited.

In future work, we plan to expand the proposed approach to an online RBS with user traffic and to investigate new procedures to balance the exploration-exploitation tradeoff that aligns with operators' requirements such as re-balancing the exploration-exploitation tradeoff to fit the operators' traffic profile.

Acknowledgments

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and by the Software Center. The authors would also like to express their gratitude for all the support provided by Ericsson.

Chapter 7

Paper C

An activity and metric model for online controlled experiments

Mattos, D. I., Dmitriev, P., Fabijan, A. Bosch, J., Olsson, H. H.

International Conference on Product-Focused Software Process Improvement, 2018, pp.182-198.

Abstract

Accurate prioritization of efforts in product and services development is critical to the success of every company. Online controlled experiments, also known as A/B tests, enable software companies to establish causal relationships between changes in their systems and the movements in the metrics. By experimenting, product development can be directed towards identifying and delivering value. Previous research stresses the need for data-driven development and experimentation. However, the level of granularity in which existing models explain the experimentation process is neither sufficient, in terms of details, nor scalable, in terms of how to increase number and run different types of experiments, in an online setting. Based on a case study of multiple products running online controlled experiments at Microsoft, we provide an experimentation framework composed of two detailed experimentation models focused on two main aspects; the experimentation activities and the experimentation metrics. This work intends to provide guidelines to companies and practitioners on how to set and organize experimentation activities for running trustworthy online controlled experiments.

7.1 Introduction

Prioritizing the development of software features and services that deliver value to customers is critical for the success of every company. One way to accurately discover what customers value is to evaluate the assumptions of the company by means of experiments. These experiments, commonly called A/B tests, provide a framework for companies to establish causal relationships between modifications on their systems and changes in metrics. Running experiments allows companies to continuously up-date their assumptions on their user behavior and preferences, along with many other benefits [72].

Several publications and reports from companies such as Microsoft, Facebook, Google, and LinkedIn, among many others [13, 18–20], report the competitive advantage that online controlled experiments, such as A/B testing, deliver [72]. Data-driven organizations make use of relevant collected data to drive decisions and directions for the organization, and experiments are one of the key techniques used by these organizations. However, the support and evolution of experimentation practices is far from simple, and several pitfalls can invalidate experiments and lead to incorrect conclusions [157].

Different models proposed in the literature [5, 7, 48] provide a general structure for data-driven development and experiment processes. Although these models can be used as a starting point for companies to move to an iterative experiment-driven development process, previous research [2, 21, 43, 157–160] also describes pitfalls, techniques to provide scaling of the experimentation process and techniques to ensure trustworthiness in the experimentation process that are not captured in or represented by the higher level of abstraction provided in these models. Because these models do not capture this level of detail, instantiating these models directly from a higher level of abstraction can lead to the limitations in the scalability and trustworthiness of the experimentation's activities already identified by previous research. Additionally, it can lead to multiple experimentation initiatives inside an organization and lack of rigor in the process, resulting in non-comparable tests and untrustworthy results.

To address this gap, this research provides a framework that captures specific experimentation details and necessary steps for running trustworthy online controlled experiments. The framework divides the experimentation process into two main inter-connected models: the set of activities that organizations should support to run trust-worthy experiments, and the role of metrics and how they align experiments with long-term business goals. The proposed framework is based on an inductive case study in collaboration with the Analysis and Experimentation team at Microsoft. The contribution of this paper is twofold. First, we present the new findings from the case study. These findings represent important characteristics of the experimentation process that were not captured in previous models and reinforces the need of a new experimentation process model. Second, we present a framework composed of two models for an experimentation process that covers the two main aspects: (1) the experimentation activities and (2) the experimentation metrics. The framework provides a detailed process which aims to help companies scale their experimentation organization with a trustworthy experimentation process.

The rest of the paper is organized as follows. Section 2 provides a background in controlled experiments and related work. Section 3 describes the research process of this case study. Section 4 presents new findings from the case study that reinforce the need for a new experimentation process model. Section 5 presents the two main aspects of the developed experimentation process framework. Section 6 concludes this paper.

7.2 Background and related work

Although there are many different techniques for learning about customer preferences and using them to evaluate ideas (e.g. interviews, focus groups, observational studies, prototypes and mock-ups) [6], online controlled experimentation is gaining significant momentum in software companies [4]. Controlled experiments are a group of techniques where users are randomly assigned to two or more variants of a product: the control (e.g. the current system) and the treatments (the system with change X). The change could be the addition of a new feature or the modification of existing functionality. The system is instrumented and key metrics of the user's behavior are computed. After a pre-determined period of time, the metrics are analyzed. If the only consistent difference between the experiment variants is the change X and external factors are spread out evenly between the two variants, the differences in the metrics are due to the change X. Based on this statistical analysis, companies can make data-driven decisions. Kohavi et al. [2] provide a detailed guide on running controlled experiments on the web.

Gupta et al. [161] describe the software architecture of the Microsoft ExP Platform, the design decisions made while designing the platform, and its main components. This platform and its components capture essential steps and activities that enable trustworthy experiments on a large scale. Kevic et al. [45] analyzed the results of over 20,000 experiments at Bing, providing an empirical characterization of the experimentation process in a product-running experiment at scale. This characterization shows that the average experimentation process takes forty-two days and includes multiple iterations to minimize the likelihood of hurting users or the business due to issues with the change that is being tested.

However, not all companies and products have the capacity to run experiments at the same scale as Microsoft Bing. Experimentation in software companies typically evolves from a few independent experiments towards a mature stage where several teams run many trustworthy experiments at the same time. Fabijan et al. [32] provide guidance on how to evolve into a data-driven company, exploring the technical, organizational and business evolution. The evolution of experiments in products is divided into four levels of maturity (crawl, walk, run, fly) across three dimensions (the technical, organizational and business). Additionally, the study presents steps and experimentation activities which are commonly used during the evolution of experimentation. One of the key challenges in controlled experiments is how to decide which metrics should be used in the Overall Evaluation Criteria (OEC). The OEC is the experiment objective, which should ideally capture the long-term interests of the company. Determining a good OEC is hard as it captures abstract concepts that are difficult to validate and compare with other metrics [162]]. If the OEC metric captures long-term goals or represent value, true movements of the metric represent the aggregated value that a variant is bringing to the system.

Deng and Shi [44] provide an extensive discussion on metrics for online experiments, classifying the types of metrics, the qualities and characteristics of good metrics, and how to evaluate and select metrics. Dmitriev and Wu [162] discuss a metric evaluation framework at Bing using an offline historical experiment dataset called experiment corpus. The experiment corpus helps to evaluate new metric sensitivity and alignment with user value. This framework helps to select suitable OEC metrics for experiments.

Previous research has provided different models and frameworks that capture and provide guidance on how to develop experiment-driven software. The QCD model (Quantitative/qualitative Customer-driven Development) [48] is an inductive model based on a generalization of approaches used by companies to guide their collection of customer feedback throughout the development process. Experimentation explores the notion of continuous validation of customer value, in contrast to the traditional up-front specification of requirements. The QCD explores this notion by treating requirements as hypotheses that need customer validation at the beginning of the development process. New hypotheses are based on business strategies, customer feedback, innovation strategies and previous hypothesis cycles. Qualitative feedback (surveys, interviews, focus groups and mock-ups), together with quantitative data (feature us-age, customer behavior and support data), allows the evaluation of hypotheses. Hypotheses that are not confirmed through any of the selected customer feedback techniques are abandoned while validated hypotheses can be refined into a more detailed hypothesis or can be implemented and deployed. This model provides a general framework for evaluating hypotheses with customer feedback. This model can incorporate online experiments at a higher level of abstraction. However, it does not provide detailed clear steps and activities for instantiating an online experiment in software systems.

Olsson and Bosch [5] present the HYPEX (Hypothesis Experiment Data-Driven Development) model as an alternative development process for companies to com-press the customer feedback loop. This model advocates for an iterative and incremental development approach, rather than spending engineering effort on larger quantities of functionalities without customer validation. The HYPEX model is composed of six steps: (1) the generation of a feature backlog from customer needs or business goals. (2) Feature selection and specification (what is the intended behavior, what is the gap it addresses, and multiple implementation alternatives). (3) The implementation and instrumentation of a minimum viable feature (MVF). (4) Analyzing whether the measured behavior of the MVF addresses the gap or not. (5) Generation of hypotheses that explain the feature behavior and why the gap was/wasn't addressed. If the gap was addressed new features are selected as in the second step. (6) If the gap was not addressed, alternative implementations can be made (step 3) or a decision to abandon the feature can be made. The HYPEX model is a general model for data-driven development and running product experiments, but it does not provide specific steps for running online controlled experiments.

Fagerholm et al. [7] present the RIGHT (Rapid Iterative value creation Gained through High-frequency Testing) model for continuous experimentation.

The goal of this model is to provide a systematic framework for developing experiment-based software. This is achieved by establishing a series of building blocks that act as pre-conditions for running experiments. These blocks are divided into two main parts: the RIGHT process model and the RIGHT infrastructure architecture model. The RIGHT process model follows the Lean Startup methodology cycle [46]: build, measure, learn. The goal of this cycle is to achieve the vision of the company (which is connected to the business model). This is operationalized through hypotheses generated due to uncertainties in how to execute the vision through the business model and strategy. The set of generated hypotheses is prioritized with the learning of previous iterations. The selected hypothesis is implemented through an instrumented mini-mum viable feature or product (MVF). The collected data from the MVF is analyzed and used to update the assumptions of the business strategy and abandon the tested hypothesis or the data is used to further iterate with the hypothesis by changing or optimizing it. The RIGHT infrastructure architecture model sketches the infrastructure needed to run experiments and specifies the roles and tasks, the technical infrastructure, and the information artefacts consumed and generated during the experimentation process. The RIGHT model was created in a startup environment by two companies starting to run their first online experiments, and it takes the approach of abstracting the underlying details of a continuous experimentation system, in order to be generalizable to a range of different experiments that can be conducted in a startup environment.

The discussed models can be used as a starting point for companies to systematically move to an iterative experiment-driven development process, providing a higher level of abstraction of the experimentation process and describing general activities. However, previous research [2, 21, 43, 157–160] describes pitfalls, techniques to provide scaling of the experimentation process and techniques to ensure trustworthiness in the experimentation process that are not captured and represented by the higher level of abstraction provided in the discussed models. Instantiating these models directly from a higher level of abstraction can lead to the limitations in the scalability and trustworthiness of the experimentation's activities already identified by previous research.

7.3 Research Method

To help companies develop and support their experimentation process and infrastructure models we conducted an inductive case study [77] in collaboration with the Analysis and Experimentation team at Microsoft.

7.3.1 Data collection

The collected empirical data consists of interview notes, white-board drawings, quotes and shared supporting data from nine semi-structured inter-views with an average and median length of thirty-two minutes each. At the time of the data collection, the second author was working with the Analysis and Experimentation team and was the main contact person for the other researchers during the data collection and analysis phases. All the interviews were conducted in the premises of the company by the first author, who was accompanied by the second author when possible. The interviewees were selected by the second author and represent a diverse selection of software engineers and data scientists working both within the experimentation platform and as users of the platform in different product groups.

The interviews were based on a questionnaire containing eight open-ended questions, starting with a short introduction and explanation of the research. The participants were asked to describe their experimentation process and how it is conducted in each product they work with. Next, the participants were asked to compare their own experimentation process and infrastructure with the existing models in the literature and point out similarities and differences. Next, they were asked about what experimentation activities they performed, the time spent on these activities, their relative impact on the experiment reliability, and the other main activities performed by other people in an experiment lifecycle (from hypothesis generation to decision). We organized the interviews by products and by an approximation of the number of experiments the interviewees ran each year. This allowed us to differentiate experiences from products that run experiments on a large scale from products that start and run experiments on a small scale.

7.3.2 Data analysis

Thematic coding was used to analyze the grouped data [80]. Recurring codes, drawings of the experimentation process and references to different parts of the platform architecture and activities helped to formalize the new findings and derive the proposed experimentation framework. For example: descriptions of different tests and analysis to ensure the experiment was properly configured were grouped in the "Pre-quality checks" while the different techniques used to evolve a metric were divided between "Online evaluation" and "Offline validation". We grouped the different experimentation activities in the development, execution and analysis categories. Based on the thematic coding and the reported activities, we compared them with the existing experimentation models to identify the differences from existing models and propose the activity and metric model. Our analysis is based only on data reported by more than one interviewee, and when available in the development of the models we triangulate the data with other research reports by the Analysis and Experimentation team at Microsoft, available at the web link: `https://exp-platform.com/`.

7.3.3 Validity considerations

To improve the construct validity of the study, and prior to the data collection stage, the semi-structured interview guide was applied to a group of two developers from a Brazilian company with experience in A/B testing, known to the first author, and two Ph.D. students in software engineering. This helped to identify potential problems, such as ambiguity in the questions and the explanations. Regarding the external validity process, although our work was conducted with only one case company, our empirical data was collected from experiences of several different products which are running trustworthy experiments at scale as well as only a few experiments per year. This helped to identify and compare trust-worthy experimentation processes at different stages of maturity. Therefore, we believe that our results can be generalized for companies that want to scale their experimentation organization with a trustworthy experimentation process.

7.4 Findings

In this section, we describe new findings obtained from the empirical data that are not presented in previous research. Together with the description of the experimentation process collected during the interviews, these findings reinforced our motivation to develop the experimentation process framework presented in the next section.

7.4.1 Customer feedback is an important source of experimentation ideas.

The first finding from the empirical data is that experimentation ideas, which are later synthesized into experimentation hypotheses, are often inspired by customer feedback, instead of high-level business goals. In this research, we differentiate experiment ideas from experiment hypotheses. Experiment ideas are the first source of potential changes that can be made to systems, and they represent the potential value of a modification. However, often experiment ideas do not represent real value and therefore need to be tested in experiments [163]. Experiment hypotheses synthesize ideas into concrete experimentation scenarios, addressing what the change is, and how it can be implemented and evaluated. Ideas are synthesized into an experiment hypothesis by experiment owners. An experiment hypothesis, after deployment, can measure the real value of the synthesized idea. Developers and product owners often collect experiment ideas using different qualitative feedback collection techniques. Experiment ideas are refined, developed, prioritized and synthesized based on the experiment owners (developers, product owners and data scientists) being convinced of the positive impact for the user and for the key metrics. Another source of customer feedback ideas are differentiator features and user feedback in competitors' open channels. Although experiment ideas can come from business strategies, often they influence the prioritization process of experiments by influencing the metrics.

> *"It is very rare that an experiment comes from the business. Most experiments come from a group of engaged developers willing to code new ideas and run the experiments . . . The ideas for the new features and their experiments are almost always inspired by customers and competition"* — Principal Data Scientist

Existing models such as the HYPEX and the RIGHT model propose that the business goals impose outcomes for experiments, but do not explicitly represent how hypotheses are identified and prioritized. The QCD model proposes business strategies, innovation initiatives, customer feedback and previous experiments as the source of new ideas. However, it does not consider differences between experiment ideas and experiment hypotheses and how they can be further developed in a concrete experiment hypothesis. This insight reinforces the direct and indirect customer feedback as the drivers of new experiment ideas. However, experiment ideas still need to be prioritized and synthetized into experiment hypothesis by experiment owners and engineers.

7.4.2 Metrics guide experiments towards long-term goals and help prioritize hypotheses.

The second finding refers to the role of metrics in learning and in hypothesis prioritization. Experiments are launched with the goal of validating a change in the system or learning more about user behavior. Both the validation of a change and the out-comes of an experiment are closely related to the validity of the metric, whether it measures its concept correctly, and whether it reflects the business strategy of the company. If a metric is misaligned with the business strategy of the company, changes and knowledge gained from experiments will also be misaligned with the strategy. However, correctly chosen metrics create incentive for teams to take actions which are aligned with the long-term goals of the company [162]. Good metrics will help them prioritize experiments that can have a positive impact on the long-term goals.

> *"If our decisions to ship are based on these metrics, these metrics have a big impact on the development and evolution of the product. They guide the teams to develop and focus their work to improve these metrics"* — Principal Data Scientist

The existing experimentation models do not describe or emphasize how metrics impact hypothesis prioritization and long-term company goals. When metrics are considered only part of the instrumentation system, they do not reflect the bidirectional influence on the business strategy of the company.

7.4.3 Metrics evolve and capture the experiment assumptions.

As discussed in the second finding, metrics can guide the long term-goals of the company and help prioritize the hypotheses. Additionally, metrics also capture uncertainty and experiment assumptions. As metrics often represent abstract and subjective concepts [162] such as satisfaction and engagement, they contain assumptions about what constitutes these concepts. These assumptions should be tested and validated during the experimentation process and should be constantly iterated in order to maintain alignment with the business strategy. This constant update and validation of the assumptions based on the results of an experiment leads to a metric evolution process. Metrics can start as low level signals, and then evolve to capture more complex concepts that are more closely aligned with the business strategy. Additionally, the evolution of metrics also reflects the evolution of the business strategy and the product focus over time. As the company changes its strategies, metrics should be updated to align with these changes.

Existing models describe the presence of uncertainty and assumptions in the business strategy and in the role of the business analyst, as they are responsible for hypothesis prioritization. However, these models do not describe or discuss how the metrics evolution and the business strategy influence each other and impact the product.

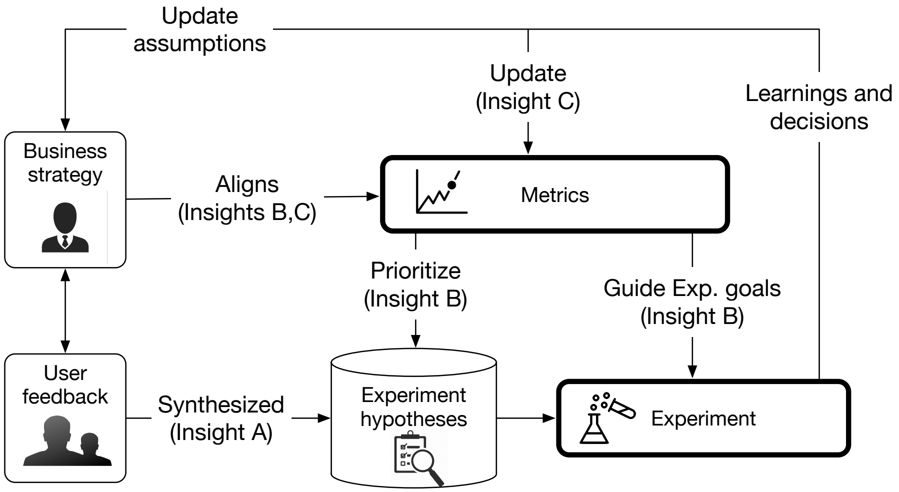

Figure 7.1: The relationship between the two main aspects (in bold), their relation with the business strategy, and how the findings connect them.

7.5 The experimentation process framework

In the previous section, we discussed the findings and compared them to previous research. Previous research did not capture all the identified characteristics from the findings nor present specific experimentation details, or the necessary steps to take when running trustworthy online controlled experiments. This led to the motivation and the need to develop a new framework that incorporates these findings. The framework is based on the collected empirical data, including descriptions and drawings of the process, comparison with other processes and the characteristics identified in the new findings and in the different components represented in the Microsoft ExP Platform described in [161]. During the data collection, the researchers asked the interviewees to describe their experimentation process and compare it with existing models. This discussion, together with findings from previous research, led to the development of this framework, which consists of the two main aspects of the experimentation process: (1) the experimentation activities, and (2) the experimentation metrics.

These aspects are related to the business long-term strategies as represented in Fig.7.1. This diagram represents how the two main aspects of experimentation used in this work relate to each other and to the business, and how they connect to the findings discussed in the previous section. Next, we detail these two aspects in two separate models, the experimentation activity model, and the metric model. These models were developed based on the collected empirical data and previous research.

7.5.1 The experimentation activity model

The experimentation activity model describes the different activities which comprise a single experiment iteration, from the experiment ideas, to the experiment analysis necessary for the running of trustworthy online experiments.

The arrows correspond to sequential connected activities. For example, when the experiment is launched pre-quality checks are run followed by ramp-up procedures before the experiment data is collected from a larger user base. Furthermore, our model divides the experimentation process into three sequential phases: the experiment development phase, the experiment execution phase, and the experiment analysis phase.

We illustrate the experimentation activity model in Fig. 7.2 and describe the three phases in greater detail.

7.5.1.1 Experiment development phase.

This phase refers to the specification and development activities of the experiment necessary to implement the variation change. This phase takes place before users are exposed to any variations. Finding A discusses how experiment hypotheses are generated and synthesized in the experimentation process. Experiment ideas are usually derived from four sources: (1) customer feedback (Finding A), (2) further iteration from previous experiments iterations [45], (3) need to understand and model the user behavior, and (4) less often through higher-level business goals (dashed line). The hypotheses are prioritized based on the experiment owners' prior analysis of how the hypotheses can impact the OEC. This analysis can be based on historical data from the feature, insights from other market segments (e.g. a feature that shows success in the US market might be prioritized for launch in another market or globally), or on experience gained from similar previous experiments or other products.

Following prioritization of the hypothesis a detailed hypothesis is elaborated. This includes specification of the experiment type (A/B, A/B/n, MVT etc.), how many variants are going to be present, cohorts or market segmentation, experiment duration and metrics to be used. Additionally, it covers the feature/change specification, including the area of functionality, actual and expected behavior, and implementation alternatives and considerations.

In addition, the detailed hypothesis specifies the target metrics that are expected to have impact and movements. The specification of the experiment metrics is closely related to the metrics used to prioritize the experiment itself. This includes lower-level signals that measure user-specific behaviors, guardrail metrics that indicate whether an experiment is within the allowed experiment conditions, and the leading metrics [72] that guide the experiment analysis. The metric selection is related to logging capabilities. The logging code represents the instrumentation of the system that interacts with the experimentation system. The logging code collects lower-level user behavior signals that can be used to compose complex metrics in the experimentation system. It is worth noting that the logging code should comply with the same standards (e.g. deployment, testing, code review, and integration pipeline) as any other product code.

Depending on the type of detailed specification of the hypothesis, the change in the system at the product level can happen in two ways, or a combination of the two. The first is through coding of the modification. This method is common when the experiment specification requires coding of a new feature or extensive modification of existing ones. The experiment set-up is a comparison of the current system with the change (treatment) and the system without

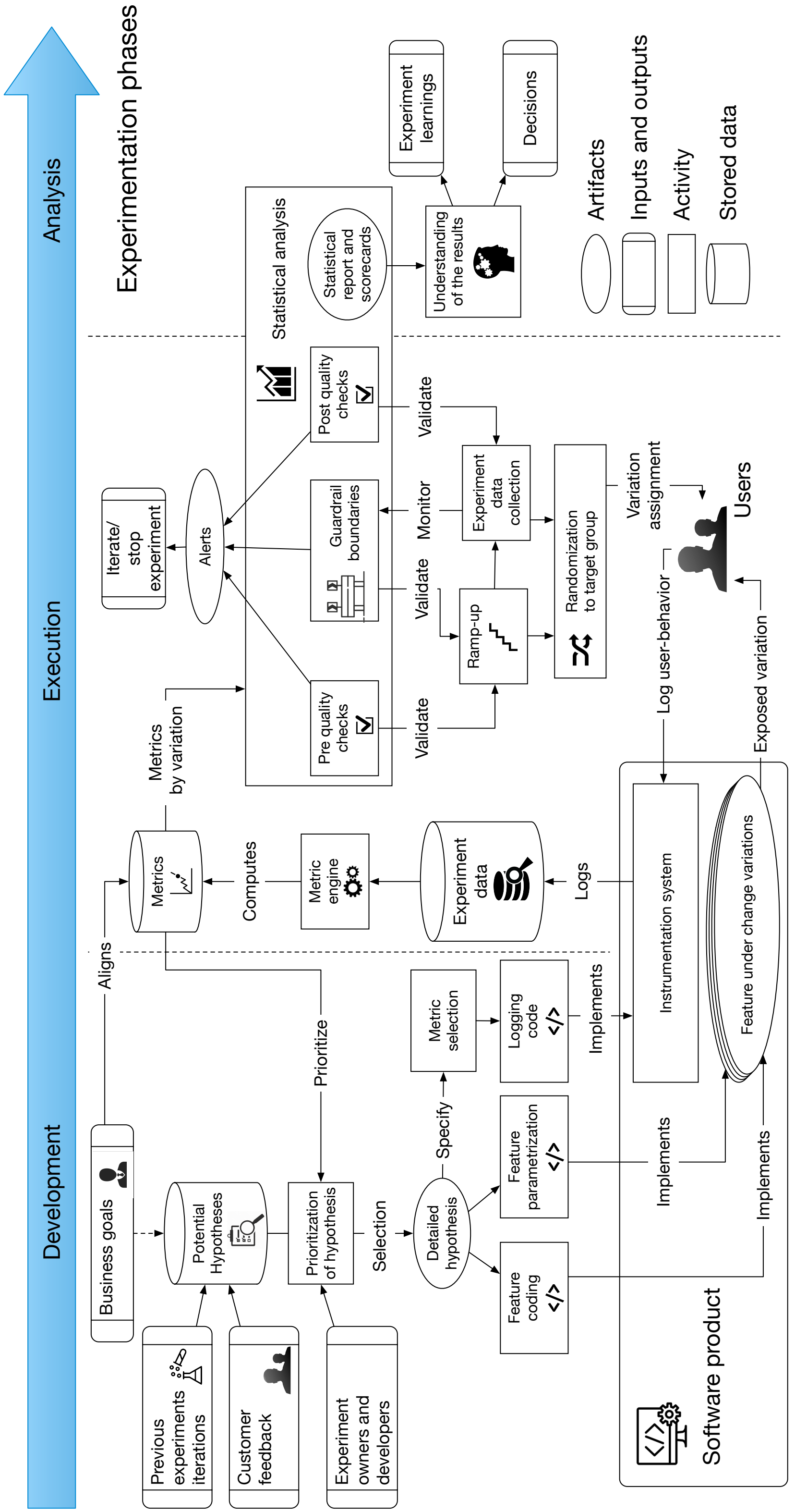

Figure 7.2: The experimentation activity model.

the change (control). The second way in which the change can be done is by parametrizing an existing functionality and running experiments to modify these parameters. In this case the functionality already exists but appears to have sub-optimal performance. The parameters of this functionality are configured during the experiment execution and are assigned to users. Although this might require a larger overhead in supporting and setting up a configuration manager, it reduces the effort and time spent on each experiment.

7.5.1.2 Experiment execution phase.

After the experiment is properly designed, the metrics selected, the change coded and instrumented, the experiment moves to the experiment execution phase. Here, users are randomized and are assigned to the experiment variations of the feature under change. The user behavior is logged, and the initial metrics are computed per variation using the metric engine and initial statistical analysis are computed using the statistical analysis tool. The metric engine is responsible for collecting and transforming raw data into experiment metrics. These metrics are consumed by the statistical analysis tool in order to run quality checks, check for guardrail-metrics and generate scorecards. The metrics, as discussed in the metric model, align the experiment goals with the business strategy and serve as input in the prioritization of experiment hypotheses.

The assignment of the users to the different variations can happened in two general ways (other methods used for websites are described in [2]). The first is to use a feature toggling system. In this case, the change in the system is parametrized using a variable, and the users are assigned to feature variations that the parameter activates (treatment group) or deactivates (control group). The second method is through traffic routing, where multiple instances of the system are run in parallel and the assignment system redirects the user to one of the multiple instances. The randomization refers to how users are assigned to a specific variation of the experiment during the execution phase. The randomization usually targets a specific group of users at the beginning and then generalizes to a larger audience. This target is specified in the detailed hypothesis. Although randomization might look intuitive, there are several techniques available to ensure that the randomization is not biased towards any variation [2, 164]. Then the instrumentation system captures the user behavior and logs the experiment data for use in statistical analysis.

The first step in the execution phase is to have confidence that the experiment will yield trustworthy results.

> *"A lot of effort goes into making sure the experiment passes the (pre-)quality checks. This is an essential step that gives us confidence in the experiment, so that we will not go to the next steps only to discover we did something wrong at the be-ginning"* — Principal Data Scientist

Before running experiments and exposing users to different variations, pre-quality tests are run to check for common pitfalls. Examples of pre-quality tests are: A/A or null tests, sample ratio mismatching, randomization checks, and offline testing. The A/A test assigns the users to the same variant A (the system without the change) with the aim of testing the experimentation system and assessing variability in the collected data [2]. The sample ratio mismatch (SRM) [43] is considered a critical diagnosis tool for online experiments. The

SRM checks the percentage allocation of the users. This allows the experiment owners to detect bias (that would invalidate the experiment results) towards any variation as well as check performance considerations. Randomization checks are tests which identify whether the randomization procedure is biased or has any patterns and checks the consistency of the randomization between sessions (to ensure that recurring users see the same variations). Offline testing uses historical data to assess the impact of the changes in the system and estimate confidence intervals [110].

After the pre-quality checks the activities that take place in the experiment execution phase are: the ramp-up, guardrail boundaries and experiment data collection. Ramp-up is a procedure where the treatment variations are initially launched to a small percentage of users. This is useful because critical errors can be detected early while exposing only a small number of users to the treatment variations. Large effect sizes in key metrics are mostly related to experiment errors [73], therefore fewer users' needs will be exposed to the change while identifying such errors. As the experiment runs without severe degradation, the percentage of users exposed to the treatment can be continuously increased until each variation has equal allocation, so that the experiment power is maximized [2]. A ramp-up procedure should be implemented together with an automated alerting capability with different significant levels and configurable actions. By checking guardrail metrics and experiment boundaries, such as key metrics that the experiment should not alter or deteriorate, the alerting system will alert experiment owners if something unusual is happening. In extreme cases, it will shut down an experiment with a significant negative impact if no action was taken. This allows organizations to invest in innovative and bold changes while reducing the risk of exposing users to bad ideas and errors [2].

After the main experiment execution, post-quality tests can be run to ensure that the experiment is valid and the data is reliable. Common post-quality checks are (1) checking for experiment invariant metrics, (2) learning effects, (3) A/A tests, (4) inter-action effects with other overlapping experiments, and (5) novelty effects [32]. Experiment invariant metrics are metrics that are not expected to change within the scope of the experiment. If there is a statistically significant movement in those metrics during the experiment execution, either the assumptions about the impact of the experiment are wrong or the implementation of the experiment is wrong. In both cases, it is worth exploring the reasons for these unexpected results. Other quality checks that are beyond the scope of this paper can be seen in [13,32,43]. The statistical analysis tool supports the whole experiment execution, computing guardrail tests and quality checks. Following the post-quality checks, the statistical analysis tool generates re-ports and scorecards for the key metrics of the experiment. Qualitative data collected from feedback boxes and other consumer feedback channels (if available) can be used together with the quantitative analysis to help explain the result.

7.5.1.3 Experiment analysis phase.

The analysis phase follows both the data collection and the conclusion of the experiment execution. The analysis phase consists of developing an understanding of the statistical output of the experimentation system in the context of

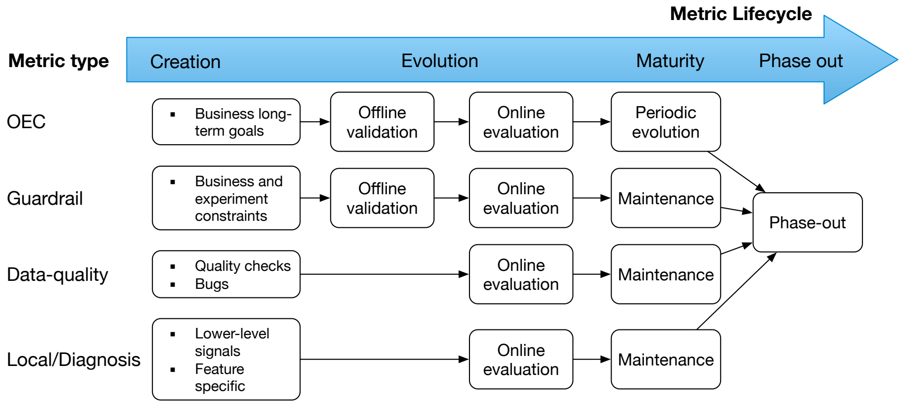

Figure 7.3: The metric model.

assumptions about user behavior.

Understanding of the results is an activity that analyzes the results from the statistical analysis in order to generate evidence about customer preferences and behavior and thus facilitate the decision-making process. It is important to reinforce that as the complexity of the experiment increases and there is not a standard OEC, key metrics can move in opposite directions, behave differently in different markets and in different user segments. In such scenarios, it is important to understand why different markets or user segments behaved differently. Not only does this generate meaningful knowledge which can be used to update assumptions about user behavior, but it also facilitates the process of decision-making and helps the evolution of the metrics and their alignment with the business strategy. Based on the results of this activity, the company can make decisions (such as ship or not ship the change) and update their assumptions and the metrics.

7.5.2 The experiment metric model.

A key component of the experimentation process described above is the metrics. Metrics guide hypothesis prioritization, the instrumentation required in the system, and the understanding of the results, and reveal whether the experiment results can be trusted. The experiment metric model that we discuss in this section characterizes two aspects of metrics: metric lifecycle and metric type. These two aspects are related to the findings B and C. These findings reinforce the central role that metrics play in the experiment design and execution. The metric lifecycle is divided into four main phases: creation, evolution, maturity and phase-out. The metric type is based on the four metric types identified in previous research [43]: OEC metrics, data-quality metrics, guardrail metrics and local feature and diagnostic metrics. The metric model is represented in Fig. 7.3.

The arrow in the metric model refers to the different stages in a metric lifecycle. In the creation phase, a first prototype of a metric is created. In this step, the metric consists of either aggregated lower-level signals (such as usage time, clicks, etc.), or modifications of existing metrics (such as linear combination or proportions)

The evolution phase consists of refining the metric and aligning it with the metric goal. During this process additional metrics can be combined with the original one to better capture more complex concepts. The refinement process can also impact the sensitivity of the metric. Offline validation and online evaluation are two techniques used to assess and support the evolution of a metric. The offline validation process analyses the metric directionality and sensitivity. Directionality refers to the direction of positive impact. The sensitivity of a metric refers to how well a metric is capable of moving due to the treatment. Techniques for offline validation of metrics can be found in [44, 162]. Online evaluation refers to analyzing the metric during an experimental run. This includes computing the metric with live users. The evaluation can be done through the comparison of the new metric with other existing metrics and through degradation experiments. Degradation experiments refer to the degrading of the user experience to find the directionality and sensitivity of metrics in the absence of an experiment corpus or analogous metrics for comparison. Strange movements of metrics during the experiment execution and quality tests can indicate instrumentation problems.

The maturity phase represents a period where the metric has been evaluated or validated and does not go through extensive modifications. For some metrics, the maturity phase represents a phase where they are updated in pre-established periods with learnings from multiple experiments or updated to accommodate changes in the business strategy of the company or the product, or only when issues arise in a maintenance process.

The last phase is the phase-out. In this phase, older metrics can be replaced with newer metrics, and metrics specific to an experiment or feature are deactivated after the experiment or feature lifecycle is over. It is worth noting that each metric might reach the different stages during different time frames. Metrics designed to be used only in one experiment can go through the creation to the phase-out process in only one experiment cycle. Metrics specific to features go through many experiment cycles, until the feature is only maintained or it is abandoned. OEC metrics that cross several features and even products can last many experimental cycles and years.

The second aspect of the metric model that we describe refers to the type of metric. Overall Evaluation Criteria (OEC) metrics guide the experiment outcomes and are a measure of the experiment's success. They represent and capture assumptions about business strategies and long-term company goals. OEC metrics are used across experiments and their evolution depends on the inputs from multiple experiments and the alignment with business goals. The evolution of OEC metrics affects multiple experiments and therefore is only updated periodically. The update of such metrics goes through offline validation and online evaluation.

Data-quality metrics are used in quality checks and inconsistency checks, such as implementation bugs, synchronization errors, and telemetry loss. Some of these metrics are feature specific, such as checking for data quality in experiments specific for a feature, while others are used in multiple experiments during pre-quality checks, such as the Sample Ratio Mismatch and checks for randomization imbalance during A/A tests. These metrics are evaluated online and their evolution and update occur when feature-specific modifications require updates or when issues arise. Guardrail metrics are metrics that are not used as an indicator of success but instead serve as boundaries for the experiment. Negative movements of guardrail metrics might be an indicator that experiment conditions were violated, generating alerts. These metrics, although they do not represent business directions as the OECs do, can represent business constraints on the OEC movement. These metrics evolve periodically in order to align with changing business restrictions. When updated, guard-rail metrics go through offline validation and online evaluation.

Local feature and diagnosis metrics are metrics used in individual functionalities of products. They do not impact other experiments and serve as diagnostic indicators used to understand the movement of OECs and guardrail metrics. Diagnosis metrics represent lower-level signals and serve as debug metrics for understanding unexpected movements of OEC and guardrail metrics. Local feature and diagnosis metrics are usually constrained to the experiment or feature lifecycle. Due to their short lifecycle these metrics are only evaluated online. To support the creation, evolution, maturity and phase-out phases of the different types of metrics, the experimentation team should support a metric management platform. This type of system prioritizes important metrics, constrains metrics to specific features and keeps track of inactive phased-out metrics for comparison and offline validation between older and newer experiments.

7.6 Conclusion

Online controlled experiments have become the standard practice for evaluating ideas and prioritizing features in most large web-facing software-intensive companies [18–20, 72]. Although companies can rely on models to start their experimentation organization and data-driven practices they might struggle to establish a trustworthy experimentation process as they scale their experimentation organization. Previous research provides models and processes for starting an experimentation organization based on higher-level descriptions of the experimentation process. However, these models do not capture all details and techniques that allow companies to scale and to ensure trustworthiness in the experimentation process [2, 21, 43, 157–160]. Based on a case study research with multiple product teams responsible for running online controlled experiments at Microsoft, we provide an experimentation framework composed of two detailed experimentation models focused on two main aspects; the experimentation activities and the experimentation metrics. This model discusses granular aspects of the experimentation process that can help companies and practitioners to scale their experimentation activities into a trustworthy experimentation process.

In future research, we plan to validate this experimentation process in other companies, to compare how the different activities map onto their experimentation process and analyze other aspects of the experimentation process, such as how the organization roles change during the evolution of the experiment.

Acknowledgments

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation. The authors would like to thank Microsoft's Analysis and Experimentation team for the opportunity to conduct this study with them.

Chapter 8

Paper D

Multi-armed bandits in the wild: Pitfalls and strategies in online experiments

Mattos, D. I., Bosch, J., Olsson, H. H.

Information and Software Technology, 2019, v.113, pp.68-81

Abstract

Context: Delivering faster value to customers with online experimentation is an emerging practice in industry. Multi-Armed Bandit (MAB) based experiments have the potential to deliver even faster results with better allocation of resources over traditional A/B experiments. However, the incorrect use of MAB-based experiments can lead to incorrect conclusions that can potentially hurt the company's business.
Objective: The objective of this study is to understand the pitfalls and restrictions of using MABs in online experiments, as well as the strategies that are used to overcome them.
Method: This research uses a multiple case study method with eleven experts across five software companies and simulations to triangulate the data of some of the identified limitations.
Results: This study analyzes some limitations faced by companies using MAB and discusses strategies used to overcome them. The results are summarized into a practitioners' guidelines with criteria to select an appropriated experimental design.
Conclusion: MAB algorithms have the potential to deliver even faster results with better allocation of resources over traditional A/B experiments. However, potential mistakes can occur and hinder the potential benefits of such approach. Together with the provided guidelines, we aim for this paper to be used as reference material for practitioners during the design of an online experiment.

8.1 Introduction

Delivering faster value to customers with online experimentation is an emerging practice in industry [2,9,11]. Web-facing software companies (such as Microsoft, Google, Netflix, Booking.com, Yelp, and Amazon, among others) often report on success cases and the competitive advantage of using post-deployment data together with online controlled experiments as an integral part of their development methodologies [2,18–20,23,32,115,165]. This competitive advantage leads companies to start experimenting with almost every change made in their systems, from developing new functionality to the fine tuning their systems. This intensive use is leading companies to deploy thousands of experiments every year [21,32,45]. A famous example of online experiment is the '50 shades of blue' experiment at Google. In this experiment, Google's engineers ran an experiment to determine the best shade of blue for a hyperlink in Google's search page. The best shade of blue resulted in an additional 200 million dollars in revenue [163,166].

To support the diversity and the scale of experiments, software companies and academic researchers are developing innovative solutions in automating experiments, scaling the experimentation infrastructure, and in developing new algorithms and experimental designs [18,23,45,110,143,162]. One emerging class of algorithms, known as Multi-Armed Bandit (MAB) [113,167], is being widely explored in the context of online experiments and have the potential to deliver faster results with better allocation of resources [110] compared to traditional experiments, such as A/B testing. However, the incorrect use of MAB-based experiments can lead to misinterpretations and wrong conclusions that can potentially hurt the company's business.

To the best of the authors' knowledge, there is no work that discusses the limitations of MAB-based experiments. This work attempts to addresses this gap from the industry perspective using a combination of a multiple case study with simulations. This study provides analyzes some limitations faced by companies using MAB and discusses strategies used to overcome them. The results are summarized into a practitioners' guidelines with criteria to select an appropriated experimental design.

The remainder of the paper is organized as follows. Section 8.2 provides a background review of the MAB problem and algorithms, controlled experiments and A/B testing and the experimentation processes. Section 8.3 discusses the research method and threats to validity. Section 8.4 presents and discusses the restrictions and common associated with MAB implementations for online experiments. Section 8.5 presents a discussion of the results, use cases where MAB algorithms are desired and a guideline process to select between traditional experimentation techniques such as A/B experiments and MABs. Section 8.6 concludes and discusses related research challenges.

8.2 Background

In this section, we consider the different aspects of running online experiments. We describe a traditional online experiment in the form of an A/B test and discuss some of the limitations of this method. Next, we present the MAB class of problems and discuss some of the advantages of MAB. In the appendix we present the MAB algorithms used in the simulations.

8.2.1 A/B experiments

A/B experiments, also known as A/B test, are a group of techniques based on the design of experiments [16] and hypothesis testing, where users are randomly assigned to different variants of the product. The control variant is the current system without any modifications and the treatment variants are the current system with a modification X_n. The modification X_n can be modifications in existing functionalities of the system (e.g. a different set of parameters for each variation), or the system with a new feature or functionality (e.g. different implementations of new features). All variations are properly instrumented and send data to the research and development organization. Metrics to evaluate the system are grouped and computed based on the variation of the system. The different variations are randomly assigned to different users, and, after a predetermined period of time, the metrics for each variation are statistically compared. If the only consistent difference between the experiments' variants is the modification X_n, and the randomization assumptions hold true, the research and development organization can establish a causal relationship between the modification in the system and the change in the metrics between the different variations. Kohavi et al. [2] provide an in-depth discussion of different experimental design techniques used in software online experiments. Additionally, to support A/B testing in software systems, different experimentation models and frameworks were developed, such as the HYPEX [5], RIGHT [7, 8] and Framework for Online Controlled Experiments [27]. These models focus on the software and organizational process to establish an experiment instead of focusing on the experimental design and statistical aspects.

Two types of decision errors are discussed in traditional hypothesis testing and A/B testing. Type I error occurs when we reject the null hypothesis when it should be accepted (a false positive). Type II error occurs when we accept the null hypothesis when it should be rejected (a false negative) [168]. Online experiments control type I error by utilizing strict level of significance in their tests while control type II error by increasing the sample size [2].

One of the challenges faced by many small- to medium-sized companies is the high number of users required to run a high confidence levels and high-power A/B experiments [41]. Collecting such an amount of data can take between weeks and months [23, 45], even in large companies. The time to run even a single experiment can negatively impact the decision-making process and time to deploy a new feature in most software organizations. Another key challenge is the choice of metrics [43]. An experiment can have multiple metrics that might move in opposite directions. The analysis of these experiments is a challenging process, as they can involve multiple stakeholders and they require a clear view on how these metrics connect to the business goals of the company [27, 43, 44].

8.2.2 Multi-Armed Bandit

The MAB problem (also known as K-armed bandit) is a problem that aims to maximize the expected gain when we have a limited set of resources that should be allocated in unknown, competing variants [113, 167]. Its name comes from the analogy of a gambler facing a row of slot machines (one-armed bandits) trying to decide which machine to play and in which order, trying to maximize the received reward [113]. MAB algorithms are classified under the umbrella of reinforcement learning algorithms examining the exploration-exploitation trade-off.

The MAB problem can be formalized as [110]:

$$a = \pi(\delta), \text{ Arm } a \in a_1 ... a_K \tag{8.1}$$

$$y = r(a, \delta'), \text{ Reward } y \in \mathbb{R} \tag{8.2}$$

where a is the arm selected, K is the number of arms, π is the user-defined policy function to balance the exploration of arms and the exploitation of the best arm so far, δ is and δ' are noise variables, y is the measured reward and r is the unknown reward function for the selected arm. A MAB algorithm is the user-defined policy π to select the arm.

In online experiments, each arm corresponds to variant of the system. This can be different implementations of a feature, different parameters or different changes. The reward y is the (user-)metric, such as clicks or conversions. The reward metric should have a positive direction (the higher value of y is considered better than a lower value). Usually, MAB algorithms restrict the reward metric to the interval $y \in [0, 1]$. Although the concept of reward is intuitive and have a direct relation to the online experiments counter-part, several MAB algorithms are formulated and discussed using the concept of regret and cumulative regret [143]. Regret is the difference of the drawn arm and the optimal arm while the cumulative regret is the difference of the cumulative optimal reward, i.e. the total reward got if the optimal arm was chosen every time, and the actual total reward received. Formally it is defined as:

$$\text{CumulativeRegret}(t) = r(a^*) \cdot t - \sum_{s=1}^{t} \mu(a_s) \tag{8.3}$$

where $\mu(a)$ is the mean reward of the arm a over the time and a^* is the optimal arm:

$$a^* = \max_{a \in a_1 ... a_K} \mu(a) \tag{8.4}$$

One of the main advantages of MAB algorithms compared to A/B experiments is the minimization of the cumulative regret. This allows the algorithms to dynamically change the user allocation to the best performing variants. Another advantage occurs in situations with clearly underperforming variants. In such cases, the algorithm changes the allocation of users from the bad variants to the better ones, helping to differentiate good arms from the best ones quicker [169]. However, MAB algorithms have several assumptions that must be met for a correct inference, such as independence of the arms, reward distribution, instant reward, reward time invariance, single metric constrain, among others. As with A/B experiments, the choice of the reward greatly impacts the result and conclusion of the experiment.

The Appendix describe four common MAB algorithms: the epsilon-first, epsilon-greedy, UCB1 and the Softmax.

8.3 Research method

In earlier discussion with practitioners, we identified that, although academic research suggests that MAB algorithms provided several benefits, companies were not using MAB extensively in practice. Some of these companies suggested that these algorithms did not provide the expected benefits and that they even showed several limitations. Based on these observations, we designed this study to identify what are the restrictions and pitfalls of MAB-based experiments from the point of view of software companies, and how these limitations can be addressed when designing an experiment. This is captured by the following research questions:

RQ1: What are the restrictions and pitfalls associated with MAB algorithms applied to software online experiments?

RQ2: What are the decisions involved in the design of MAB-based online experiment?

To answer the two research questions, we conducted a multiple case study and triangulated the data with simulations. The multiple case study allowed us to explore the rationale behind the experimental design decision process, such as the decision of choosing a traditional A/B testing over a MAB-based experiment. The simulations allowed us to verify situations mentioned by the interviewees in the case study and, when possible, triangulate the data.

8.3.1 Multiple case study

The case study process was conducted following the guidelines presented by Runeson and Höst [77]. This multiple case study can be divided in three stages: (1) Definition and planning, (2) Data selection and collection and (3) Data analysis. Next, we provide an overview of each of the three stages.

8.3.1.1 Definition and planning

Based on the observations from earlier research that despite the benefits suggested by research, online companies were not using MAB algorithms, we designed a multiple case study to understand: (1) the restrictions and pitfalls that companies running online experiments encounter when using MAB algorithms (RQ1), and (2) how these companies overcome these restrictions and pitfalls (RQ2).

8.3.1.2 Data selection and collection

The main data-gathering instrument used in this multiple case study was thematic semi-structured interviews based on an interview guide. Our choice for thematic semi-structured interviews was based on the flexibility of asking open questions to discuss a broad range of problems inside the same theme [77]. The case study companies and the subjects were selected based on the criteria that the companies run online experiments on a daily basis or develop solutions for online experiments, including A/B testing, factorial experiments, and/or MAB algorithms. All the subjects selected for the case study had experience of online experiments and knowledge of either developing or running MAB algorithms in their systems. This selection criteria limits the interviewees to a restrictive pool of experts available outside academia. This is reflected in the position occupied by the interviewees, where almost all are senior or principal data scientists and developers.

This multiple case study was conducted together with five companies between the period of August 2017 and January 2018. The primary data sources of this study are transcripts of audio recordings, interview and meeting notes, emails, and other information shared by the interviewees, such as slides and technical reports. A total of 11 interviews were conducted. with minimum duration of 33 minutes, maximum duration of 70 minutes and a mean value of 40 minutes and a median of 38 minutes. The number of interviews was decided based on the saturation criterion, where no new information or points of view were gained from new subjects [77].

The interviews consisted of a questionnaire containing four general open-ended questions aimed at identifying problems using MAB algorithms seen in practice, identifying the restrictions and pitfalls of these algorithms that prevented their utilization in practice, identifying good use cases where these algorithms can be used, and identifying the strategies used by the companies to overcome these limitations. The interview started with a brief introduction and description of the goals of this research. Next, we asked the participants about their familiarity with MABs and A/B testing, and their industrial experience with them. Next, we asked if they had experienced any limitations or problems with MAB-based experiments compared to A/B testing. Next, we asked about strategies used to address these problems and limitation, and what would be good cases for MAB-based. During the interview we reported limitations faced by other companies, anonymously, with the goal to verify if similar limitations were also present. During the data analysis, any information that was not clear or that needed further explanation was discussed again with the interviewees.

The companies Due to reasons of confidentiality, we have provided only a short description of the companies and their domain. The interviews were conducted either in person on the company's premises, or online via video conference. Table I provide a list of the interviewers' position in each company.

Company A is a multinational conglomerate company that manufactures consumer electronics and provides software solutions for consumers, professionals, and business-to-business solutions. Different teams inside this company are running A/B/n experiments and have conducted the preliminary research and implementation of MAB algorithms for experimentation in their software products. In company A, we interviewed four practitioners working with two different products.

Company B is a multinational technology company that develops, manufactures, and sells software solutions and services ranging from operating systems to web solutions. Several of the company's products run A/B/n experiments on a regular basis or at scale, and they have also successfully implemented and run MAB in some of their products and have evaluated MAB algorithms for their online experimentation platform. In company B, we interviewed a total of four practitioners working in the same experimentation team but with different products.

Company C is a company that develops experimentation solutions for its customers. The company offers A/B/n, MVT and other experimentation tools for websites along with frameworks for experimentation in mobile platforms. The company developed their own statistics engine and offers solutions using MAB algorithms to customers. The company's customers include several multinational companies from different domains, from software companies, to entertainment and large news agencies. We interviewed two practitioners involved with the MAB solutions.

Company D is a software company focused on website optimization and offering experimentation tools and solutions for A/B testing and MABs. One practitioner was interviewed, the director of data science in the company. However, since the interview this interviewee has left the company and provides consultancy in the area of experimentation and MAB algorithms, with experience in developing A/B experimentation platforms and MABs for several multinational companies, from software companies to large news agencies.

Company E is a travel fare aggregator and travel engine provider. It develops booking and travel solutions used by both individuals and the travel industry. A/B testing methodologies are an integral part of the development process of the company. The company is introducing and evaluating MAB algorithms in their online experimentation process. One practitioner was interviewed. The interviewee is a senior data scientist and is responsible for the development and evaluation of a MAB tool to be used in internal development.

8.3.1.3 Data analysis

First, the empirical data for each interview was transcribed, where applicable, and thematic coding was applied [170]. From the coding, we identified the limitations as well as the potential solutions. Although some of the reported data was specific to the context of their work, some of the codes could be grouped into a more general limitation or solution. This grouping procedure was analyzed and interpreted by all authors.

8.3.2 Simulations

During the data collection of the multiple case study, some of the interviewees also shared simulation designs and ideas to illustrate some of the concepts. These simulations were reproduced in this paper with the aim to triangulate the data shared in the interviews with hypothetical scenarios that could happen in MAB-based experiments. Each simulation is presented as a hypothetical scenario and it is presented and discussed in detail in the respective section.

8.3.3 Threats to validity

8.3.3.1 Construct validity.

This study was designed with semi-structured interviews, where the open questions allowed the interviewees to express their opinions and elaborate more in the problems, they faced with MABs. In all cases, we ensured that we were using a consistent vocabulary with the technical terms with which the participants were familiar. In cases where there was a difference in vocabulary terms, we had other technical synonyms and examples from the available literature and textbooks in both online controlled experiments and in MAB areas.

8.3.3.2 External validity.

We identified two issues that can limit the generalization of the identified results for other companies: (1) the representativeness of the studied case companies, and (2) the number of practitioners who participated in the interviews. We tried to minimize the first issue by selecting companies that both developed experimentation systems for their own products and companies that commercialize and help other companies to run experiments. We believe that this provides a broad view of the different limitations faced by medium- to large-sized companies in the web domain. However, the results discussed in this study are general and could be of potential benefit for small-sized companies and companies that are not in the web domain. Regarding the second issue, we conducted this multiple case study with highly qualified people in both fields until we reached saturation [77]. We acknowledge that some other limitations might not have been identified due to the high experience of the interviewees or because of their problem domain. To address this, we conducted interviews with practitioners working on different products and in different companies.

8.3.3.3 Internal validity.

To minimize internal validity threats [77], we took the following precautions: (1) during interviews we discussed potential limitations identified by other companies or by traditional online experimentation methods such as A/B testing. This procedure was applied after the interviewees had the opportunity to comment on their own experience, without the bias of different companies or groups. (2) the interviewees were selected based on their expertise and experience in the area, therefore we excluded inexperienced practitioners with both techniques (A/B experiments and MABs). The goal was to reduce limitations that were confounded with the lack of experience. However, we recognize that there might other limitation associated with first time introduction of MAB-based experiments in a team or company. (3) at the end of all interviews we summarized the discussed limitations and strategies, giving the opportunity for the participants to comment on our interpretation. This summary was important to clarify all the discussed aspects and remove interpretation ambiguities. (4) Some of the companies did not permit us to record of the interviews so the researchers had to rely on their interview notes. Immediately after each interview, the researchers discussed and complemented their interview notes with a self-contained summary of the discussion of the results. In the case of inconsistency, the researchers contacted again the interviewees for clarification.

8.4 Results

This section discusses the results obtained from the collected empirical data from the interviews and from the simulations.

8.4.1 Decision errors in naïve MAB implementations

8.4.1.1 Pitfall

The first identified pitfall is the naïve implementation of MAB algorithms in experiments that require the control of type I errors (false positives). We refer to naïve implementation as those that select the best ranked arm as the desired decision without considering confidence intervals or controlling for type I errors. This is often the output of MAB algorithms in research [143]. As identified by the interviews, this pitfall is usually observed in the first few MAB-based experiments. A factor that favors the occurrence of this pitfall is the introduction of MAB algorithms without in-depth knowledge of the algorithms, without explicit goals for the outputs, or when using third-party experimentation tools that have naïve implementations.

This was experience by the developers of company A, when using third-party systems and by the senior data scientist in E in first-time implementations of MAB algorithms. Company E also identify this pitfall when adapting and using MAB-based recommendation/content-serving systems in online experiments applications. According to the senior data scientist in company E, one of the reasons this pitfall persists is because it is common that academic MAB research is concerned with other aspects do not mention or implement statistical analysis on top of MAB algorithms.

> *"We already made a few decisions based on the result of this third-party tool when we decided to compare it with another A/B testing tool. The results were so different that we decided to investigate the reason"* — Senior Developer Company A

8.4.1.2 Simulation

To illustrate this scenario, we simulated a simple decision-making process based on a traditional A/B experiment and three naïve implementations MAB algorithms (ϵ-greedy 10%, Softmax 10% and UCB1). Suppose an online company decides to use a third-party experimentation service that provides both A/B and MAB algorithms. The feature that is going to be experimented with does not influence the metric. However, adopting this feature increases the time spent by a team maintaining and developing the feature. The service provides statistical analysis and confidence intervals for the A/B experiments (based on a χ^2 test using a significance level of 95% [2]), as several competitors also offer. The experimentation service advertises MAB as the next level for online experiments and considers it a big differentiator as most competitors do not have it. However, for MAB only a ranking of the best arms is provided (the naïve approach often available in research and commercial systems). This

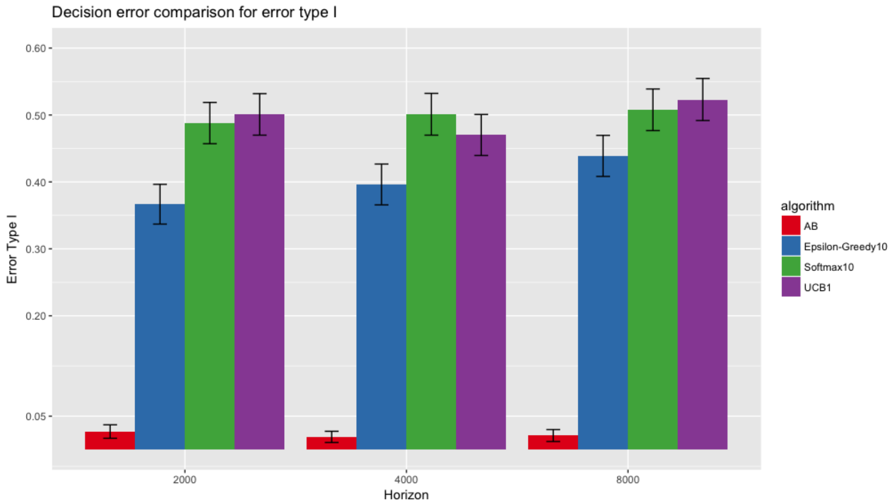

Figure 8.1: Comparison of the decision error for type I error in A/B experiments (red) and naïve implementations of MABs. This comparison suggests that in the absence of a rigorous statistical framework, MABs algorithms are prone to decision errors when applied in contexts where minimization of error type I is required.

simulation investigates type I decision errors in three different sample sizes in a similar situation, where there is no difference between the variants.

The conditions of the simulation are: each algorithm was simulated with a Monte Carlo (simulated 1000 times) procedure for three different sample sizes. The first horizon has 2000 unique samples representing a power of less than 60%. The second horizon has 4000 unique samples, representing a power of 80%. The third horizon has 8000 unique samples, representing a power of over 95%. All the sample size calculations are calculated based on the necessary power to conduct an A/B experiment with a significance level of 5%, initial conversion baseline of 10%, and absolute effect size of 2% [2]. The confidence intervals were obtained by utilizing a Bootstrap resampling (with n=1000) procedure in the simulation results.

$$A, B \sim \text{Bernoulli}(0.1) \tag{8.5}$$

Figure 1 shows the percentage of error type I in the Monte-Carlo simulation. It can be seen in this picture that the A/B experiment process still makes error type I, but this error is constrained to the pre-defined significance level of 5%. However, the MAB algorithms often make error type I. If MAB algorithms are used in such situations, the experiment has a negative impact in the company, as many features that were incorrectly validated now are maintained and developed by the company, but without any real impact in the metrics.

8.4.1.3 Strategies

Experiences from companies A, B and E suggest that the experimentation organization and the experiment owner should first estimate the consequences of committing each type of error. Next, identify how the outcome of the experiment will be used in the future development and evolution of the system.

If the estimated cost of a type I error is high, a good strategy is falling back to the traditional A/B experiment scenario, which already provides consistent processes and tools. If the estimated cost of a type I error is low and the aim of the experiment is to minimize regret (opportunity cost of missing the best solution) or type II errors (false negatives), MAB-based experiments can aggregate value to the experiment. Type I errors can lead companies to invest development effort and maintenance in features that do not deliver value to the system. If the goal of the experiment is to both minimize regret and control for type I errors, it is necessary to implement a more rigorous statistical analysis on top of the MAB algorithm. In this scenario, the algorithm is still minimizing regret during its execution, but the decision-making process also follows the statistical analyses. The disadvantage of this method is the lack of balance between the variants reducing the statistical power of the experiment, this implies in a longer experimentation time.

Some experimentation tools utilize a MAB algorithm as the default (e.g. in Google Analytics [169]). For those systems, it is necessary to conduct a careful analysis of the algorithm implementation (if available), the experiment assumptions and the expected outcome. If possible, the results from MAB algorithms should be analyzed with their respective confidence intervals as this can provide a more in-depth comparison between the different arms and generate additional insights.

8.4.2 Bad variation lockdown

8.4.2.1 Restriction

User experience consistency is one important requirement for some online experiments [2, 19]. If the user is exposed to a variant, the user should have continuous access to the same variant during the experiment. Several methods can be used to keep the experience consistent, such as: (1) using deterministic randomization reassignment [2, 19]; (2) requesting a variation only once and caching it in the application (such as in cookies); (3) cross-checking and comparing the variant during every assignment. Deterministic randomization reassignment is traditionally computed through pseudo random number generation and hashing functions (such as SHA1) on the unit of randomization (e.g. cookies, user ID, instance number). This procedure ensures that every time the system requests a variant, the assignment system always replies with the same variant. After the experiment is completed the users can be assigned to a fixed variant (one of the treatments or the control variant). The caching and the cross-checking methods have drawbacks compared to the deterministic randomization reassignment, such as being difficult to scale in both the number of simultaneous experiments and number of users, lack of support for ramp-up and automated shut-down [2].

In A/B experiments, the user's experience consistency depends on the experiment definition (e.g. choice of the randomization unit) and how the experiment affects the user experience. Implementations of MAB should also consider the user experience consistency. However, several MAB algorithms do not support deterministic randomization reassignment in their current formulation and implementation, as they do not use randomization units in their assignment process. Therefore, MAB algorithms rely on cross-checking and caching methods, which introduces complexity, limits the scaling, and restricts the use of ramp-up and automated shut-down. An additional consideration should also be analyzed: if a user is assigned to a clearly bad variant (or arm), will this user be locked to this variant until this experiment is shut down in a new deployment? This could potentially hurt a small percentage of the users for a long period, if workarounds in the caching and cross-checking mechanisms are not implemented. This was identified by the architect in company A and two principal data scientists in company B.

Company C reports a challenging in keeping user consistency in MAB algorithms in long-term experiments (such as continuous optimization experiments) with limited and recurring users. Long-term experiments can run for between months and years and are often associated with content serving experiments. In this scenario, the experiment has a limited number of users that are exposed to the same variant multiple times. The users are asymmetrically assigned towards one variation, which might not be the optimal one, as the optimal variation is under-sampled. This situation prevents both the minimization of the regret and limits the statistical power in future statistical analysis, threatening the validity of the experiment and the confidence in the experiment results. If implementing an extra reassignment procedure, how often and in what ways are users reassigned? If the reassignment occurs seldom, the validity of the experiment and the regret minimization are at stake. If the users are reassigned very frequently, this could create a user experience inconsistency. Other alternatives such as only reassigning the users of a lower variant could influence the exploration rate and the regret minimization, if not carefully analyzed.

> *"If user consistency is necessary, we strongly encourage our customers to go with A/B testing route instead of bandits"* — Senior Statistician Company C

8.4.2.2 Strategies

If user consistency is mandatory, a reasonable alternative is to use traditional A/B experiments, where tools and techniques for keeping this consistency while maintaining ramp-up and automated shut-down are available with deterministic randomization reassignment. If using caching or cross-checking methods, it is necessary to implement an extra layer to the application code in order to handle ramp-up and automated shut-down cases, increasing the complexity of the experiment. However, if such a layer is already present, user consistency can be implemented when using MAB algorithms.

Long-term MABs should not be used for exploration purposes or for understanding of the system and the user behavior. In such cases, having control of type I errors and predetermined power are of greater importance than regret minimization. If the goal of the experimentation procedure is to understand both the system and the users, the A/B experimentation procedure can provide better insights without the reassignment and variant lock restriction. If the aim of the experiment is to conduct a long-term optimization procedure, a short-term A/B experiment can be executed first. This experiment aims to understand the functionality/feature under experiment and how the unit of diversion performs. What is the randomization unit? How often do recurring randomization units request a new variant? To what extent should user experience be consistent for this experiment? Do users perceive a significant experience change between variants? Those questions can be answered with an A/B experiment prior to the long-term MAB optimization. Content-serving MAB applications (such as serving ads and news) often do not impact the user experience negatively but optimizing layouts and other user interface elements might pose problems that need to be carefully analyzed.

8.4.3 Decision errors due to violations of assumptions

8.4.3.1 Pitfalls

A common pitfall is to assume that the MAB-based experiments have the same assumptions as A/B experiments. For example, some MAB algorithms assume that the reward observations are identically and independently distributed. If these assumptions do not hold in a MAB-based experiment the experiment conclusion might be biased and the regret is not minimized as presented in theory. A common violation is when there are changes in the distribution over time, as discussed by company D. For example, the distribution of clicks in a website could change if it is weekdays or weekends. In weekdays, the system has more business activity, while in weekends it has more home activity.:

$$X(t) \sim \begin{cases} \text{Bernoulli}(p_1), & \text{if } t = \text{ weekdays} \\ \text{Bernoulli}(p_2), & \text{if } t = \text{ weekends} \end{cases} \tag{8.6}$$

Similarly, in ecommerce, the click distribution changes depending on the day of the month and period of the year. With those temporal changes, the MAB algorithm can learn and allocate exploitation to a suboptimal arm. Therefore, questions such as "should I launch my experiment in a Tuesday or on a Saturday" [171] and seasonality effects [157] arise. In this case, a MAB-based experiment will take a longer time compared to A/B experiments to compensate for these effects.

Another violation case occurs with novelty effects, as they also change the distribution in time [13,19]. The novelty effect of a new feature might increase a metric (such as usage) in the beginning and then have a drop of usage over time. After a long period, the metric of the treatments might have a statistically significant lower metric compared to the control variant.

Other violation assumptions, pointed out by the empirical data are:

[a] the correlation between different arms. Algorithms such as UCB1 are not appropriated to understand and exploit correlations in the arms. Using algorithms that make independence assumptions with correlated arms might lead to an overestimation of the best arms as well as a weak exploitation of the solution space [172].

[b] the presence of lagging metrics or delayed reward feedback (compared to the experiment horizon) [173]. Some commonly tracked metrics have dependency on time, e.g. 3 days' or 15+ days' user retention. Using MAB algorithms that assume instant feedback with non-negligible delay metrics (compared to the experiment horizon), can lead to incorrect conclusions.

According to a principal data scientist in company B, as metrics align more to the business and represent more abstract concepts they start to lag, increasing the delay in the reward. With these metrics, MAB-based experiments are less attractive.

[c] the re-scaling and normalization of metrics. Some MAB algorithms, e.g. UCB1, require the reward to be constrained between 0 and 1. Existing metrics need to be re-scaled and normalized to fit into this constraint. However, the re-scaling and normalization transformations need to be carefully analyzed, implemented, and validated. Common pitfalls in this process that can change are: 1) Transformation on the fly, all new data or group of data points change the transformation rule, e.g. subtracting current expected value or dividing the result by the current standard deviation. As the experiment progresses, the metric changes and cannot be compared. 2) Utilizing historical data that are no longer valid, e.g. a transformation procedure that relies on data that are not representative anymore. 3) Carrying transformations from previous experiments and metrics to new ones, e.g. using 3 days' user retention transformation for a 15+ days' user retention metric might result in an incorrect measurement. The rescaling was discussed by one the developers in company A.

8.4.3.2 Simulation

The following simulation explores an experiment with the novelty effect is explored in the following simulation. The goal of this simulation is to show that even with traditional statistical analysis on top of the algorithms, MAB can still inflate error type I. Suppose a company is launching a new voice in the voice assistant tool for their mobile application. Prior to launch the company post videos of the feature and make advertisements creating some user expectation. Consider that the variant A is the current voice and B is the new voice. The company is measuring whether the user uses the feature during the day.

For this simulation, we consider the following distribution for the usage of the voice assistant tool. The reward 1 represents that the user has used the tool and 0 represents that the user has not used the voice assistant tool during the day. t is a discrete positive variable that represents the day after launch (t=0 means the launching day, t=10 represents the feature usage after 10 days). For the variant B, we consider that it starts with a higher usage due to the novelty effect, but it slowly declines until it reaches a constant usage level of 3% lower than the baseline variant A. The distribution of the response for each variation can be represented as:

$$A \sim \text{Bernoulli}(0.1) \tag{8.7}$$

$$B(t) \sim \begin{cases} \text{Bernoulli}(0.12 - 0.005t), & \text{for } t < 10 \\ \text{Bernoulli}(0.07), & \text{for } t \geq 10 \end{cases} \tag{8.8}$$

The simulation runs for a total of 20 days. Each day has the same number of users. Therefore, the variant B performs poorly more than half the experiment time. This simulation concerns with the number of times we make an incorrect

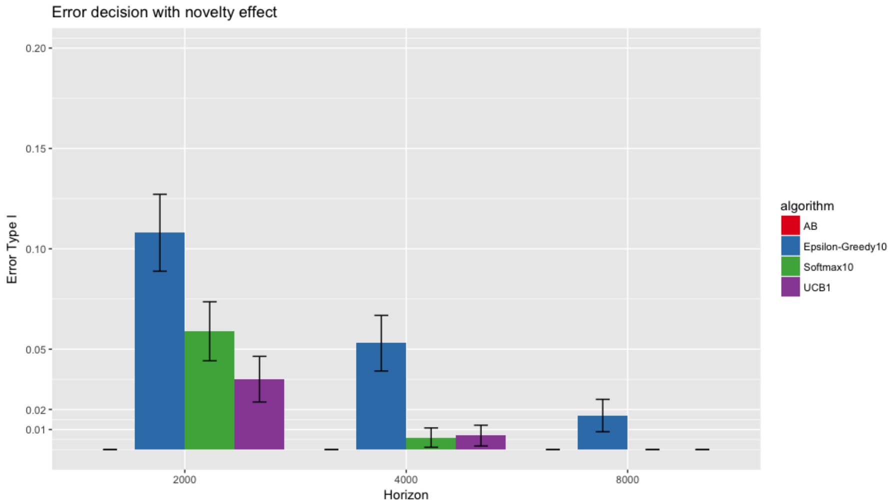

Figure 8.2: Comparison of the decision error for type I error for A/B experiments and MAB algorithms in the presence of a a simulated novelty effect. In the picture above, there were no A/B experiments' decision errors. It is possible to see that these algorithms can still lead to decision errors, in particular in underpowered experiments. Similarly, if the variants start performing differently overtime, MAB algorithms can bias towards earlier variations.

decision in selecting alternative B (making a type I error). To compensate for the decision errors' effects of naïve MAB decision-making (as discussed in section A), this simulation implements the same statistical analysis and decision criteria for all algorithms (A/B and the MAB algorithms). The decision is based on a final collected data and the process detailed in [2] using a significance level of 95% (or $\alpha = 5\%$).

The conditions of the simulation are: each algorithm was simulated with a Monte Carlo (simulated a 1000 times) procedure with for three different sample sizes. The first horizon has 2000 unique samples, representing a power of less than 60%. The second horizon has 4000 unique samples, representing a power of 80%. The third horizon has 8000 unique samples, representing a power of over 95%. All the sample size calculations are calculated based on the necessary power to conduct an A/B experiment with significance level of 5%, initial conversion baseline of 10% and absolute effect size of 2% [2]. The confidence intervals were obtained by utilizing a Bootstrap resampling (with n=1000) procedure in the simulation results.

Figure 8.2 shows that shows that in the presence of non-identically and independently distributed users and arms MAB algorithms can inflate error type I ($\alpha = 5\%$), and this is proportional to the power of the experiment. In underpowered experiments, MAB algorithms with the same decision-making process of A/B experiments are not able to control for type I errors and have a significantly higher incidence of incorrect decisions due to biasing earlier variations.

8.4.3.3 Strategies

Different strategies can be used to overcome violation assumptions. As highlighted by one of the interviewees, they require careful analysis and modeling of the problem.

> *"You need to be careful when deciding to go for bandits, if you know that some of these assumptions were violated you need to do a proper statistical modelling of the problem and validating it before deploying your algorithm"* — Director of Data Science, Company D

The first strategy is to choose a re-weighting scheme for the MAB. Most algorithms use a constant average scheme that places equal importance to all samples. Different schemes, such as moving averages or exponential averages can be used to shift the reward importance with time [172].

A second strategy is to use contextual bandit algorithms. These algorithms consider domain-specific knowledge to provide and personalize the exploration space. However, these strategies add complexity to the algorithm in both implementation and the number of hyper-parameters to tune. Moreover, these extensions require specific domain knowledge of the system and the user behavior with time. Such knowledge might not be available at the time of the experiment or might require a pre-A/B experiment to test some of the assumptions.

> *"Research commonly takes the bandit reward and context for granted, but the process of selecting a good context and reward is iterative and slow. In practice, it is very difficult to compare two bandits and how they perform against each other if you are using different contexts"* — Senior data scientist, Company E

Different solutions for the MAB with delayed reward problem. However, this is still a computationally difficult problem [163]. One solution is to run long-term MAB algorithms in order to minimize the effects of the delays. However, this requires modification in the metrics and new contextual information. Traditional A/B experimentation provides a more robust framework to evaluate experiments when there are uncertainties in the assumptions.

8.4.4 Lack of Sample Ratio Mismatch quality check in MAB algorithms

8.4.4.1 Restriction

Sample Ratio Mismatch (SRM) is considered to be one of the critical diagnosis tests for A/B experiments [163]. This test allows checking if the percentage allocation of users for each variant is within an expected confidence interval and validate the randomization system. Suppose that a company is running an A/B experiment where 50% of the users are allocated to each variation. Variation A is assigned to 99,000 users and variation B is assigned to 101,000. Comparing this user distribution with the designed proportions of 50% indicates that the current user distributions is very unlikely to be in the 50% designed proportions ($p < 0.01$ with a χ^2 test). Early detection of SRM helps to prevent randomization bias towards any variation and allows the experimentation organization to check performance and instrumentation issues. Sample Ratio

Mismatch is commonly conducted in the pre-experiment A/A test phase [2] and during the experiment execution as a guardrail metric [43].

It is not possible to run SRM checks in MAB-based experiments, as the regret minimization depends on the asymmetric sampling, favoring one of the variations. Therefore, it is hard to identify randomization and instrumentation bias in the system during the experiment execution. The lack of data quality checks can hinder problems in the experimentation solution and lead to a lack of trust in the data. As highlighted by one of the interviewees:

> *"We don't deny the advantages of MABs for recommendation systems, but if we can't run quality checks, we don't trust our data and then we don't have a business case for using it"* — Principal Data Scientist, Company B

Without proper validation, MAB-based experiments can be minimizing the regret towards a biased variation. This restriction was mainly identified with Company B. However, companies A, C and E also confirm the difficulty in implementing MAB quality checks and validating and guaranteeing that the MAB implementation is not being biased due to external factors such as the instrumentation and randomization system.

8.4.4.2 Strategy

A strategy employed by company A is to evaluate the MAB experimentation system (instrumentation, metrics, randomization techniques, etc.) using traditional A/A experiments, and even A/B/n experiments in a reduced arm space. A/A experiments allow the experiment organization to not only test run sample ration mismatch, but also other quality checks to validate the instrumentation and randomization system before moving towards a MAB. However, the MAB algorithm implementation is not validated. A/B/n experiments in a reduced arm space allow the organization to determine the best arm and then proceed to checking MAB in this limited arm space. If the MAB confirms the results of the A/B/n experiment, the experimentation organization can proceed with more confidence to a MAB algorithm in the full arm space. The disadvantages of this strategy are the extra time and resources taken to validate a MAB experiment prior to its launch. The algorithms implementation can be validated against multiple theoretical test and use cases, as done by Company C.

8.4.5 Increased complexity in ramp-up procedures in MAB algorithms

8.4.5.1 Restriction

Ramp-up is a risk-minimization technique [2], where the treatment variations are only launched to a small percentage of the population. If the experiment does not present any severe degradation, the percentage of users that are exposed to the treatment is increased until the variations have equal allocation to maximize the power of the experiment [2]. If the experimentation system detects large movements (that are typically related to experiment errors [73]), it can stop the experiment execution minimizing the impact on users. For example, in an A/B experiment, a ramp-up procedure can start assigning 10% of the users for the treatment while 90% continue in the control group. As confidence is gained in the experiment, the organization slowly increases the allocation to the treatments. LinkedIn reports using in four manual ramp-ups, with average duration of 6 days for each iteration [23].

Most MAB algorithms will allocate a very small slot to bad variations and therefore minimizing the exposure to bad variations. Therefore, it is often thought that MAB can replace ramp-up procedures. However, they will also allocate a large slot to metrics that have a large positive movement. Large positive movements in metrics are rare [73] and can indicate instrumentation problems. Extra steps should be taken to avoid bias if MAB is combined with ramp-up. If variant consistency is kept, the algorithm can be biased between steps and converge to a suboptimal variant, similarly to the novelty effect in section 8.4.3. This restriction was identified by company B, but also discussed with companies C and D.

8.4.5.2 Strategy

The later reassigns the users to a new variant at every step of the ramp-up. While this solution minimizes the bias in the final step, it will take longer for the algorithm to converge to a solution, the regret minimization is lost in each step and it might create user experience problems.

We present three approaches suggested by companies C and D to deal with ramp-up in MAB. The first alternative is the use of algorithms that permit adjusting the exploration step (ϵ-greedy variations) with user consistency. This solution can be combined with ramp-up procedures, where, the experiment starts with a high level of exploitation, and at every ramp-up step, the exploration rate is also reduced. This approach has the disadvantage of reducing the regret minimization. A second approach is to provide a hard allocation boundary for all treatment variations. This approach should be carefully designed in such a way that it does not degrade the exploration and the decision process as the percentage of users allocated to the experiment increases. The third approach is creating a hard reset for each ramp-up step. This will reset the variant assignment between ramp-up steps and allow the system to re-explore. While this solution minimizes the bias in the final step, it will take longer for the algorithm to converge to a solution, the regret minimization is lost in each step and it might create user experience problems.

One principal data scientist in company B suggests avoiding the first ramp-up in MAB-based experiments and instead run pre-quality checks with and an A/B experiment with ramp-up. If the no problems occur, then the MAB can be launched.

8.4.6 Increasing regret in experiments due to Simpson's Paradox in MAB algorithms

8.4.6.1 Pitfall

The Simpson's Paradox [110, 174] (SP) refers to the observation that "a statistical relationship observed in a population – i.e. a collection of subgroups or individuals – could be reversed within all of the subgroups that make up that population" [174] . Kievit et al. [174] provide several examples of Simpson's Paradox observed in experimental in the areas of social science, medicine and biology. SP is also discussed in the context of online experiments [43, 110, 158]. The following example adapted from Crook et al. [158] illustrates SP in the context of online experiments.

Suppose an experiment on a website text is implemented in more than one country (UK and Sweden). The experiment consists of reducing a long sentence instruction to a short direct instruction. This experiment is launched in the two countries using the same sample size. The power calculation indicates that the UK would run with 10% of users assigned to the treatment and in Sweden it requires 50% in the treatment group. It is possible that the treatment variant performs better in both countries but when both countries are combined the control variant performs better. However, note that the SP can occur just in one or all countries (subgroups). Kievit et al. [174] provide a guide on how to identify SP and discuss ways to minimize the occurrence of SP. However, there is no single mathematical property that SP instances have in common.

In presented example, a confounding variable (the country) makes the randomization process not randomly distributed, in UK it samples a smaller percentage of the user group. In the (unknown) presence of the SP together, MAB algorithms reinforce the selection of the suboptimal variant, increasing the regret of the algorithms. Although SP is also a problem in other online experiments such as A/B experiments, the identification of the sampling bias conditions [43, 158] is aggravated by the non-randomly distributed form that MAB allocate the users to the variations.

This pitfall was identified by Company B, although they report that they haven't made an in-depth analysis, as MAB is not the core part of their experimentation strategies. Company A discussed the presence of SP as a problem seen in their A/B experiments.

As their MAB applications are aimed in known situations after A/B experiments, SP can be identified in stage prior to the implementation of MAB. However, they argue that the presence of SP would invalidate the goal of minimizing regret in MAB applications.

8.4.6.2 Simulation

Next, we present a simulation illustrating the usage of MAB in the presence of SP. This simulation is based on the hypothetical SP case presented by Lihoung Li in [175]. Suppose we are recommending two news articles sports, movies to users male, female. The confounding factor (due to an algorithm bias) is that male users receive more sport news articles (75%) than female users (25%), and female users receive more movies news articles (75%) than male users (25%).

Table 8.1 below shows the click-through rate for each subgroup and the general group. From the table we can see that the sports perform better than movies for both male and female, but movies perform better if the results are aggregated. The SP case is compared to the case (called no-SP) where the selection between male and female are not confounded with the group sports or movies. In this case the sports group will have an overall CTR of 0.6, and the movies group will have an overall CTR of 0.5. Both the cases with and without SP are simulated using the algorithms UCB1 and Epsilon-Greedy 10%. Figure 8.3 shows the cumulative regret with a horizon of 10,000.

This simulation shows that the presence of an unknown SP in MAB can lead

CTR per group	Male	Female	Overall
Sports	0.4	0.8	(0.4x0.75+0.8x0.25) = 0.5
Movies	0.3	0.7	(0.3x0.25+0.7x0.75) = 0.6

Table 8.1: SP case for the simulation

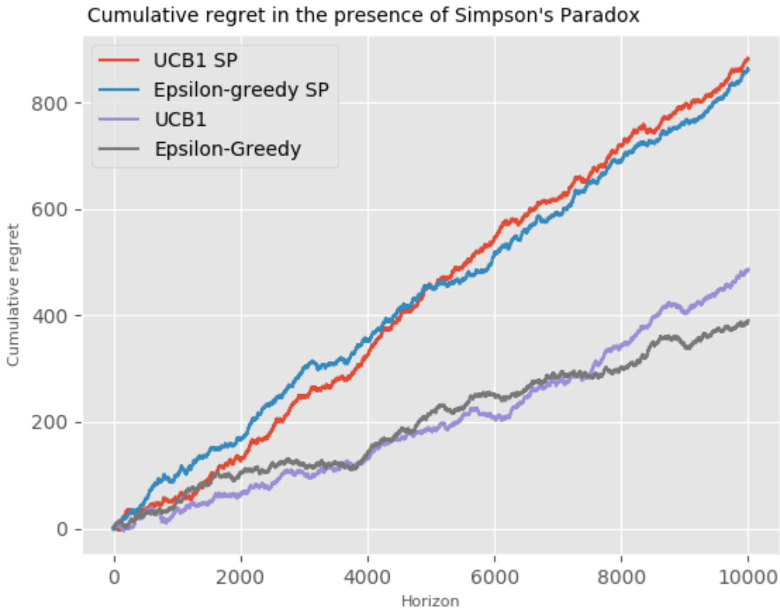

Figure 8.3: Comparison of the cumulative regret in MAB algorithms in the presence of SP and without SP. In the picture lines red and blue indicates the cases where SP is present and lines purple and brown indicates the cases without SP.

to a significant increase in the regret and therefore minimizing the advantage of this kind of algorithm in this situation. The cumulative regret grows linear with the horizon, but at different rates depending on the presence of SP. This suggests that introducing multi-armed bandits in long-term experiments in the presence of SP can lead to unexpected outcomes in terms of the predicted regret bounds.

8.4.6.3 Strategy

The identification of SP is not a trivial task, as there are several types of SP and there are no shared property between them [174]. The foremost recommendation by Kievit et al. [174], is not to assume that relationships at the group level also hold for subgroups or individuals over time. This analysis requires incorporating in research designs data collection to facilitate the comparison of patterns across the different group levels. Crook et al. [158] illustrate some situations where SP were identified in online controlled experiments and give a caution against of combining metrics over type, especially when the proportions of the control and treatments vary, or when the subpopulations are sampled at different rates. Traditional experiment designs offer a better framework for identifying SP compared to MAB, as it is possible to run A/A test and sample ratio mismatch tests for each segment group. Alternatively, running A/B experiments and segmenting the data collection prior to the MAB experiment can help practitioners to identify cases where there is a suspicion of SP.

If there the organization is aware of an existing SP, common strategies are eliminating the confounding variable (if it is a sampling error bug), choice of different statistical tests, eliminating the data that generates the SP (i.e. when SP appears in ramp-up situations). If the SP is intrinsic to the problem, the confounding variable is identified and minimization of regret is still desired the experiment organization can introduce an exogenous contextual variable (representing the confounding variable) in a contextual bandit algorithm [115].

8.4.7 Adaptive allocation based on a single metric

8.4.7.1 Restrictions

The decision-making process of online experiments involves the analysis and trade-off of several metrics, from both feature-level to product-level metrics. Well-developed experimentation organizations use multiple metrics in their decision process: a composed success metric, the Overall Evaluation Criteria [32], guardrail and data quality metrics. During the decision process of an experiment several stakeholders are involved in order to understand the implications the different metrics for the business and the software system. Small changes in the system are unlikely to make changes in an OEC metric, so companies often utilize other lower-level metrics to make a business decision regarding the modification.

In a MAB-based experiment, a solution is exploited from the beginning of the experiment based on the success metric (the reward). This prevents the usage of other supporting metrics for an in-depth analysis. This can bias the algorithms towards an arm in a way that is not totally comprehensible and that can potentially some business goal. Additionally, incorporating additional metrics and a traditional statistical analysis on top of the MAB can be compromised because of the lack of power in some arms. Finally, some system level metrics can have a lower sensitivity and the delay in the reward can lead to assumptions problems, as discussed in the subsection 8.4.3.

Suppose a company was evaluating multiple new layouts for the home screen of a news website using a MAB approach. Each new variation layout could be a combination of different grids varying the number of rows and columns. If only a composed metric such as overall customer satisfaction is being used, the decision of the best arm could exclude the analysis of other important metrics. For example, the new variant customer satisfaction might favor free users instead of subscribers. Some negative impacts might not be seen during the experiment duration, but it was identified in previous experiments by company and was set as guardrails metrics. As an example, the new variant might significantly increase loading time for mobile users. Initially users might attribute this to their connection, but if this persists for a time longer than the experiment duration and competitors are providing faster loading time, mobile users can move to new services. Although these restrictions apply also to traditional experiments (if the additional metrics are not being measure or used in the decision process), MAB incorporate the decision to exploit an arm during the execution time. Therefore, potential bad variations might be overly exploited while hurting metrics that could be used in traditional analysis. This restriction was identified by all companies. Company C and D recommend avoiding MAB if their customers have multiple metrics.

> *"...if more than one metric is present, and our customers can't combine and generate a single metric for the bandit, we recommend them to avoid this feature at all"* — Product Manager Company C

The difficulty of incorporating other metrics in the exploitation decision is one of the reasons that prevents Company B of applying MAB in more general experiments, as highlighted by one of the interviews:

> *"Despite the hype with bandits, in our workflow they can only be used in very narrow cases where we know very well what we want to optimize."* — Principal Data Scientist Company B

8.4.7.2 Strategy

The first strategy is the use of MAB extensions for multi-objective optimization with Pareto relations. However, such solutions add technical complexity and multiple failure modes to an experiment. Also, providing a validated Pareto curve to a MAB algorithm requires a deep understanding of the system and the users that not all companies have. This solution is discussed mainly in academic setting [176] and none of the companies reported using this solution.

A second strategy is to use the current Overall Evaluation Criteria that is being used in A/B experiments. However, this metric must have sufficient sensitivity and be a leading indicator [72], so it does not fall in the case of the delayed reward pitfall described in section C. Company C recommends this approach to their clients in specific experiments. However, they suggest to run a MAB-based experiment in a smaller fraction of the user base and use the winning arm in an A/B experiment against the current variation to check against all other metrics

A third approach suggested by Company D, is to add the additional information from other metrics to a contextual bandit algorithm as exogenous variables. However, this solution has the same drawbacks discussed in section C. The first is to have a good understanding of the problem as this solution is not exploratory. The second is that the external context also needs validation, so it does create bias towards a particular context.

8.5 Discussion

Feature experiments, powered by MABs, can provide a competitive edge for organizations, but only when skillfully applied. Several potential pitfalls can hinder the benefits of using MABs. For example, popular experimental models, such as the HYPEX [5] or the RIGHT [7], may not be well-aligned with MAB-based experiments. In particular, these models often assume that the experimental process should minimize type I errors (false positives) instead of minimizing regret (opportunity cost of missing the best solution). Further, when sufficient users can be obtained, traditional experimental models control for both type I and II errors (false negatives), whereas MAB-based experiments have less statistical power. Experimental tactics designed for traditional A/B experiments (ramp-up or user consistency), if directly applied in MAB experiments, can result in a more complex design and higher practical significance requirements, which may ultimately reduce the value of MAB. Further, misaligned experimental goals can lead to suboptimal experiments. For example, if the goal is to explore user behavior in different feature variations, then A/B/n or full factorial experiment may be more appropriate.

The introduction of MAB-based experiments into production environments yields more considerations. Experimental systems require complex modifications such as user consistency and reassignment, new assignment and randomization procedures among others. Like, the introduction of any other machine learning algorithm, MAB algorithms can increase technical debt, maintenance costs, and introduce uncertainty in its reliability [104]. Due to scalability issues [104], often in direct feedback loop problems (where the algorithm influences the selection of its own future training data), it is common practice to use standard supervised algorithms instead of bandit algorithms.

Finally, cultural and training barriers can make the adoption of MAB-based experiments more difficult. Stakeholders may find less transparency in the results of MAB-based experiments or may not properly understand its associated limitations.

> *"One challenges I have seen in bandits is: how do we communicate the results to other people? If we want to adopt it, we need to train our organization to understand not only MABs but also machine learning in general"* — Senior data scientist Company E

In contrast, A/B experiments are better understood by managers, developers, and product owners. To overcome these barriers, some companies, such as B, provides training in feature experimentation for the employees once a month, in addition to tailored training for product teams. In the future, specialized training can be given to product teams who could benefit from MAB, as well as generalized training if MAB becomes more widespread.

A summary of restrictions, pitfalls, and strategies associated with MAB algorithms can be seen in Table 8.2.

8.5.1 Use cases for multi-armed bandits

Despite these limitations, MAB algorithms still can provide several benefits in appropriated situations; especially, when regret minimization is an essential property.

8.5.1.1 Content-serving systems

Content-serving systems are designed to select and display content to users, such as newspaper headlines, ads, or songs [177]. The goals of this type of system is to provide to the users the most relevant content from a pool of possibilities. For this type of system, failing to recommend relevant content

Pitfalls and Restrictions	Reasons	Strategies
Increase in type I error	Naïve implementations of MABs	• Adding a statistical framework on top of the MAB implementation • Usage of the MAB in applications where the cost of making a type I error is low • A/B experiments
	Assumption violations	• Contextual bandits, re-weighting scheme, delayed reward bandits, long-term optimization. These strategies require a prior understanding of the problem and the user behavior. • A/B experiments
	Using MABs in exploration problems	• Design of experiments: A/B/n, full factorial, or fractional factorial experiments
Detecting experimentation problems	Lack of sample ratio mismatch in MABs	• Prior A/A experiment • A/B/n experiment in a reduced arm space
	Increased regret in MAB with Simpson Paradox	• Prior A/A and A/B/n experiments • If identified, contextual bandits can be used
	Complexity in ramp-up procedures	• MAB algorithms with adjustable exploration rates • Hard allocation boundaries • Hard reset of the algorithm for each ramp-up iteration
Increased design complexity	Adaptive allocation based in a single metric	• Composed Pareto relations between metrics • Increase sensitiveness in the Overall Evaluation Criteria (OEC)
	User consistency	• Prior A/B/n experiments • Variation reassignment procedures
	Communicating experiment results	• Capacitating the organization in machine learning and MABs • Providing a similar presentation layer as A/B experiment results

Table 8.2: Summary of the restrictions and pitfalls

that could lead to a conversion or click is costlier (type II error). Furthermore, the content-serving system should be able to operate autonomously without expensive manual intervention. For both these reasons, an A/B algorithm would be inappropriate. In contrast, a moving window MAB would be well suited: exploiting the arm with the highest return, while exploring arms that can have higher potential reward in when the content pool is updated.

8.5.1.2 Short-term campaigns

Short campaigns refer to a limited time presentation of a variant and after the campaign ends, the system returns to the original variant. Examples of short-term campaigns are emailing campaigns [178] and a Christmas advertisement campaign. The advertisement has fixed time window to be displayed (between Black Friday and Christmas). Again, failing to display the best ad that leads to a higher conversion is costlier (type II error). In such situations, A/B experiments fail to exploit enough in the limited period of time. This case is well suited for MAB-based experiment, where the it is desired maximize the cumulative reward of the short-term campaign.

8.5.1.3 Targeting experiments

Sometimes, organizations already have prior knowledge about preferences across user segments. Using this knowledge, organizations can run targeted experiments. Traditional A/B/n and factorial experiment designs do not support these situations because of their uniform variant assignment process. However, contextual multi-armed bandits have the ability to incorporate this prior-knowledge into the experiment design. For example, Yahoo News has integrated contextual bandits [115, 179] in a content-serving system to provide personalized content for identified segment groups. Targeting experiments are also part of online ad placement engines [110].

8.5.1.4 Other cases

When the discussed limitations are analyzed and addressed, MAB can be used in online experiments to achieve a faster decision regarding the best arm when the effect size and the difference in mean-reward is considered high. The algorithms used in this situation are based on variation of ϵ-greedy algorithms, to ensure that the other arms will still have been explored enough for a proper comparison. A known implementation for this case is present in Google Analytics [167, 169].

Another use of MAB is in controlling false discovery rate in sequential A/B experiments [24, 180]. In this case, a sequence of A/B tests are replaced by a sequence of best-arm MAB instances. This allows the complexity to stay relatively low, with high power experiments and low false-discovery rate.

8.5.2 Guidelines

The selection of the appropriated experimental design can avoid most of the pitfalls. In order to aid future practitioners in designing future feature experiments, we have summarized our results into guidelines shown in Table 8.3, which can be used when planning a new experiment. The guidelines contain five questions which can help in making a decision of the appropriate experimental design, including specialized MAB algorithms as well as traditional A/B/n or factorial experimental designs.

8.6 Conclusion

Delivering faster value to customers with online experiments is an emerging practice in industry. MAB algorithms have the potential to deliver even faster results with better allocation of resources over traditional A/B experiments. This work describes common models, paradigms, and algorithms for MAB-based feature experiments currently used industry. Based on a study with 11 experts across 5 companies, we identified potential mistakes that can occur when designing a feature experiment and several strategies for avoiding them. We synthesize these results into practitioner guidelines, which provide criteria for selecting and using particular types of MAB algorithms or in some cases, when more conservative approaches are desired, using the traditional A/B/n algorithms.In future work, we plan to explore recommender systems that can provide interactive feedback for online feature experiments. Further, we will explore ways to seamlessly include MAB and A/B experiments, thus allowing consensus between algorithms or fallbacks in the event of a failure or limitation is encountered.

8.7 Appendix

8.7.1 Multi-Armed Bandit algorithms used in the simulations

The presented algorithms were chosen because different implementations and more complex algorithms are based on strategies similar to those present in these algorithms. An extensive review of different MABs and extensions is provided in the surveys by Burtini et al. [143] and Kuleshov and Precup [181].

8.7.1.1 Explore-First algorithm with parameter N

The simplest algorithm for the MAB problem is the explore-first strategy with parameter N. This strategy consists of uniformly exploring all the arms equally for a finite horizon of NK play, where each arm is played N times, and K is the number of arms. After NK plays, the arm with the highest average reward is selected. Ties between arms are broken arbitrarily.

[a] Exploration: In this step, all arms are selected with a uniform probability during the first NK rounds.

$$a = U(a_i), \text{ Arm } a \in \{a_1 ... a_K\}$$

[b] In this step, the arm with the highest average reward is selected:

$$a^* = \max_{a \in \{a_1 ... a_K\}} \mu(a)$$

Question	Decision
1. What is the goal of the experiment?	• Learning: A/B/n, full factorial and fractional experiments • Innovation: A/B experiments • Optimization: A/B/n experiments, sequential A/B experiments and MABs
2. What is the cost of making type I and type II errors?	• If both types of error are costly: high power traditional A/B/n and full factorial experiments • If only type I error is high: traditional A/B/n, full factorial experiments • If only type II error is high: MABs
3. How well known are the problem and the assumptions?	• If not well-known traditional A/B/n and full factorial experiments • If the system needs validation: A/B/n experiments with pre-quality tests • If well-known analysis if it matches the assumptions of different MABs – If the context is well understood and validated: contextual bandits

4. Is there a single decision metric?	• If there is only one metric
	    – And this metric is sufficiently sensitive, MABs can be used
	    – Delayed metrics and less sensitive metrics: A/B/n experiments
	• If there are multiple metrics that cannot be grouped in single OEC: A/B/n experiments

5. How long will the experiment run?	• Short-term experiments
	    – If there are external deadlines regardless of sample size: MABs
	    – Otherwise: traditional A/B/n and full factorial experiments
	• Long-term experiments
	    – If it is to collect sufficient sample size: A/B experiments
	    – If there is no user-consistency requirement: MABs
	    – If it is to adaptively change the variation with time: contextual bandits and non-stationary bandits

Table 8.3: Guidelines to select and experimentation techniques

[c] Exploitation: In this step, all the remaining rounds are played with the best arm a^*.

This strategy is often compared to A/B/n testing. However, in A/B/n the selection of the best arm follows a statistical procedure with a fixed significance level. The strategy of selecting the highest reward without computing the significance level or minimizing type I errors is, in this text, called the naïve MAB arm selection. The ϵ-greedy algorithm

This algorithm exploits the best arm (the arm with the highest mean reward) with probability $1-\epsilon$ and selects, with probability ϵ, an arm at random (including the current arm with the highest mean reward), uniformly and independently of the arm-value estimates [162]. This solution is equivalent to the one-state Markov decision process problem [143].

In terms of the exploration and exploitation trade-off, this algorithm explores $\epsilon \cdot t$ while exploits $(1-\epsilon)t$ of the time. This strategy can be represented as:

$$a = \begin{cases} \max_{a \in \{a_1 \ldots a_K\}} \mu(a), \text{ with probability } 1-\epsilon \\ U(a), \text{ Arm } a \in \{a_1 \ldots a_K\}, \text{ with probability } \epsilon \end{cases} \tag{8.9}$$

The case where $\epsilon = 1$ reduces the ϵ-greedy algorithm to the ϵ-first strategy.

8.7.2 The Softmax algorithm

In a similar manner to the ϵ-greedy algorithm, the Softmax algorithm also exploits the best arm but explores by ranking and weighting the other arms according to their value estimates. The Softmax algorithm used in this work uses the Boltzmann function to give each arm its probability [172]. The Softmax algorithm has the ability to assign a small probability to the worst arms [113]. Each arm has a probability of being selected depending on the arm mean reward according to:

$$P(a_i) = \frac{exp \mu(a_i)/\tau}{\sum_{i=1}^{K} exp \mu(a_i)/\tau} \tag{8.10}$$

The Softmax algorithm uses a parameter τ called temperature. High temperatures cause the arms to be nearly equi-probable, while $\tau \xrightarrow{0}$ makes the arm selection to become the same as the greedy algorithm. Both the Softmax and the ϵ-greedy algorithms have one parameter for tuning. However, setting τ requires a deeper understanding of the likelihood of each arm value [113].

8.7.3 The UCB1 algorithm

The Upper Confidence Bound (UCB), or worst-case regret bound, algorithm is a different algorithm implementation that not only explores how much reward each arm receives, but also the confidence in each arm. The algorithm balances the exploration/exploitation tradeoff by selecting the arm that has the highest empirical reward estimate. The aim of this algorithm is to minimize regret in making an arm decision [143]. The user-defined policy can be written as:

$$a(t) = \underset{i=1 \ldots K}{\arg\max} (\hat{\mu}_i + \sqrt{2 \ln t}/n_i) \tag{8.11}$$

where $a(t)$ represents the arm selected at the time t. This user-defined policy does not use randomization in the arm selection, but it rather depends on the randomization of the reward. Additionally, the UCB1 does not have free parameters, making it easier to deploy without prior tuning. However, one of its assumptions is that the reward value should lie between 0 and 1 [172].

8.7.4 Further extensions

The MAB and the exploration/exploitation dilemma are widely studied and have several extensions and improvements. The first extension to be considered is the case of contextual bandits. When prior information of how the reward works (e.g. if it follows a known pattern or if it behaves differently from groups of users) is available, an exogenous context variable (x) representing this information can be included in the problem formulation:

$$a = \pi(x, \delta),\ \text{Arm } a \in \{a_1 ... a_K\} \tag{8.12}$$

$$y = r(x, a, \delta'),\ \text{Reward } y \in \mathbb{R} \tag{8.13}$$

A second extension is the use of multiple rewards. As formulated, the MAB problems utilize a one-dimension reward. In the case of optimizing for multiple parameters, also known as the multi-objective MAB, one approach is to utilize Pareto relationships to drive the optimization process [176].

Other extensions utilize Bayesian learning, Thompson sampling, stochastic and non-stationary bandits, and also MABs where the number of arms is considerably larger than the exploration space, or belong to a continuous space [143]. However, several of those extensions are research implementations, described only through simulations and datasets, and have not yet reached industrial use.

Acknowledgments

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The authors also thank the companies and the interviewees involved in this study for the opportunity to conduct this study with them. Finally, the authors gratefully acknowledge anonymous reviewers, whose comments significantly improved this paper.

Chapter 9

Paper E

Statistical Models for the Analysis of Optimization Algorithms with Benchmark Functions

Mattos, D. I., Bosch, J., Olsson, H. H.

In IEEE Transactions on Evolutionary Computation, 2020.

Abstract

Frequentist statistical methods, such as hypothesis testing, are standard practices in studies that provide benchmark comparisons. Unfortunately, these methods have often been misused, e.g., without testing for their statistical test assumptions or without controlling for family-wise errors in multiple group comparisons, among several other problems. Bayesian Data Analysis (BDA) addresses many of the previously mentioned shortcomings but its use is not widely spread in the analysis of empirical data in the evolutionary computing community. This paper provides three main contributions. First, we motivate the need for utilizing Bayesian data analysis and provide an overview of this topic. Second, we discuss the practical aspects of BDA to ensure that our models are valid and the results are transparent. Finally, we provide five statistical models that can be used to answer multiple research questions. The online appendix provides a step-by-step guide on how to perform the analysis of the models discussed in this paper, including the code for the statistical models, the data transformations, and the discussed tables and figures.

9.1 Introduction

With the increasing number of optimization algorithms being developed, benchmarks are used not only to show that the new algorithms work but also to compare algorithms against each other [61, 62, 182, 183].

Over the years, algorithms for black-box optimization have been demonstrated to work and have been compared with each other using simple descriptive statistics (such as mean, standard deviation, and median), boxplots and performance profiles [184]. Recently, frequentist statistical methods such as null hypothesis testing have become standard practice in papers that provide benchmark comparisons [185–187].

Unfortunately, frequentist methods for null hypothesis testing have often been misused by scientists and practitioners looking for a dichotomy tool to assess a particular problem, without an evaluation of the size of the observed effect or a discussion with complementary analysis [86]. By utilizing statistical tests as black-box tools, many pitfalls and misuses have been observed in different fields of science. We list some of the observed pitfalls. (1) Lack of separation between the effect size and sample size in the p-value [87]; (2) Lack of information regarding the null hypothesis [87–89]; (3) Misinterpretation of the actual meaning of the p-value (including by instructors in statistics) [89–91]; (4) Misinterpretation of the meaning of confidence intervals [92, 93]; (5) Lack of transparency in the reporting of the statistical procedures (such as providing the value of test statistics, the actual value of the p-value, confidence intervals) [92]; (5) Common problems related to the misuse of the statistical tests such as not verifying the statistical test assumptions, not controlling for correlated samples or not controlling for family-wise errors in multiple group comparisons [94].

Bayesian Data Analysis (BDA) [95] has gained attention as a potential replacement for frequentist statistics by providing an easier to interpret data analysis framework and by addressing many of the previously mentioned shortcomings. Despite the popularity of Bayesian-based optimization algorithms, BDA has not been widely used for the analysis of benchmark data in Evolutionary Computing. With this paper, we argue for the adoption of BDA in evolutionary computing, in particular in the analysis of benchmark data. Specifically, we provide an overview of this topic discussing practical aspects of BDA to ensure that our models are valid and the results are transparent to avoid misuse and pitfalls. Focusing on the interpretation and answering specific research questions, we provide five statistical models together with a reproducible appendix that contains a step-by-step analysis, including the code for the statistical models, the data transformations, and the discussed tables and figures. We reinforce that BDA is not a black-box framework for decisions. BDA requires convergence checking, validation of the statistical models, reporting transparency and it is complementary with other types of analysis such as plots and tables. A correct interpretation of the results still requires domain specific knowledge and understanding of the statistical models and their limitations.

The remainder of this paper is organized as follows. Section 9.2 discusses related work. Section 9.3 provides an overview of Bayesian data analysis, including a discussion of practical aspects of Bayesian data analysis to ensure that our models are valid and the results transparent. Section 9.4 provides a description of the empirical data used to exemplify all the statistical models. Section 9.5 provides a discussion of each statistical model, interpretation and presentation of the results. Section 9.6 concludes this paper.

The online appendix can be found at: `https://doi.org/10.5281/zenodo.4067712`

9.2 Related work

In this section, we provide an overview of related works in statistical analysis for the development and comparison of algorithms.

In the frequentist setting, the work by Eftimov et al. [188, 189] provides an analysis of different ranking schemes for analysis of benchmarking in evolutionary computation, concluding that different statistical tests can lead to different ranking schemes. To overcome this, they propose the use of a different scheme that uses the whole distribution instead of only the average or the median and a new analysis method. The proposed approach is shown to be more robust to outliers and the ranking scheme. However, the proposed approach is used only in ranking situations and it does not take into account additional covariates and the internal correlation between the data given by the benchmark functions. Despite the focus on the practical significance, it is still subjected to the other problems of frequentist analysis [93].

Gagliolo and Legrand [190] provide an overview of the survival analysis to runtime distributions, discussing its usage in the algorithm selection. The paper discusses the application and interpretation of the frequentist variation of survival models such as the parametric Cox Proportional Hazard model and the non-parametric Kaplan-Meyer estimator. In Section 9.5.5. we discuss survival analysis and its application in the context of Bayesian Data analysis. Chiarandini and Goegebeur [191] discuss the use of frequentist linear and multilevel and models for different experimental designs. They reinforce the need to separate the effects of the algorithms from the problem instances by modeling the instances with random factors. All the statistical models in Section 9.5 utilize random factors to separate the effects of the benchmarks from the algorithms.

Bartz-Beielstein et al. [63] provide an extensive survey that discusses different topics for promoting good benchmark practices, from objective statement and selection of problems to the analysis and presentation of results. However, from the analysis perspective, the paper focuses solely on the use of frequentist statistics, while the known limitations and pitfalls of frequentist statistics are not considered and BDA is not mentioned.

In the Bayesian setting, the work by Calvo et al. [192] provides the first paper for Bayesian estimation in evolutionary computing. In the paper, they provide a practical application of Bayesian data analysis on the comparison of eleven algorithms on 23 optimization problems. The authors discuss the Plackett-Luce model for ranking algorithms and compare it with the frequentist approach. The algorithms are analyzed in stratified benchmark functions (e.g. easy, medium, hard). In our work, we discuss an extension of the Bradley-Terry model that, under some conditions, is equivalent to the Plackett-Luce model for complete ranks but we also consider the random effects of the benchmarks.

Furia et al. [93] provide a discussion of frequentist and Bayesian data analysis in empirical software engineering (including a re-analysis of early research). They discuss many of the shortcomings of frequentist statistics and provides an introduction to Bayesian data analysis from the Software Engineering perspective.

Carrasco et al. [193] discuss the application of both frequentist and Bayesian non-parametric tests in the comparison of machine learning algorithms. They discuss three Bayesian tests, a variation of the t-test that takes into account correlation in the results, and two nonparametric tests. The discussed Bayesian tests are based on the work of Benavoli et al. [194, 195] that provide closed formulas for the nonparametric tests under specific prior conditions. The work by Lacoste et al. [187] discusses a Poisson binomial test to compare algorithms and demonstrates that the approach is more reliable than the sign test and the Wilcoxon signed-rank test.

In our work, we base our analysis on parametric models that are of interest for "scientific experimental analysis, where the interest is in explaining the causes of the success of a certain optimization approach rather than in mere comparative studies" [191]. At the expense of computational resources, our models can utilize flexible priors since the posterior is computed numerically (instead of analytically) by a Markov Chain Monte Carlo (MCMC) sampler. Additionally, our models take into account the correlation by the benchmarks and can be easily extended to additional correlations either nested or on the same level.

9.3 Bayesian Data Analysis

In this section, we provide a short overview of the Bayesian data analysis process and some practical aspects to ensure that our models are valid and our results transparent. A full comparison between the Bayesian and the frequentist framework is beyond the scope of this paper and we refer to other sources [88, 93, 196, 197].

The main idea behind Bayesian data analysis is the reallocation of credibility across possibilities [88]. In practical terms, this means that we start with a prior explanation of the results before seeing any data and a model on how the data is generated. As we collect new data, our beliefs about the system are reallocated. The probability of candidate explanations that do not fit well the data is therefore reduced. In this updating process, we get a probability distribution of each possible explanation of the data. This allows us to obtain the credible (or uncertain) intervals [93, 95].

The process of allocating explanations into probability distributions happens through the principles of conditional probabilities and the Bayes theorem [93]:

$$\mathcal{P}(h|d) = \frac{\mathcal{P}(d|h) \cdot \mathcal{P}(h)}{\mathcal{P}(d)}, \tag{9.1}$$

where d represents the data, h the explanation (or hypothesis), $\mathcal{P}(h|d)$ is the conditional probability of the hypothesis given the observed data. Below are common names for the factors in the Bayes theorem:

- $\mathcal{P}(d|h)$ is called the likelihood of the data d under the hypothesis h.

- $\mathcal{P}(h)$ is called the prior.
- $\mathcal{P}(h|d)$ is called the posterior. The posterior represents the probability distribution of each parameter estimate (the hypothesis h) given the observed data
- $\mathcal{P}(d)$ is called the marginal likelihood and it is a constant, that is often impossible to compute analytically.

9.3.1 Bayesian tools

Several tools are capable of performing Bayesian data analysis, such as IBM SPSS, SAS, Stata, JASP, R, TensorFlow, Stan among others. Although the discussed models can be used in most statistical software, we utilize the Stan software and its modeling language [198], while we utilize the R language for data transformation and plotting. Stan can be easily integrated with R, with the `rstan` package[1]. Many R packages make modeling in Stan easier (e.g. `rstanarm`[2] and `brms` [3]). However, in the online appendix, we decided to provide the raw Stan model, since it can be used together with many different programming languages such as Python, Stata, Julia, Matlab, where readers can perform the data transformation and plotting in their preferred language while utilizing the same statistical models that we provide.

9.3.2 Bayesian inference and MCMC

While the Bayes theorem provides how our initial beliefs are going to be updated based on the observed data to generate the posterior, the actual computational process is more complicated due to the marginal likelihood. The posterior can be approximated without explicitly computing $\mathcal{P}(d)$ by using a class of sampling algorithms called Markov Chain Monte Carlo (MCMC). The goal of an MCMC is to generate an accurate representation of the posterior of the model parameters [197] by generating a large representative sample of credible intervals that represents the posterior distribution [88]. Although there are several MCMC algorithms, we utilize in this work the Hamiltonian MCMC No U-Turn algorithm (NUTS) [199] available in the Stan program [198], as it provides faster convergence, handles correlated parameters in the posterior better than others, and provides good diagnostic tools when the chain diverges.

One of the main disadvantages of Bayesian data analysis is the time necessary to compute the posteriors of an MCMC process compared to maximum likelihood estimates from the frequentist approach.

9.3.3 Posterior and intervals

One of the benefits of the Bayesian approach is that we get a posterior distribution of the estimated parameters from a model. With the posterior, we can get not only point estimates (such as the mean or median of a parameter) but also credible intervals (also called uncertain intervals) of these parameters.

[1] `https://cloud.r-project.org/package=rstan`
[2] `https://cloud.r-project.org/package=rstanarm`
[3] `https://cloud.r-project.org/package=brms`

These intervals are useful to estimate the uncertainty of the parameters without making assumptions of repeated sampling, such as the confidence intervals given by the frequentist approach (that assumes that a fixed point estimated will lie within an interval if the sampling process is repeated many times and assuming that the null hypothesis is true). The credible interval is a probabilistic statement about the real value of a parameter while the confidence interval reveals only the uncertainty about the interval (if it contains the value or not). Confidence intervals, due to their assumptions, cannot be used to understand the probability of a point estimate parameter.

In Bayesian data analysis, we make use of the full posterior of the parameter to make interval inferences instead of single-point estimates. To help our analysis, three common intervals are used:

9.3.3.1 Equal tail interval

this is a credible interval that divides the posterior lower and higher tails equally, based on quantiles. For example, the 95% interval will exclude 2.5% of the data in each tail. This is the default interval that Stan provides after sampling in R.

9.3.3.2 Highest Posterior Density (HPD) Interval

this is the narrowest interval in a unimodal distribution that will contain the specified probability mass, the area under a density distribution. This is the interval that best represents the parameter values consistent with the data [196]. Throughout this paper, we will present the HPD intervals for the parameters we estimate.

9.3.3.3 Region of Practical Equivalence (ROPE)

is a practical interval that encloses the values that are considered negligible from a practical perspective [197]. This interval combined with a credible interval like the HPD interval can be used for decisions. For example, if from a practical perspective an improvement or degradation of an algorithm of $x_0 \pm 10\%$ around a baseline x_0 is considered irrelevant, this is the ROPE interval. If all or almost all of the HPD interval (given a threshold such as 95% of the interval) falls in the ROPE interval we can say that the algorithm doesn't provide practical improvement. If the HPD interval falls above or below the ROPE interval we can say that there is a real improvement or degradation respectively. If there is a large overlap, we cannot make an accept or reject statement like that. Note that different to the frequentist approach, with such tools you can accept the null hypothesis or reject it and do so without being concerned with one or two-tail hypothesis and the family-wise error. It is worth noting that the ROPE interval is highly dependent on the practical values that determine it and the scale of the parameters (e.g. they are at different scales in a binomial and a normal regression).

To avoid misuse of this interval and make mistakes similar to the ones often seen in the null hypothesis significant testing, the use and reporting of the ROPE interval should be explicitly specified and justified. Otherwise, it is preferred that it remains unspecified to allow readers to use and assess the results with their own ROPE intervals [197]. In Section 9.5, we opted to omit the ROPE intervals.

9.3.4 Model checking

Since we can get inference from prior-to-posterior on any reasonable model we should perform additional model checking procedures to ensure the validity of the models. Therefore, in a good Bayesian data analysis we should check for proper convergence of the MCMC, for the adequacy of the model fit with the data, and check for the model robustness against different modeling choices [95].

9.3.4.1 Sampling convergence

After specifying the model, we need to specify the sampling parameters and assess the convergence of chains for valid inference of the posterior. To allow diagnostics of the sampling process, we follow the recommendations of the Stan software[4] and initialize the sampling with four chains, random initial values and target Metropolis acceptance rate equals to 0.8. For the number of iterations and warmup, we adjust accordingly to the number of effective samples of the posterior and whether there are divergent iterations.

Trace plots of the chains: These are diagnostic plots to look at the sampling of each chain. All chains should be well-mixed without any pattern or trend [196].

Number of effective samples of the posterior n_{eff}: Markov Chains are typically autocorrelated, which will result in the autocorrelation of the samples. The number of effective samples of the posterior indicates the number of independent samples. Stan provides warning messages if there is a low number of effective samples in the posterior. As a rule of thumb [196], 200 effective samples of the posterior are enough to estimate the mean of a parameter but we might require more if estimating quantiles or highly skewed posteriors.

Gelman-Rubin potential scale reduction ($\hat{R}$): this statistic measures the ratio of the average variances of samples within each chain to the variance across chains [200]. This is a parameter that indicates the convergence of the chains. If the chains have not converged, the $\hat{R}$ will be greater than one. In practical terms, we require values of $\hat{R} < 1.05$ and preferably $\hat{R} < 1.01$ [196]

Number of divergent iterations: During the sampling procedure we specify a warmup period in which the sampler learns which parameters to use. During this period, we can have divergent iterations. However, after warmup, there should be no divergent iterations. If there are, the posterior estimates cannot be considered valid.

9.3.4.2 Choice of priors

One aspect commonly discussed and criticized in BDA is the subjectiveness of the priors. Priors are part of the modeling flexibility that BDA provides to researchers. It adds the possibility of incorporating prior knowledge of previous research on the model to create better and more robust models. For example, a prior indicating that a parameter should be within a range of -10 to +10 might

[4] `https://mc-stan.org/users/documentation/`

be added o constraint the parameter value. This is often a more reasonable approach for the vast majority of cases than allowing a parameter to have a range between $-\infty$ to $+\infty$ (as the frequentist data analysis does).

This flexibility also allows researchers to choose between *non-informative*, *weakly informative*, and *informative* priors for their models. A non-informative prior is based on bounded or unbounded uniform distribution and does not aggregate any information to the posterior.

Weakly-informative priors are those that do not impact or aggregate much information in the posterior parameters but it is not as vague as the non-informative prior. They can act as regularizing priors and improve the inference and convergence of the MCMC [196]. An example of such a prior would be a normal distribution with a large variance compared to the expected parameter value.

Informative priors are those that incorporate previous knowledge on the subject to improve the model. These priors impact the posterior parameters. As a rule of thumb[5], if the posterior standard deviation of a parameter is more than 0.1 times the prior standard deviation, the prior is considered informative. A classification of the priors in informative and weakly informative is not only a matter of the prior distribution (and its parameters) but the joint effect of the prior, the number of parameters in the model, and the amount of collected data. For small datasets, the prior will have a larger influence in the parameter estimate compared to larger datasets. Therefore adjusting the prior distribution parameters to be weakly-informative should be done together with the actual data and model in question.

For all the models in Section 9.5, we adjusted the priors and hyperpriors to be weakly-informative priors based on the presented rule of thumb.

9.3.4.3 Model comparison

For the same data, we might have different valid model candidates (including different priors and likelihoods), and we should compare these models and verify their performance. A recommended approach is to start with simple models and start building more complex models. If the complexity does not increase the predictive accuracy and it is not justifiable theoretically, simpler models are preferred. Comparing the predictive accuracy can be done by analyzing the model entropy information with the Watanabe-Akaike Information Criteria WAIC [201] or the Leave-One-Out Cross Validation method (LOO-CV) [95]. The calculation of the WAIC and the LOO-CV requires the log-likelihood, which is not calculated automatically in Stan. However, the models we use, and available in the repository, calculate this value.

Note that, one should not use the AIC or BIC criteria in the Bayesian context since both methods assume that the model utilizes flat priors and the maximum a posteriori estimate [196]. These assumptions are often not true since flat priors are discouraged and the estimation method is the MCMC. Information criteria such as the WAIC provide results equivalent to the AIC (when assumptions are met) and can also be used under different priors and estimation methods [196].

[5] `https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations\#how-informative-is-the-prior`

9.3.4.4 Sensitivity analysis

During the development of a model, many alternative models might be considered, including different choices of likelihood, prior, predictors, etc. Sensitivity analysis is a process to evaluate how much the posterior inferences change when we change different aspects of the model. For example, we might have different choices of priors. With sensitivity analysis, we can evaluate the impact of these priors in the inferences we make. If after the modifications, the inference results remain unchanged, we can say that the posterior inferences are robust. We can perform a sensitivity analysis directly on the posterior parameters, verifying if they still have similar magnitudes and directions, but also in terms of their posterior predictive checks. Sensitivity analysis overcomes the critique of the subjectiveness of BDA with transparency [196]. We provide an example of sensitivity analysis in the online appendix.

9.3.4.5 Posterior predictive checking

Finally, the last step to analyze the validity of a model is through a posterior predictive check [95]. Posterior predictive checking is a way to assess how large are the residuals of the model, i.e. the difference between the predictive values of the model compared to the observed values. If the model does not predict or can't explain the data well, it might not be a good or even valid model.

9.3.4.6 Sample size and power analysis

In BDA analysis, it is also possible to perform both prospective and retrospective power analysis without some of the criticisms of frequentist power analysis [197, 202]. Prospective power analysis (such as determine an ideal sample size) consists of generating point estimate hypothesized parameter values for the model and using this model to generate data with different sample sizes [197]. The generated data is used to estimate the posterior parameters of the model. If the HPD interval is contained inside a ROPE interval (that specifies the desired size of the uncertainty interval), for each parameter estimate, we consider that we have satisfied the uncertainty interval restriction with an appropriated sample size. Since this study does not place a restriction on the size of the uncertainty intervals for each model, we do not perform a sample size analysis to collect the data. Therefore, our estimates and uncertainties are in accordance with the data collected as described in Section 9.4.

9.4 The empirical data

In this section, we first present an overview of the algorithms and the benchmark functions that are used to collect the empirical data. Then we present the research questions that guided the development of the models in Section 9.5.

Since the goal of this work is to illustrate the statistical models and not to provide an in-depth comparison of the state-of-the-art algorithms, we created a simplified experimental simulation scenario with enough complexity to fully illustrate the statistical methods. We performed an empirical evaluation of eight well-known algorithms for black-box optimization against 30 benchmark functions under different noise and budget conditions. In this experimental simulation, we focus only on continuous benchmark functions, although the discussed models can be applied and extended to other problems.

9.4.1 The algorithms

The choice of the first six algorithms is based on their widespread use, the computational speed (CPU time) for each function evaluation, and the easy availability of a Python 3 implementation, on which the simulation framework is based. Also, we selected a random search algorithm to be used as a baseline in some of the comparisons, and a variation of the random search (Random Search x2). The Random Search x2 can be used as a baseline for the comparison in the cases in which there is noise in the output value of the benchmark function.

We utilize the following algorithms with their respective default parameters from the software package:

- **Particle Swarm Optimization** [203]. We utilize the implementation from the `NiaPy` package [54] with the following parameters: $C_1 = 2$ (cognitive component) , $C_2 = 2$ (social component), $w = 0.7$ (inertial weight), $v_{min} = -1.5$ (minimal velocity), $v_{max} = 1.5$ (maximal velocity) and population of 30.

- **Cuckoo Search** [204]. We utilize the implementation from the `NiaPy` package [54] with the following parameters: $p_a = 0.2$ (proportion of worst nests) and $\alpha = 0.5$ (scale factor for the Levy flight) and population size of 30.

- **Simulated Annealing** [205]. We utilize the implementation from the `NiaPy` package [54] with the following parameters: $\delta = 0.5$ (movement of neighbor search, $T = 2000$ (starting temperature), $\Delta_T = 0.8$ (change in temperature), $\epsilon = 1e - 23$ (error value) and a linear cooling method.

- **Differential evolution** [206]. We utilize the implementation from the `NiaPy` package [54] with the following parameters: $F = 1$ (scale factor), CR= 0.8 (crossover probability), random cross mutation and population size of 30.

- **Nelder-Mead** [56]. We utilize the implementation from the `NiaPy` package [54] with the following parameters: $\alpha = 0.1$ (reflection coefficient), $\gamma = 0.3$ (expansion coefficient), $\rho = -0.2$ (contraction coefficient) and $\sigma = -0.2$ (shrink coefficient).

- **Covariance Matrix Adaptation Evolution Strategy** (CMA-ES) [55]. We utilize the implementation from the package `pycma` [207]. We utilize the following parameters $\sigma_0 = 0.5$ (initial standard deviation).

- **RandomSearch1** (Random Search x1) Random samples are selected to search the space. Each sample is evaluated only once. We utilized our own implementation.

- **RandomSearch2** (Random Search x2) is a variation of Random Search in which the same point in the search space is evaluated twice before getting a new sample. We utilized our own implementation.

9.4.2 The benchmark functions

For the benchmark functions, we randomly selected 30 benchmark functions from a pool of 220 benchmark functions from both the BBOB-2009 [208] function definitions as well as from a literature survey [183]. All the benchmark functions are used for the optimization of continuous parameters. To ensure that they were correctly specified, they were tested nightly, for over a month, with a random search algorithm to verify whether all the global minima were identified and if these were indeed the global minima. Additionally, due to the computational time required, we restricted the benchmark functions to a limit of 6 dimensions, which is sufficient to illustrate the generality of the statistical models.

For a mathematical definition of these functions, we refer to the survey [183] and the BBOB-2009 function definitions [208]. The used benchmark functions are: Sphere 6-D, Tripod, ChungReynolds 2-D, Pinter 6D, StrechedVSineWave 2-D, Trigonometric-1 6-D, BentCigar 6-D, ChenV, Discus 2-D, Schwefel2d20 2-D, ChenBird, Schwefel2d21 6-D, Zakharov 2-D, Damavandi, Schwefel2d4 6-D, Whitley 6-D, Shubert, XinSheYang2 2-D, Mishra7 6-D, Schwefel2d23 6-D, Exponential 2-D, Salomon 2-D, RosenbrockRotated 6-D, Qing 2-D, Lunacek-BiRastrigin 6-D, ThreeHumpCamelBack, Schwefel2d26 6-D, Trefethen, Price1 and Giunta.

9.4.3 The experimental conditions

All algorithms were run for all benchmark function ten times in each combination of an experimental condition:

- **Noise**: 0 or 3.0. When noise is present, we added a Gaussian random variable on the output of the benchmark function with a mean in the benchmark value and a standard deviation of 3.0. Although this might impact differently each benchmark function result, it represents a constant measurement error in real-world conditions.

- **Budget**: 20, 10^2, 10^3, 10^4 and 10^5 function evaluations per number of dimensions of the benchmark function.

These conditions resulted in a total of $24,000$ data points in our data set, in which each point corresponds to one algorithm run.

9.4.4 The logged metrics

Apart from the benchmark functions metrics (e.g. number of dimensions) and the experimental conditions (noise level, budget), we logged additional metrics for each algorithm. For these metrics we use the following notation:

$\mathbf{X} \in \chi$ from a compact subset space $\chi \subset \mathbb{R}^d$ is a vector of the input for the benchmark function and has dimension d .

$\mathbf{X}^*$ is the global minimum of the benchmark function $f_{\min} = f(\mathbf{X}^*)$. We consider that a benchmark function can have more than one global minima. The relationship between these variables is represented by: $\mathbf{X}^* = \arg\min_{\mathbf{X}\in\chi} f(\mathbf{X})$.

$\mathbf{X}_{\text{opt}}$ is the output value/best solution of the optimization algorithm at the end of the budget.

$\mathbf{X_i}$ is a sampled point of the search space selected by the algorithm at function evaluation i.

The logged metrics are represented below

- **Final reward difference**: $\Delta f_{\text{reward}} = f(\mathbf{X}_{\text{opt}}) - f(\mathbf{X}^*)$,
- **Euclidean distance**: $D = \|\mathbf{X}* - \mathbf{X}_{\text{opt}}\|_2$
- **Solved at precision** $\epsilon \in [1, 0.1, 1e-3, 1e-6]$: this is a boolean variable that indicates if the problem was solved or not at the end of the budget: $\Delta f_{\text{reward}} < \epsilon$
- **Solved at FEval**: the function evaluation (FEval) in which the algorithm solved the problem with a specific precision $\Delta f_{\text{reward}} < \epsilon$
- **CPU time**: this computes the time spent by each algorithm in each problem for the whole budget.

The presented models often make use of a transformation of these variables, in such cases, we describe the specific transformation in each model.

9.4.5 The research questions

The presented statistical models in Section V address the following research questions.

- **RQ1-a:** What is the probability of each algorithm solving a problem at precision $\epsilon \leq 0.1$?
- **RQ1-b:** What is the impact of noise in the probability of success of each algorithm at precision $\epsilon \leq 0.1$?
- **RQ2:** What is the expected improvement of these algorithms against the Random Search in noiseless benchmark functions in terms of approaching a global minimum based on the Euclidean distance to the location of the closest global minimum?
- **RQ3:** How can optimization algorithms be ranked in the conditions of 10,000 evaluations per dimension budget in noisy benchmarks?
- **RQ4-a:** What is the average number of function evaluations (FEval) taken by an algorithm to converge to a solution at a precision of $\epsilon \leq 0.1$ and with a maximum budget of 100,000 function evaluations per dimension?
- **RQ4-b:** What is the impact of noise in the number of function evaluations (FEval) taken by an algorithm to converge to a solution at a precision of $\epsilon \leq 0.1$ and with a maximum budget of 100,000 function evaluations per dimension?
- **RQ5:** Is there a difference in the CPU time taken per function evaluation between the PSO, the Random Search x1, and the Differential Evolution algorithms?

9.5 Statistical Models

In this section, we provide an overview of five statistical models for answering different practical research questions in benchmark data. For each model, we present an introduction to the model, the model, and an analysis of the results focusing on the interpretation aspect of the intervals with plots. We conclude the discussion of each model with some final remarks, indicating possible extensions or practical issues that one may find.

It is worth reinforcing that these models are not unique and several variations can be made. Our choice for these models was based on the simplicity and ability to answer many practical questions. More complex models can be made and derived from these models. In BDA, starting with simple models and extending them is encouraged, and reporting these models can provide a greater level of transparency for research and replication studies.

Before we present the models, we provide a short overview of hierarchical/multilevel models. We emphasize the need of using hierarchical models for benchmark comparison since they can compensate for the clustering effect of the benchmarks.

In the online appendix, we provide the empirical data used in this text, a step-by-step code used in all the data transformation, cleaning, plots, and tables. Additionally, we provide the Stan code for all the models together with the exact data used to fit these models and the analysis of the convergence and validity of the models.

Notation convention:

- For notation clarity, we omit the indexing variable that indicates each observation of the dataset, for example, instead of $y[i] \sim \text{Normal}(a + b \cdot x[i], \sigma)$ we represent as $y \sim \text{Normal}(a + b \cdot x, \sigma)$. Similar notations are widely used in other Bayesian data analysis texts [95, 196].

- All dependent variables are indicated as y.

- All predictors (independent) variables are indicated with x with optional subscripts i to indicate the algorithm.

- All intercepts (the independent terms of the linear regression without any predictor), including the random effects of the models, are indicated by a with optional subscripts i to indicate the algorithm and j for the benchmark.

- All slopes of the models are indicated with b with optional subscripts i to indicate the algorithm.

- The subscript index i indicates that there is one parameter for each algorithm. For example, a_i indicates that there is one intercept for each algorithm, a_1 for the first algorithm.

- The subscript index j indicates the parameter of the benchmark function. For example $a_{\text{bm},j}$ indicates that there is one intercept for each benchmark function.

- If there is no subscript index, the parameter is common for all algorithms or benchmark functions.

9.5.1 Compensating the effects of benchmarks

When the measured units are drawn from the same cluster within a population (e.g. multiple samples from the same benchmark function), these can no longer be considered independent samples. This situation can add bias from unobserved variables into the model and shift the posterior distributions [209]. A strategy to overcome such problems is called multilevel modeling or hierarchical modeling, and it is not restricted to the Bayesian framework. Snijders and Bosker [209] present a full treatment of multilevel modeling in the frequentist setting. One approach to compensate for the clustering problem is to add a blocking variable that estimates the effect of each cluster. With a large number of clusters (e.g., 30 benchmark functions), we can model the effect of the clusters utilizing a random-effects variable. This random effect variable indicates that every benchmark will be drawn from a probability distribution (and therefore we can only estimate the parameters of this distribution), reducing model complexity and allowing us to evaluate the impact of the benchmark functions overall. Of course, it is also possible to observe the effect of each function in this framework.

In the Bayesian framework, we can condition the priors of the random effects variables over new random variables called hyperpriors. For example, let's consider the example of a simple linear regression in which each of the intercepts depends on the algorithm a_i and the slope b is constant for all algorithms. Variables a and b are not modeled as random effects and therefore are modeled only with their priors.

If we consider that each observation comes from a finite number of clusters (benchmarks), in which the cluster is represented by the index j, we can create the Model 9.1 to include a random variable intercept that represents the effect of each cluster on the observed variable. The random variable intercept for the benchmarks is represented by $a_{\mathrm{bm},j}$. The exponential distribution is a common choice for modeling variance in random effects [197], where lower rate parameters create proper but weakly-informative hyper prior. However, other common choices are the half-normal and the Cauchy distribution.

Model 9.1: Bayesian linear regression considering the effect of the benchmarks

$$
\begin{aligned}
y &\sim \mathrm{Normal}(a_i + a_{\mathrm{bm},j} + b \cdot x, \sigma), && \\
a_i &\sim \mathrm{Normal}(0, 10) && \text{[Prior]}, \\
b &\sim \mathrm{Normal}(0, 10) && \text{[Prior]}, \\
\sigma &\sim \mathrm{Exponential}(1) && \text{[Prior]}, \\
a_{\mathrm{bm},j} &\sim \mathrm{Normal}(0, s) && \text{[Prior]}, \\
s &\sim \mathrm{Exponential}(1) && \text{[Hyperprior]}.
\end{aligned}
$$

In terms of interpretation, although we analyze the impact of each benchmark function (since we estimate the intercept of each benchmark function), we are more concerned with the interpretation of the standard deviation of the random effects (the s parameter). This parameter indicates how much variance we can attribute to the clustering information of the benchmark functions. Additionally, suppose the selection of benchmark functions is representative of the set of problems that the algorithms are going to solve. In that case, we can interpret how much variance we can expect in the model due to a change of problem.

The multilevel approach can be easily extended for additional levels in the hierarchy (if the function can be classified as easy or hard to solve, or other properties, such as separability or modality) and include different hierarchies in parallel. For more information regarding these extensions and other applications of multilevel models, we refer to [95, 196]. Note that the separation between the main effects of the cluster effects introduces n parameters in the model, in which n is the number of clusters.

9.5.2 Probability of success

In this subsection, we utilize a multilevel generalized linear model with a logit link function to model the binomial response and answer the research questions RQ1-a and RQ1-b.

RQ1-a: What is the probability of each algorithm solving a problem at precision $\epsilon \leq 0.1$?

RQ1-b: What is the impact of noise in the probability of success of each algorithm at precision $\epsilon \leq 0.1$?

9.5.2.1 The model

One model that can be used for addressing these research questions is the generalized linear model with the binomial distribution and the inverse logit. The binomial is a common choice of the likelihood distribution in generalized linear models when one wants to estimate how many out N tries are successful given a probability p [95, 196, 210]. Through the use of generalized linear models, we can include random effects terms and predictors. This requires a link function to transform the continuous linear equation to the input of the binomial distribution. Common choices for link functions are the logit or the probit functions. These link functions allow us to map the continuous output of the linear regression to discrete values used in the binomial distribution. The binomial model is represented by Model 9.2.

Model 9.2: Binomial model

$$
\begin{aligned}
y &\sim \text{Binomial}(N, p),\\
p &= \text{logit}^{-1}(a_{\text{alg},i} + a_{\text{bm},j} + \text{b\_noise}_i \cdot x_{\text{noise}}),\\
a_{\text{alg},i} &\sim \text{Normal}(0, 5),\\
\text{b\_noise}_i &\sim \text{Normal}(0, 5),\\
a_{\text{bm},j} &\sim \text{Normal}(0, s),\\
s &\sim \text{Exponential}(0.1).
\end{aligned}
$$

Model 9.2 uses the following notation. Let's consider the example of one row in the dataset that indicates that the algorithm PSO was tried ten times with noise=3.0, budget=1000 for one benchmark function. On those ten tries, it solved the problem at $\epsilon = 0.1$ two times. The inverse logit function, as defined below, maps the values of x from $(-\infty, +\infty)$ to the interval $(0, 1)$, so its output can be used as the probability parameter of the binomial distribution

- $\text{logit}^{-1}(x) = \frac{1}{1+\exp(-x)}$.
- N: is an integer, parameter of the binomial distribution, that represents the total number of tries (in our case the aggregated value of repeated measures of a single algorithm). In the example $N = 10$.
- y: is an integer that indicates from the N tries, how many of those were successful (or had a result of 1). In example, $y = 2$.
- $a_{alg,i}$: represents the mean (intercept) effect of each algorithm.
- b_noise_i: is the influence of noise in each algorithm.
- x_{noise}: indicates the noise used. In the example, $x_{\text{noise}} = 3.0$.
- $a_{bm,j}$: indicates the random effect of the benchmarks.
- p: is a variable modeled by the linear equation and it indicates the probability of success. We can use this probability to assess how the different parameters impact the probability of success.

The model above captures different coefficients for the influence of noise and budget on each algorithm. If desired (e.g. to measure the impact of factor regardless of the algorithm), these parameters could be aggregated in a single parameter.

9.5.2.2 Model interpretation

After running this model in Stan (chains=4, warmup=200, iterations=3000), we obtain the posterior distribution of the model parameters. Table 9.1 shows the mean and HPD intervals of the model parameters as well as the mean odds ratio (OR) and the OR HPD interval. The OR measure indicates the relative probability of success compared to the probability of failure. If the odds ratio is greater than 1, the parameter increases the probability of success. If the odds ratio is between 0 and 1, it decreases the probability of success. It is worth noting, however, that if an algorithm has a high odds ratio (or parameter value) the influence of the benchmark might be small (due to the asymptotic characteristic of the inverse logit function)

Table 9.1 indicates that the algorithms PSO, Differential Evolution, and CMAES have a significantly higher mean of the OR compared to the other algorithms, and are the only ones that, on average, have a higher probability of solving a problem than not solving. However, all three also have a wide OR HPD interval, meaning that their performance is greatly affected by external random factors (such as choice of seed for example). The large overlap between the OR HPD interval of those three algorithms indicates no statistical difference between them.

Table 9.1: Estimated parameters of the model. OR indicates the odds ratio of the respective parameter

Parameter	Mean	HPD low	HPD high	OR Mean	OR HPD low	OR HPD high
a_CMAES	-0.10	-0.99	0.78	0.90	0.37	2.18
a_Cuckoo	-2.55	-3.40	-1.63	0.08	0.03	0.20
a_DiffEvol.	-0.21	-1.10	0.66	0.81	0.33	1.94
a_NelderM.	-4.35	-5.24	-3.42	0.01	0.01	0.03
a_PSO	-0.39	-1.26	0.49	0.67	0.28	1.63
a_RandomS1	-2.01	-2.84	-1.07	0.13	0.06	0.34
a_RandomS2	-2.22	-3.09	-1.33	0.11	0.05	0.27
a_SimAnneal	-2.56	-3.47	-1.69	0.08	0.03	0.18
b_CMAES	-1.27	-1.37	-1.18	0.28	0.26	0.31
b_Cuckoo	-0.81	-0.93	-0.70	0.44	0.40	0.50
b_DiffEvol.	-1.35	-1.46	-1.26	0.26	0.23	0.28
b_NelderM.	-0.39	-0.52	-0.25	0.68	0.59	0.78
b_PSO	-1.15	-1.24	-1.06	0.32	0.29	0.35
b_RandomS1	-0.97	-1.08	-0.86	0.38	0.34	0.42
b_RandomS2	-0.72	-0.83	-0.62	0.49	0.44	0.54
b_SimAnneal	-0.79	-0.91	-0.68	0.45	0.40	0.51
s	2.44	1.78	3.16	11.47	5.94	23.68

From the analysis of the odds ratio of the noise coefficient, we can see that all algorithms perform more poorly in the presence of noise (all the odds ratios are lower than one). Noise has a greater relative impact on the Differential Evolution, CMAES, and PSO algorithms. However, this relative effect should be analyzed in the context of the total probability of success. Since all other algorithms have a much lower probability of solving a problem, the effect of noise on them is smaller as they would probably not solve the problem even without noise. Since the logit function is not linear, it is recommended to evaluate both the marginal and absolute impact of the parameters in the model predictive accuracy.

In terms of the effect of the benchmark functions, the effect of the benchmark (by our model) is drawn from a normal distribution with a mean of 0 and a standard deviation that has the posterior distribution of s. Since the effect of the benchmarks are sampled from this distribution, it can have an impact with the same magnitude or even higher than the choice of algorithms in the probability of solving a problem. This reinforces that the choice of benchmark functions greatly impacts the results of algorithms and that using higher hierarchy levels for the benchmarks may improve the estimates of the algorithms.

9.5.2.3 Remarks

The same model can be used to answer many other research questions, such as, the probability of an algorithm solving problems when duplicating the budget (such question could be used to determine an appropriated budget for some algorithms, especially in expensive functions), the probability to solve a particular problem (now the focus is on specific benchmark functions), among others. An equivalent variant of this model, one that does not use aggregated data, utilizes a Bernoulli distribution as the likelihood instead of the binomial, the inverse logit part and the priors can remain the same.

Additional discussion about categorical models (such as the Binomial and Bernoulli model) in the context of generalized linear models and possible extensions can be found in both Bayesian and frequentist statistical textbooks [196, 210].

9.5.3 Algorithm relative improvement over Random Search

In this subsection, we address RQ2. For this question, we will use a model based on a linear regression. This model is analogous to the frequentist linear regression models but we have estimations of the full posterior distribution, control over the priors, and credible intervals (instead of confidence intervals) for inference. Linear models such as this one facilitate the direct comparison of the magnitude and direction of the parameters. Since we are assessing a linear model directly, interpretation mistakes over the absolute impact of transformations such as the odds-ratio do not occur.

RQ2: What is the expected improvement of these algorithms against the Random Search in noiseless benchmark functions in terms of approaching a global minimum based on the Euclidean distance to the location of the closest global minimum?

9.5.3.1 The model

Model 9.3 uses a regression model with normal distribution for the likelihood and model the standard deviation (of the error term) with a single parameter σ.

Model 9.3: Regression model

$$
\begin{aligned}
y &\sim \text{Normal}(\mu, \sigma), \\
\mu &= a_{\text{alg},i} + a_{\text{bm},j}, \\
\sigma &\sim \text{Exponential}(1), \\
a_{\text{alg},i} &\sim \text{Normal}(0, 1), \\
a_{\text{bm},j} &\sim \text{Normal}(0, s), \\
s &\sim \text{Exponential}(0.1).
\end{aligned}
$$

In Model 9.3, we have the following notation:

- y: is the metric that indicates the relation between the Euclidean distance of the algorithm and the random search. There are multiple ways to measure improvement but, in this example, we will use the difference between the Random Search x1 Euclidean distance and the algorithm, divided by the Random Search x1 Euclidean distance. The results of this ratio are caped between -1 and 1. The Random Search x1 Euclidean distance is averaged in the repeated measures.

- $a_{alg,i}$: represents the mean (intercept) effect of each algorithm.

- $a_{bm,j}$: indicates the random effect of the benchmarks.

- μ: represents the mean value of the likelihood, modeled by the linear equation. This is a transformation parameter and not an estimated parameter of the model. μ represents the average improvement to random search after counting for the effect of the algorithm and the benchmark but it does not include the variance added by the σ parameter.

- s: represents the standard deviation of the effect of the benchmark functions.

9.5.3.2 Model interpretation

After running this model in Stan (chains=4, warmup=200, iterations=2000), we get the posterior of each parameter and compute the HPD intervals for the intercept of each algorithm. The appendix provides a table with the estimates of every posterior parameter obtained by the model. Table 9.2 shows the obtained posterior parameters and their HPD intervals. The mean value on this table corresponds to the estimated parameters and not the mean value of the likelihood (μ).

Table 9.2: Estimated parameters of the relative improvement model and their respective HPD intervals

Parameter	Mean	HPD low	HPD high
σ	0.64	0.63	0.65
a_CMAES	0.15	0.09	0.21
a_CuckooSearch	-0.38	-0.44	-0.32
a_DifferentialEvolution	0.30	0.24	0.36
a_NelderMead	-0.64	-0.70	-0.58
a_PSO	0.32	0.26	0.38
a_SimulatedAnnealing	-0.57	-0.63	-0.51
s	0.15	0.11	0.19

The algorithm intercept represents the average improvement over random search in an average benchmark with a null effect. The random effect standard deviation s, has a similar interpretation as before. The impact of the benchmark over the relative improvement given by the benchmarks is sampled from a normal distribution with a mean equal to zero and standard deviation s. The parameter σ represents the dispersion of the results around the average μ.

From Table 9.2, we can see that only the algorithms PSO, Differential Evolution ,and CMAES have a positive relative improvement over the Random Search x1 algorithm. All the other parameters had negative estimates, which result that they perform on average worse than random search in approaching the global minima. We can notice that the standard deviation of the measurements (σ) is high compared to the estimates of the algorithms. This indicates that there is a lot of non-explained variance in the data. For this case, since the comparison is relative between algorithms, we see that the benchmark functions have a smaller impact on the estimates, with a much lower standard deviation for the random effects s.

9.5.3.3 Remarks

This model can easily be extended to include other predictors as any regression model, such as to evaluate the impact of noise and the log of budget, among others.

Often, the first regression model we try is based on the normal likelihood. However, if we see long-tail distributions or outliers, a robust regression might be more appropriate. Robust methods are discussed in our last model in this section. The choice between the different types of regression is often subject to some experimentation to see which model works and predicts the data better. The differences between the performance of all of these models can be compared with information criteria methods such as WAIC.

It is worth noting that the MCMC is quite sensitive to numerical stability. Inferences in which data goes from wide ranges such as 0.001 to 1000 usually makes the MCMC fail to converge. This problem is particularly common in benchmark data since functions have very different ranges. One solution to this problem is to normalize the input data in respect to another baseline algorithm (which we did). Another solution commonly used approach is to compare ranking statistics. Ranking statistics have the disadvantage of throwing out part of the data. Ranking allows researchers to assess which algorithm performs better but not by how much.

9.5.4 Ranking comparison

In this subsection, we utilize a Bayesian variant of the Bradley-Terry Model [211, 212] to answer RQ3.

RQ3: How can optimization algorithms be ranked in the conditions of 10,000 evaluations per dimension budget in noisy benchmarks?

9.5.4.1 The model

The Bradley-Terry model [211] is a popular approach to investigate paired comparisons [213]. In this model, a pair of competitors are compared and one of them is classified as the winner. This model has been used in various applications, from medicine to machine-learning applications on search engines.

The model is based on the idea of a comparison contest between players (or in our case, algorithms) without the possibility of ties. The model can be expressed through latent variables α_i that represent the "strength" or the ability of each algorithm [214]. The odds that algorithm i beats j is expressed by α_i/α_j, and the probability that i beats j is expressed by:

$$\begin{aligned} \Pr[i \text{ beats } j] &= \frac{\exp \alpha_i}{\exp \alpha_i + \exp \alpha_j}, && (9.2) \\ &= \text{logit}^{-1}(\alpha_i - \alpha_j). && (9.3) \end{aligned}$$

Since we do not observe the probabilities, we will model it through a Bernoulli distribution. The goal of this model is to estimate the strength of the parameters and the order of these parameters will give a rank of the algorithms. The main advantage of this model over the frequentist approach with the maximum likelihood estimator is the ability to restrict the priors and obtain a full posterior estimation of the parameters.

If we want to include other predictors as well as control for random effects in the latent strength parameter, we can use the following linear transformation for each latent strength variable [215].

$$\alpha_i = \sum_{k=1}^{p}(\beta_k x_k) + a_{\mathrm{bm},i,j}, \tag{9.4}$$

in which p is the number of predictors we want to add, β_k are the coefficients we want to estimate, x_k are the independent variables for each predictor, and $a_{\mathrm{bm},i,j}$ is the estimated effect of the random variable, such as the clustering effect of the benchmark j in the latent variable α_i. Here we note that each benchmark will provide a different effect for each strength that is estimated in the random effects variable. It is worth noting that in each paired comparison, if the same covariates are available for both contestants, they will cancel the effect on each other out.

Model 9.4 represents the Bayesian Bradley-Terry model with random effects for the benchmarks:

Model 9.4: Bayesian Bradley-Terry Model

$$\begin{aligned}
&y \sim \mathrm{Bernoulli}(p),\\
&p = \mathrm{logit}^{-1}(a_{\mathrm{algo1}} + a_{\mathrm{bm,algo1},j} - a_{\mathrm{algo0}} - a_{\mathrm{bm,algo0},j}),\\
&a_i \sim \mathrm{Normal}(0, 2),\\
&a_{\mathrm{bm},i,j} \sim \mathrm{Normal}(0, s),\\
&s \sim \mathrm{Exponential}(0.1).
\end{aligned}$$

In Model 9.4, we have the following notation:

- y: indicates which of the two algorithms (algo1 or algo0) won the contest. It can have only two values 0 for algo0 or 1 for algo1.
- a_{algo1} and a_{algo0}: represents a paired comparison between two algorithms
- a_i: indicates the latent strength variable of each algorithm.
- $a_{\mathrm{bm,algo0},j}$ and $a_{\mathrm{bm,algo1},j}$ are the random effects due to the benchmark j in the algorithm 0 or 1.
- s: represents the standard deviation of the effect of the benchmark functions.

9.5.4.2 Model interpretation

After running this model in Stan (chains=4, warmup=200, iterations=4000), we get the posterior of each parameter and compute the HPD intervals for the intercept of each algorithm strength. Table 9.3 shows the summary statistics of

Table 9.3: Estimated parameters of the ranking model and the respective HPD intervals

Parameter	Mean	HPD low	HPD high
a_CMAES	1.04	-0.48	2.59
a_CuckooSearch	-0.41	-1.87	1.19
a_DifferentialEvolution	1.99	0.43	3.51
a_NelderMead	-2.98	-4.51	-1.43
a_PSO	1.58	0.08	3.13
a_RandomSearch1	0.28	-1.28	1.78
a_RandomSearch2	0.25	-1.25	1.81
a_SimulatedAnnealing	-1.69	-3.20	-0.11
s	1.82	1.59	2.05

the posterior distribution parameters of the model, the latent strength variables, and the standard deviation of the random effects of the benchmarks.

These strength parameters can be used to either assess the probability of one algorithm beating the other or to rank the algorithms. However, despite the apparent large overlap between these latent parameters, it does not indicate that the algorithms perform similarly. To compare a specific algorithm with the other, it is possible to either compute the posterior distribution of one algorithm beating the other (as in equation 9.2) or to calculate the posterior distribution of the ranks.

To calculate the posterior distribution of the ranks, we use 1000 samples from the posterior of the strength parameters (taking into account the effect of the benchmarks) and rank them. With this procedure, we have a distribution of the ranks, which gives us also information regarding to the uncertainty of the ranking process. These results are shown in Table 9.4, which shows the median rank and the rank variance for each algorithm.

Table 9.4: Ranking the algorithms based on the reward difference taking accounting for the effect of the benchmarks

Algorithm	Median Rank	Variance of the Rank
DifferentialEvolution	1	0.20
PSO	2	0.30
CMAES	3	0.34
RandomSearch1	4	0.47
RandomSearch2	5	0.51
CuckooSearch	6	0.21
SimulatedAnnealing	7	0.01
NelderMead	8	0.00

9.5.4.3 Remarks

The Bradley-Terry model is the simplest model for analyzing ranks. Note that this model does not take into account the differences between the algorithms (despite how big or small they might be). We did not include any predictor in our model since the predictor variable is always the same for each algorithm comparison (since this is how we designed the experimental data collection) and they would cancel each other.

There are extensions and alternative models to the Bradley-Terry model, such as the Thurstonian model (that uses a probit instead of the logit function). The Plackett-Luce model described in [192] is equivalent to the presented Bradley-Terry model when a complete rank is converted to paired comparisons with independence between the algorithms in the rank [216]. The advantage of converting ranks into paired comparisons and using the Bradley-Terry model is the possibility to use partial ranks.

Our model does not accept ties between contestants in its formulation. Since we are comparing the true reward difference, ties only occur when the algorithms achieve the global minima, which is a rare event. One approach to solve ties is to randomly assign a winner [215]. However, in cases where ties are common and estimating the probability of two contestant algorithms to tie is relevant, such as in combinatorial optimization, extensions to the Bradley-Terry model can be used. The most common extension to accommodate ties is the Davidson generalization of the Bradley-Terry model [212, 217]. This extension adds an additional parameter $\nu > 0$ that estimates the overall maximum probability of a tie and the dependence of the probability of a tie in the strength parameter. If $\nu \to \infty$ the probability of a draw is 1. If $\nu \to 0$ the probability of tie depends only on the strength of the algorithms. In the online appendix we provide the Stan statistical model for the Davidson extension. The model adds a conditional statement in the probabilities as seen below:

$$\begin{aligned}
\Pr[i \text{ beats } j | \text{not tie}] &= \frac{\exp \alpha_i}{\exp \alpha_i + \exp \alpha_j + \exp\left(\nu + (\alpha_i + \alpha_j)/2\right)}, \\
\Pr[i \text{ ties } j] &= \frac{\exp\left(\nu + (\alpha_i + \alpha_j)/2\right)}{\exp \alpha_i + \exp \alpha_j + \exp\left(\nu + (\alpha_i + \alpha_j)/2\right)}.
\end{aligned}$$

9.5.5 Number of function evaluations to converge to a solution

In this subsection, we will consider two research questions that address the number of function evaluations (FEval) to the occurrence of an event, RQ4-a and RQ4-b. Such questions are usually discussed in the area of survival analysis, in which the primer interest is when an event occurs [218].

RQ4-a: What is the average number of function evaluations (FEval) taken by an algorithm to converge to a solution at a precision of $\epsilon \leq 0.1$ and with a maximum budget of 100,000 FEval per dimension?

RQ4-b: What is the impact of noise in the number of function evaluations (FEval) taken by an algorithm to converge to a solution at a precision of $\epsilon \leq 0.1$ and with a maximum budget of 100,000 FEval per dimension?

As we noticed with the previous models, some algorithms have a very low success rate. For this example, we will use only the top 4 algorithms indicated by the previous Bradley-Terry Model: Differential Evolution, PSO, CMA-ES, and RandomSearch1. The proposed research questions address the average FEval to converge to a solution divided by the number of dimensions because we are utilizing a fixed budget per dimension as the experimental condition. This leads benchmark functions with 6 dimensions to be evaluated up to 3 times higher than benchmark functions with only 2 dimensions. Without correcting the number of FEval by the dimensions, the number of dimensions in a benchmark function would create a higher difference than the choice of algorithms. This transformation ensures that the results of different benchmarks are comparable regardless of the number of dimensions.

9.5.5.1 The model

One important aspect of survival models, that makes them different from other analyses, is the presence of censored data. Censored refers to the characteristic that an event might not occur in the window of observation. If we do not consider censoring, we will be eventually creating a downwards bias in our inference [196]. Although there are many ways that data can be censored, in the analysis of benchmark functions, we are concerned primarily with uninformative and *right censoring*. Uninformative, because the censored data does not contain information regarding the survival (e.g. the data is censored because the algorithm has a bug and never converges to a solution) and right censor because we do not observe the event due to the end of the window of observation [218]. In our example, right censoring is when an algorithm is unable to solve the problem given the budget.

Survival analysis is usually modeled in terms of two related probability functions, the survival and the hazard functions. The survival function $S(t)$ models that an algorithm will not converge until function evaluation t. The hazard function $h(t)$ models the probability that an algorithm will converge at function evaluation t during that particular evaluation, i.e., it is the instantaneous event rate of an algorithm converging to a solution if it hasn't yet. More commonly, we use the cumulative hazard function $H(t)$, obtained by integrating $h(t)$ over the number of FEval. In summary, the hazard function models the occurrence of the event (converge to a solution) and the survival function models the non-occurrence of the event. The relationship between $S(t)$ and $H(t)$ is given by: $H(t) = -\log S(t)$.

We will use the most common (and simple) survival model, the Cox's Proportional Hazard model and the Bayesian formulation proposed by Kelter [219] for the random effects. This model assumes a time-invariant exponential hazard function and is easily extended with additional predictors. In this model, the hazard function is constant in time and we refer to it as $\lambda(\mathbf{X})$, in which $\mathbf{X}$ is the matrix of covariates, a is a constant baseline hazard (if all covariates are zero) and $\mathbf{b}$ is the corresponding matrix of the coefficient of the covariates. The expected value for the occurrence of the event is the inverse of the hazard function and defined as:

$$h(t) = \lambda(\mathbf{X}) = \exp(a + \mathbf{b}\mathbf{X}), \tag{9.5a}$$

$$\mu(\mathbf{X}) = \frac{1}{\lambda(\mathbf{X})}. \tag{9.5b}$$

In the Bayesian framework (instead of using the partial likelihood method), we divide the model into censored and non-censored parts. While the non-censored data uses the hazard function above, the censored data uses the complementary cumulative probability distribution function. Our model is represented by Model 9.5:

In Model 9.5, we have the following notation:

Model 9.5: Bayesian Cox's Proportional Hazard

$$
\begin{aligned}
&\text{If event} = 1: \\
&y \sim \text{Exponential}(\lambda_{i,j}) \\
&\text{If event} = 0: \\
&y \sim \text{Exponential-CCDF}(\lambda_{i,j}) \\
&\lambda_{i,j} = \exp\left(a_{\text{alg},i} + a_{\text{bm},j} + b_{\text{noise},i} \cdot x_{\text{noise}}\right), \\
&\mu_{i,j} = \frac{1}{\lambda_{i,j}}, \\
&\text{Priors}: \\
&b_{\text{noise},i} \sim \text{Normal}(0, 2), \\
&a_{\text{alg},i} \sim \text{Normal}(0, 10), \\
&a_{\text{bm},j} \sim \text{Normal}(0, s), \\
&s \sim \text{Exponential}(0.1).
\end{aligned}
$$

- event: is a binary variable that indicates if the algorithm has found a solution or not.
- y: if event=1, y represents the function evaluation that the algorithm finds a solution divided by the number of dimensions. If event=0, y is considered a missing value.
- $a_{\text{alg},i}$ represents the baseline effect of each algorithm.
- $b_{\text{noise},i}$: is the influence of noise in the FEval to find a solution of the algorithm.
- x_{noise}: indicates the noise of the benchmark function.
- $a_{\text{bm},j}$: indicates the baseline of the random effects of the benchmarks.

9.5.5.2 Model interpretation

After running this model in Stan (chains=4, warmup=200, iterations=3000), we get the posterior of each parameter and compute the HPD intervals for each parameter. Table 9.5 shows the obtained posterior parameters and the HPD intervals. Combined with equations 5a and 5b, we can infer that the algorithms that have a lower baseline effect have a higher probability of surviving, i.e., not finding a solution in the specified budget. The average FEval to converge in the noiseless case can be seen in table 9.6. The presence of a random noise also increases the survival probability, as expected, and the lower the value of the noise variable, the more it impacts the ability of the algorithm to find a solution.

The average number of FEval taken by an algorithm to converge (RQ4-a) is based on the expected value of the exponential distribution which is given by the $\mu_{i,j}$ variable. By sampling 1000 values of the posterior distribution of

Table 9.5: Estimated parameters of the time to converge model and the respective HPD intervals

Parameter	Mean	HPD low	HPD high
a_CMAES	-5.09	-6.03	-4.16
a_DifferentialEvolution	-6.37	-7.29	-5.42
a_PSO	-6.30	-7.22	-5.36
a_RandomSearch2	-8.95	-9.91	-8.02
b_CMAES	-0.79	-0.93	-0.66
b_DifferentialEvolution	-0.96	-1.09	-0.83
b_PSO	-0.68	-0.80	-0.56
b_RandomSearch2	-0.41	-0.55	-0.27
s	2.44	1.85	3.12

the Cox's hazard model, we can compute the average function evaluation to converge for any experiment condition or benchmark. Table 9.6 shows the average FEval to converge and the HPD intervals for the noiseless condition and the average of the benchmark functions (the condition where $a_{\text{bm},j} = 0$). We can see that in these conditions CMAES has a lower average number of function evaluations to converge compared to the others, while both PSO and Differential Evolution have approximately the same interval range. Note that these intervals are relatively wide due to the diversity of the benchmark functions and the uncertainty added by right censoring (when an algorithm is not able to find a solution in the determined budget). To investigate the average for each benchmark or for the scenario with noise, we can substitute the equivalent estimated parameters and predictors in the equations of the model 9.5 to obtain the $\mu_{i,j}$. A computational example of this procedure is shown in the appendix.

To facilitate the interpretation of the Cox's regression model, we can also analyze the hazard ratio (HR) quantities and the baseline hazard $h_0 = \exp(a_{\text{alg},i})$. The HR represents the contribution of a parameter in the probability of occurring an event (solving the benchmark problem). The HR is defined as $\text{HR}(b) = \exp(b)$, if HR is greater than one the parameter increases the chance of the occurrence of the event, if it is less than 1, it reduces the chance of the occurrence of the event. Table 9.6 shows the HR of the mean value of the parameters. We can see that all algorithms have a low baseline for the hazard and that noise reduces this hazard even further, therefore, increasing the number of function evaluations to converge to a solution. We can notice as well that noise impacts the hazard of the Differential Evolution algorithm more and the RandomSearch1 much less.

Table 9.6: Average FEval to converge and the Hazard Ratio for the FEval to converge model

	Avg FEval			Hazard Ratio	
Parameter	Mean	HPD low	HPD high	Baseline	Noise
CMAES	179	42	349	0.006	0.456
DifferentialEvolution	663	177	1292	0.002	0.386
PSO	601	142	1159	0.002	0.509
RandomSearch1	6663	1746	13252	0.000	0.561

9.5.5.3 Remarks

In the model of survival data with the Cox regression, it is important to add the random effects of the benchmark functions. Since the algorithms often cannot solve a problem regardless of the budget, if we do not include the effects of the benchmarks, we underestimate the hazard ratio of the algorithms.

The proposed model assumes that the number of function evaluations in which the algorithms converge is independent of the number of dimensions of benchmark function and therefore this effect is included in the random term. However, if the set of benchmarks includes multiples times the same function with different dimensions and the researcher wants to investigate the effect of the number of dimensions in the number of function evaluations to converge, the number of dimensions can be added as a linear predictor to the Cox hazard model similar to how it was conducted with the effect of noise.

In our case, we assume survival functions based on the exponential distribution, however, often different likelihoods such as the Weibull, Gamma, and Log-Normal distribution can provide a better fit if the predictive accuracy of the exponential model is low.

9.5.6 Multiple group comparison of CPU time

This last model addresses the specific problem of robust multiple group comparisons. We consider RQ5:

RQ5: Is there a difference in the CPU time taken per function evaluation between the PSO, the Random Search x1, and the Differential Evolution algorithms?

The motivation for this question comes from an exploratory visual analysis from Figure 9.1. From this figure, it is clear that the Differential Evolution is slower than the others but although the box-plot suggests that RandomSearch1 is faster than the PSO, we have multiple outliers and heavy-tail distributions in the data. Besides the discussed problems in frequentist analysis (including the impossibility to accept the null hypothesis), we cannot perform a frequentist regression or ANOVA, because the residuals are not normally distributed and the homoscedasticity assumption is not met [197]. The CPU time per function evaluation is not exponentially distributed and transformations such as the log of the CPU time still present heavy-tailed distributions (additional information such as plots to support this statement are available in the appendix).

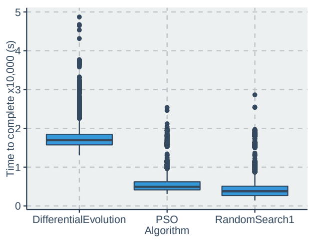

Figure 9.1: Box-plot of the CPU time to complete of the three algorithms

9.5.6.1 The model

This model is an extension of the BEST (Bayesian Estimation Supersedes the t-Test) approach presented by Kruschke [197] with added pooled random effects for each group. Each algorithm is modeled individually utilizing a Student-T distribution, and assuming that there is not homogeneity of the variances between the algorithms. In the Student-T distribution, we estimate a single degree of freedom ν parameter for all distributions. The model is represented by Model 9.6.

Model 9.6: Robust multiple comparison model

$$
\begin{aligned}
y &\sim \text{Student-T}(\nu, \mu_i, \sigma_i), \\
\mu_i &= a_{\text{alg},i} + a_{\text{bm},j}, \\
\sigma_i &\sim \text{Exponential}(1), \\
\nu &\sim \text{Exponential}(1/30), \\
a_{\text{alg},i} &\sim \text{Normal}(0, 1), \\
a_{\text{bm},j} &\sim \text{Normal}(0, s), \\
s &\sim \text{Exponential}(1).
\end{aligned}
$$

In Model 9.6, we have the following notation:

- y: is the metric that indicates the CPU time to complete one function evaluation. We compute the total CPU time of the optimization including the CPU time spent on evaluating the benchmark function and divide it by the number of function evaluations multiplied by 10,000.
- $a_{alg,i}$: represents the mean (intercept) effect of each algorithm.
- σ_i: indicates the standard deviation of the Student-T distribution of each algorithm.
- $a_{bm,j}$: indicates the random effect of the benchmarks.
- μ_i: represents the mean value of each algorithm in the linear equation.
- s: represents the standard deviation of the effect of the benchmark functions.
- ν: represents the degrees of freedom of the Student-t distribution modeling the tails of the distribution for the robustness. We initialize it with a long tail prior.

9.5.6.2 Model interpretation

After running this model in Stan (chains=4, warmup=200, iterations=3000), we get the posterior of each parameter and compute the HPD intervals for the intercept of each algorithm and its standard deviation parameter. Table 9.7 shows the obtained posterior parameters of the model and the HPD intervals.

Table 9.7: Estimated parameters of the multiple group comparison model and the respective HPD intervals

Parameter	Mean	HPD low	HPD high
a_DifferentialEvolution	1.78	1.66	1.90
a_PSO	0.57	0.45	0.69
a_RandomSearch1	0.44	0.32	0.56
σ_DifferentialEvolution	0.09	0.08	0.09
σ_PSO	0.07	0.07	0.07
σ_RandomSearch1	0.04	0.04	0.04
s	0.31	0.23	0.39
ν	2.75	2.59	2.90

The degrees of freedom parameter ν is low, indicating that when we estimate the model, the data indeed have long tails, which reinforces the need for a robust regression. If $\nu > 30$, the Student-T distribution approaches a normal distribution and indicates that the model could also be modeled with similar results by a normal distribution and that the presence of outliers did not impact much the estimation of the parameters. Additionally, σ_i parameters indicate that, due to the non-overlapping intervals, there is no homogeneity of variances, which prevents the use of the traditional ANOVA for the multiple group comparison (as it is a pre-requisite of many familywise comparison methods). The effect of the benchmarks can be estimated from a normal distribution with a mean equal to zero and with a standard deviation equal to the posterior distribution of s. The intercept parameters of the benchmarks drawn from this distribution indicate which functions introduce or reduce the CPU time to the completion of the algorithms compared to the average CPU time.

Since we want to estimate the difference between the pairs of algorithms, the difference can be calculated by sampling (in this case 10000 times) the posterior distribution of the intercepts of the algorithms and calculating the difference. This results in a new posterior distribution of the differences. Table 9.8 shows the HPD intervals of the differences between groups. The non-zero overlapping HPD intervals indicate a real difference between the CPU time to the completion of each of these algorithms.

Table 9.8: HPD interval for the difference between the groups

Difference	Mean	HPD low	HPD higher
PSO - RandomSearch	0.13	0.12	0.13
DiffEvolution - PSO	1.21	1.21	1.22
DiffEvolution - RandomSearch	1.34	1.33	1.34

9.5.6.3 Remarks

Other distributions can be used instead of the Student-T distribution for other types of robust regression, such as a double exponential distribution [95]. This robust multiple comparison model can also be easily extended to incorporate other predictors in the linear regression of each algorithm, which is not possible in non-parametric frequentist models.

9.5.7 Extending the models

The presented models in this section can answer a variety of research questions. However, different problems, research questions, and experimental conditions might require modifications and extensions that go beyond the proposed models. In the remarks subsection of each model, we discuss possible extensions specific to those models. The proposed models are aimed at being the first step into the BDA. We reinforce that the models we presented, despite the simplicity of being based on the linear regression, still address the clustering information from benchmarks and can take other predictors into account. Models based on linear regression are useful to answer questions about the direction and the magnitude of the effect of independent variables [93].

Different experimental conditions, as well as new covariates, can be controlled and their effects investigated by adding the predictor terms in the linear equation, similar to how the noise covariate was compensated in the binomial and in the Cox's Proportional Hazard models. Extending the model with transformation in the covariates, for example adding the log of the maximum budget, and investigating the effects of interactions as well as adding higher-order predictors (if the relationship with the predictor is not linear) are also possible [95, 196].

The random effects models we used only take into account the repeated measures of the benchmarks. However, higher levels can be introduced to investigate other effects [95, 209] (such as the difficulty level, if it is separable or not, etc). Adding additional clustering information follows a similar procedure as presented at the beginning of this section. Continuous random effects (instead of categorical variables) are also possible through the usage of Gaussian or Dirichlet processes as priors and are often discussed in Bayesian hierarchical modeling textbooks [95, 196].

As mentioned in the model comparison discussion (9.3.4.3), one important aspect of BDA is the comparison of different valid model candidates. A recommended approach is to start building complex models from simple ones, such as the ones presented. If the new complex model indeed increases predictive accuracy and reduces information entropy, then the complex model is more adequate. Sensitivity analysis (9.3.4.4) is also valuable to analyze how much the conclusions of a new complex model diverge from simple ones and why it happens. If the conclusions do not diverge, it increases the confidence that the results are not specific to the proposed model.

9.6 Conclusion

Bayesian Data Analysis (BDA) can address many of the shortcomings of the traditional frequentist analysis and provides a greater level of flexibility in the modeling and the transparency of the model assumptions. However, the use of BDA is not widely spread in the analysis of empirical data in the evolutionary computing community.

With this paper, we argue for the adoption of BDA and present related concepts to ensure the validity of the models. We then present and discuss a set of five Bayesian statistical models that are capable of addressing a range of different research questions, that can be easily extended for new covariates and that take into account the clustering information that the benchmark functions introduce in the results by making use of multilevel models.

Acknowledgments

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, by the Software Center and with support from Google Research Cloud Credits for Ph.D. candidates. The authors would like to thank the comments made by Erika M. S. Ramos.

Chapter 10

Paper F

Challenges and Strategies for Undertaking Continuous Experimentation to Embedded Systems: Industry and Research Perspectives

Mattos, D. I., Bosch, J., Olsson, H. H.

19th International Conference on Agile Software Development, 2018, pp.277-292.

Abstract

Context: Continuous experimentation is frequently used in web-facing companies and it is starting to gain the attention of embedded systems companies. However, embedded systems companies have different challenges and requirements to run experiments in their systems. Objective: This paper explores the challenges during the adoption of continuous experimentation in embedded systems from both industry practice and academic research. It presents strategies, guidelines, and solutions to overcome each of the identified challenges.
Method: This research was conducted in two parts. The first part is a literature review with the aim to analyze the challenges in adopting continuous experimentation from the research perspective. The second part is a multiple case study based on interviews and workshop sessions with five companies to understand the challenges from the industry perspective and how they are working to overcome them.
Results: This study found a set of twelve challenges divided into three areas; technical, business, and organizational challenges and strategies grouped into three categories, architecture, data handling and development processes.
Conclusions: The set of identified challenges are presented with a set of strategies, guidelines, and solutions. To the knowledge of the authors, this paper is the first to provide an extensive list of challenges and strategies for continuous experimentation in embedded systems. Moreover, this research points out open challenges and the need for new tools and novel solutions for the further development of experimentation in embedded systems.

10.1 Introduction

Traditional embedded systems companies continuously rely on software to be a differentiator on their products. As the software size of the products increases, these companies are moving from being mechanical producers to software companies. In their development process, these companies traditionally make use of up-front requirements and rigid methodologies to ensure quality or safety attributes in their products. Nevertheless, the requirements of several parts of their systems are not clear or cannot be defined in advance [70]. In this context, developers either negotiate with requirement teams or they make implicit assumptions about the requirements [220].

Even during the requirement specification, several requirements are written based on assumptions and does not necessarily deliver value to the company or the customers. Often, research and development effort is spent on features that are never or rarely used [107] by the users of the product. To minimize the full development of features that do not deliver value, companies make use of post-deployment data of current products to iterate in future software releases or in even in new products. In the web domain, companies provide empirical evidence of the use of continuous experimentation in their development, decision-making and feature prioritization process [2, 18, 32].

As software becomes the key differentiator for many embedded systems companies, these companies started to adopt continuous development practices, such as continuous integration, deployment, and experimentation to develop faster, better and more cost-effective products. A typical pattern that companies follow is shown in the "Stairway to Heaven" model [221]. When these companies start to move to move to continuous deployment scenarios, they see opportunities to run their first experiments as well. Although the research in continuous experimentation in web systems is continually growing, there are few examples of works investigating the use of continuous experimentation in embedded systems.

This paper identifies and analyzes the different challenges that embedded systems companies face when adopting continuous experimentation in their development processes. Moreover, it also presents strategies, guidelines, and potential solutions to overcome each of the identified challenges.

The scope of this research is captured with the following research question.

RQ: How can embedded systems industry adopt continuous experimentation in their development process?

This research question is further developed in terms of the following sub-questions:

RQ1: What are the recognized challenges towards continuous experimentation faced by the embedded systems industry?

RQ2: What are the recommended strategies to facilitate the use of continuous experimentation in the embedded systems domain?

The contribution of this paper is twofold. First, it identifies the key challenges faced by embedded systems companies when adopting continuous experimentation. These challenges are identified from both the industry perspective, through a multi-company case study, and the academic perspective, through a literature review. Second, this paper proposes different strategies and guidelines to overcome the identified challenges. This paper, to the knowledge of the authors, is the first to present an extensive set of challenges and strategies that embedded systems companies face when adopting continuous experimentation. Moreover, the analysis of the challenges points out the need for new tools and novel solutions for the further development of experimentation in embedded systems.

The rest of the paper is organized as follows. Section 2 provides a background review in continuous experimentation. Section 3 presents the research method. Section 4 presents and discusses the results in the form of identified challenges and suggested strategies. Section 5 discusses the validity threats of this research. Section 6 concludes and discusses research challenges and future works.

10.2 Background

Continuous experimentation refers to the research and application of controlled experimentation to drive software development, for reliably evaluate and prioritize development activities [32].

Studies show that the prioritization of features is traditionally driven by past experiences, beliefs, and organizational role [2, 11]. The decision to invest development resources in a full feature can result in inefficiency and opportunity cost if the feature does not have a confirmed value [5]. Companies traditionally rely on customers interviews and qualitative studies to derive requirements for the system in the early stages of the development [222]. However, customers usually are not good in predicting what they want or they are not aware of other potential solutions [70].

In the post-deployment stage, companies usually collect customer and product data. Most software companies, from both the embedded and web systems domains collects and logs usage and operational data [222]. In embedded systems, these log data are mostly used for troubleshooting and improving subsequent products. However, over the last decade, software companies are showing an increasing interest in using the collected data to improve not only future products but also to improve the current products. Recent technological trends focus on not only identifying and solve technical problems but also delivering value to their customers and users [7]. The Lean Startup methodology proposes the cycle build-measure-learn [46]. In this methodology, the collected post-deployment data is also used in the improvement of the current product. The HYPEX model [5] presents an approach to shorten the feedback loop between companies and customers. The model uses hypotheses, customer feedback and the minimum viable product (MVP) to continuously decide upon the full development or abandonment of a feature.

Web-facing companies continuously report the use of post-deployment data and controlled experiments to develop and continuously improve their systems. The uncertainty raised by the environment, interaction with humans and other agents impact in the system behavior in unknown and unpredictable ways. Controlled experiments help companies to establish the causal relationship between a variation in their system and the observed behavior [2].

In software development, A/B test is the simplest version of a controlled experiment. "A" stands for the control variation and "B" stands for the treatment variation. The treatment (variation "B") represents any point in the system that you want to modify and compare to the control (variation "A"). Both variations are deployed to randomized users, to avoid bias, and the analyzed behavior is the measured in both cases. Statistical analysis helps to determine if there is a causal difference between the observed behavior and the variations. Other experimentation techniques are described in [2].

Kohavi et al. [2] provides a guide on how to run controlled experiments in web systems. The paper discusses the important ingredients, limitations of experimentation, architectures for experimentation systems, how to analyze and how to design controlled experiments for the web. Kohavi et al. [73], presents some rules of thumb and common pitfalls when running experimentation, such as iterating in the experiment design, the impact of speed and performance, number of users and how experiments impact key metrics.

Fagerholm et al. [7] provides a general infrastructure for running continuous experimentation systematically. The RIGHT framework describes how to design and manage experiments, and how different stakeholders (business analyst, product owner, data scientists, developers, and DevOps engineers) interact with an experimentation infrastructure.

Fabijan et al. [32] describes the Experimentation Evolution Model, based on experimentation at Microsoft. This model analyzes how teams scale their experimentation from a few experiments to a data-driven organization. The model divides this evolution into four steps: crawl (teams are running and setting their first experiments), walk (teams already run a few experiments and determining metrics and experimentation platforms), run (the teams run several experiments and iterate quickly to identify effects of experiments on the business) and fly (experiments are the norm for every change to any product). Each of these phases is discussed in three different perspectives, the technical, the organizational, and the business perspectives.

One of the challenges in controlled experiments is defining an Overall Evaluation Metric (OEC) [2, 32, 223]. The OEC is a quantitative measure of the experiment's objective. It provides a balance between short and long-term effects considering the business objectives. Olsson and Bosch [223], present a systematic approach to model the value of experiments. This approach allows companies that are starting to run the first experiments to understand and improve their own OEC metrics.

To the knowledge of the authors, the first research discussing the experiments in embedded systems appeared in 2012 [64]. This paper discusses experimentation in the context of Innovation Experiment Systems. It identifies some challenges with experimentation in embedded systems, such as experimentation in safety systems, managing multiple stakeholders and hardware limitations. It also presents an initial infrastructure to run experiments in embedded systems

Giaimo and Berger [67], discuss continuous experimentation in the context of self-driving vehicles. The paper presents functional (such as instrumentation, logging, data feedback to a remote server) and non-functional (separation of concerns, safety, short cycle to deployment) requirements to achieve continuous software evolution. Bosch and Olsson [26], extended the concept of experimentation towards automated experimentation. Automated experimentation aims to leverage the number of experiments by letting the system own and control the experiments, opposed to the R&D organization. Mattos et al. [65, 66], identified a set of architectural qualities to support automated experimentation that was implemented in a research mobile autonomous vehicle.

10.3 Research method

The research process used in this study combines a literature review with multiple case study. This research method aims to strengthen the evidence of the challenges and strategies found in a multiple case-study with others found in the research literature. Research in continuous experimentation generally utilizes the case study as the research method, combining results from both approaches reinforce the empirical evidence of the findings.

The method is composed of two parts. The first part consists of a literature review in the continuous experimentation domain. This literature review collects challenges and strategies to overcome them from academic research. The second part consists of semi-structured interviews with software companies in the embedded systems domain. It aims to be exploratory, collect and confirm challenges and strategies from the embedded systems industry. Below, the research method is described in details. The results of both parts were aggregated and described in Section 10.1. Table 10.1 summarizes the research process used in this paper.

Step	Description
1	Search definition and execution (LR)
2	Papers review (LR)
3	Identification of literature challenges and strategies (LR)
4	Data selection: Contact with companies (CS)
5	Semi-structured interview protocol definition (CS)
6	Data collection: Interviews and workshop (CS)
7	Data analysis: thematic coding and categorization (CS)
8	Case study report (CS)

Table 10.1: Summary of the research method. LR stands for the literature review part and CS for the multiple case study part.

10.3.1 Literature review

The first part of the research method consists of a literature review in continuous experimentation. Although most of the studies in continuous experimentation focus on web-facing companies, the experiences from this domain, sometimes, can be extrapolated to the embedded systems domain. In this literature review, the authors identified challenges recognized in academic collaboration with industry, regardless of the industry domain. The identified challenges were discussed with the embedded systems companies to see if the literature challenges were also relevant in this domain.

Relevant works in the literature covering continuous experimentation were identified by searching the Scopus digital indexing library, by keywords, tile and abstract. The used search phrase was “((continuous experimentation)

OR (field experiments) OR (innovation experiment systems)) AND (software engineering). This search query was restricted to the fields of engineering and computer science and limited from 2000 to 2017. This search phrase resulted in 534 articles. Positioning papers and papers with less than 5 pages were excluded. From this subset of articles, the results were filtered based on the abstract. After the first screening process, the papers were read in their integrity. Continuous experimentation is also largely studied from the statistical/algorithmic side. Research papers that focused solely on improving or evaluating algorithms without industry evaluation or application were excluded.

After this screening process, the authors identified 30 articles with relevance to this study. An additional set of 12 articles were included using a snowballing [224] process, where new references were added according to the references mentioned in the other articles. Thematic coding was used to [77] identify the challenges from the literature. These challenges were categorized according to the three different categories of the Experimentation Evolution Model [32] discussed in Section 10.2, the technical, the organizational and the business perspective. The identified set of challenges were also used as input for the semi-structured interviews as discussed in Section 10.3.2. The strategies are categorized in three groups: changes in the development process, changes in the system's architecture and changes in how the experiment and organizational data is handled and analyzed.

The complete set of papers can be found at the following link: `https://github.com/davidissamattos/public_documents/blob/master/LR-XP18.png`.

This part of the research process allowed the identification of challenges that served as input for the multiple case study and confirmation of identified challenges inside the company.

10.3.2 Multiple case study

The second part of the research method consists of a multiple case study [77] with semi-structured interviews conducted with software companies in the embedded systems domain. This study was conducted from December 2016 to October 2017 with five companies in the embedded systems domain. The empirical data consists of interviews and a workshops transcripts and notes. There were 8 individual semi-structured interviews with an average of one hour each, three in Company A, two in Company B, one in Company C, one in Company D and 2 in Company E. The workshop session was conducted with 8 people from Company A lasting 3 hours. The analysis of the empirical data consisted of thematic coding of [77] interviews transcriptions and notes to identify and categorize the challenges and solutions. Additionally, during the interviews challenges identified in the literature were clarified to the interviews and asked if the current company relates to the challenge partially or not.

The empirical data were aggregated together with the identified challenges and strategies from the literature review. The current published research already provides guidelines and solutions for the challenges that were also identified in the literature review phase. Other guidelines and solutions were suggested by practitioners during the interviews. Challenges identified in the literature that was not confirmed neither through a previous case study nor by the case study companies are not shown

Due to confidentiality reasons, only a short description of each company and their domain is provided:

Company A is a multinational conglomerate company that manufactures embedded systems and electronics and provides software solutions for both consumers and professionals. This study was conducted with two teams, one providing mobile communications solutions and the other providing business-to-business products. In recent years, the company started to adopt experimentation in their software solutions and is looking for data-driven strategies in their embedded systems products. The interviewees were developers, managers and data analysts.

Company B is a multinational company that provides telecommunication and networking systems. The company is adopting continuous development practices and is looking for new strategies to deliver more value to their customers by optimizing their products. The interviewees were managers.

Company C is a global automotive manufacturer and supplier of transport solutions. As the company's products are continuously growing in complexity and software size, the company is looking for strategies to prioritize their R&D effort and deliver more value to their customers. As some employees have experience in web and pure software-systems development, experimentation is getting attention in some development teams. Challenges in experimentation arise since the company is subjected to several regulations and certification procedures. The interviewee was a senior engineer.

Company D is a global software company that develops and provides embedded systems software solutions related to autonomous driving technology for the automotive industry. Autonomous driving is an emerging and fast-moving technology and the company is looking to deliver competitive solutions faster by adopting continuous development practices. However, as it interfaces with the highly regulated automotive domain its software is also subjected to regulation and certification. The interviewee was a manager.

Company E is a global software company that develops both software and hardware solutions for home consumers. The company already has experience running continuous experimentation in their web systems and is starting to run experiments in their hardware solutions. The interviewees were senior data analysts working in experimentation in their embedded systems.

10.4 Challenges and proposed strategies

This section presents results obtained from the research process. The challenges are grouped in the three different perspectives as discussed in the Experimentation Evolution Model [32]: the technical challenges, the business challenges and the organizational challenges. The technical challenges refer to challenges related to the system architecture, experimentation tooling and development processes. The business challenges refer to challenges faced in the business side, such as evaluation metrics, business models and privacy concerns. The organizational challenges refer to challenges faced by the cultural aspect of the R&D organization.

All the strategies identified in this study are used, suggested by companies, or supported by strategies identified in previous literature case studies. The strategies are categorized in three groups: (1) changes in the development process. This refers to how companies organize their development activities. (2) changes in the system's architecture. Often restrictions in the running experiments comes from limitations in the system's architecture, that does not support data collection, or does not allow parametrization of features for experiments. (3) changes in how the experiment and organizational data is handled and analyzed. This refers to how the company stores data, comply to data regulations or use data analysis tools. The challenges are not presented in any specific order as they might reflect different challenges the companies are facing.

Figure 10.1 represents a summary of the identified challenges and strategies. In Figure 10.1, it is possible to see the relation of how each strategy relates to the different challenges, as some of them are part of the strategy of one or more challenge. This figure was obtained using the thematic codes generated in the analysis of the interviews. It maps the identified challenges within their groups with the obtained strategies groups. The rest of this section discusses each challenge individually and presents strategies to overcome them.

10.4.1 Technical Challenges

10.4.1.1 Lack of over the air (OTA) updates and data collection,

Continuous experimentation requires over-the-air (OTA) post-deployment data collection and updates. When testing a different hypothesis, the system needs to have the ability to measure the specific behavior under investigation and to update the system with the new variants as well. It is possible to run experiments without OTA, however, several experiments pitfalls can be identified in the first hours and be corrected [2]. Moreover, experiments for optimization are looking in practical significance as low as $1-2\%$ in their metrics [2, 73]. If OTA updates and data collection are not available the cost of the experiment and the practical significance level are high and the optimization process might not be worth it.

Strategies: At the moment of this study, embedded system companies are not looking into experimentation in low level systems, but in computing systems that already support modern operating systems with connectivity and the necessary infrastructure for OTA updates. OTA updates and post-deployment data collection should be part of the functional requirements of the system when designing the hardware. Mobile companies already provide such functionality in their operating systems. Car manufacturers are also introducing continuous delivery of new software to their vehicles in the context of autonomous vehicles (Tesla Motor's Model S, Volvo Drive Me and the Volvo XC90).

10.4.1.2 Lack of experimentation tools that integrate with their existing tooling

Continuous experimentation started in web-facing companies. Today several experimentation tools, both commercial and open source, are available on the website and mobile applications domains. However, in the embedded systems domain, companies lack tools that integrate with their development process.

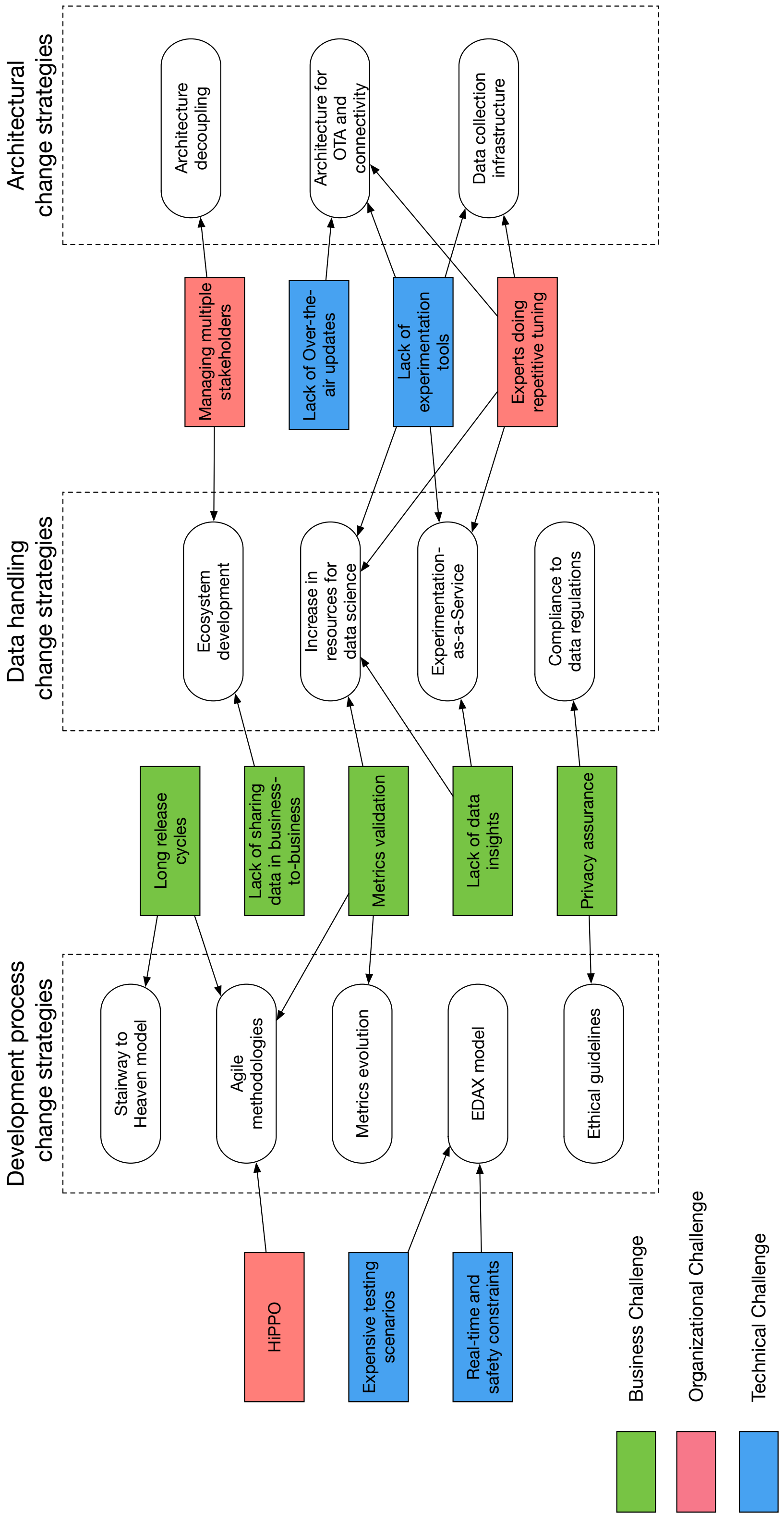

Figure 10.1: Summary of the challenges and the strategies faced by embedded systems companies adopting continuous experimentation.

Setting up an infrastructure to run experiments from scratch increases the cost of running the first experiments while hindering the benefits.

Strategies: Several tools available for websites are open source or have open source SDKs. Although not ideal, some of these tools can be modified to support experimentation problems. Experimentation-as-a-Service (EaaS) is a business model that provides a working platform for continuous experimentation. EaaS have the benefit of avoiding the cost and pitfalls of development of an experimentation platform from scratch. EaaS platforms also provide SDKs that can be incorporated in the product, such as Optimizely [1]. However, the system under experimentation should support data collection so it can be integrated with EaaS tools.

10.4.1.3 Expensive testing environments

Software-intensive embedded systems are extensively tested before release. One of the challenges faced by embedded systems companies is to include experimentation as part of the verification and validation process. In some cases, such as in the development of a new vehicle, the testing environment is expensive and not all experiment hypotheses are allowed to go to a physical testing platform. This high cost also increases minimum level necessary to reach practical significance and demotivates teams to formulate hypothesis beyond the basic requirements of the system.

Strategies: the development of experiments in the embedded systems domain require additional steps from the hypothesis to the final user. The development of a feature in embedded systems follows a testing procedure, beginning with integration and going to simulation, test beds, internal deployment until user deployment. The experimentation procedure should follow similar testing procedure, to identify early pitfalls, and even improve the system behavior during each testing phase.

The practical significance level to implement a new hypothesis increases with the associated costs of such testing procedure. The EDAX model [26] describes how experimentation and automated experimentation is incorporated in this process. Automated experimentation [65] also suggests that it can reduce the experimentation costs and therefore the practical significance level.

10.4.1.4 Experimentation constraints in real-time and safety-critical systems.

Embedded systems are employed in several real-time and safety-critical systems. These products have subsystems that are constrained to regulations and certification. Experimenting with these systems in the field might not be allowed by regulation or might impact substantially the performance of the system.

Strategies: Embedded systems companies are starting to run their first experiments. Safety-critical or real-time systems provide additional challenges, as it is subjected to legislation and certification. The initial recommendation in all case study companies is not to run experimentation in these subsystems.

[1] `https://www.optimizely.com/`

However, these safety-critical subsystems can run experiments in the earlier phases prior to the deployment, as discussed in the EDAX model [26].

10.4.2 Business Challenges

10.4.2.1 Determining good experimentation metrics and metrics validation.

One of the biggest challenge faced by companies is to determine good business metrics to understand and compare different experiments, and validate that the current metrics are aligned with the company strategy.

Strategies: Web companies traditionally rely on conversion metrics such as Click-Through-Rate in the beginning of their experimentation process. As their experimentation teams and the number of experiments increase the metrics start to become more tailored to the business and stable [32]. Embedded systems companies can have very different and complex metrics, depending on the product. However, team level optimization experiments can use customized metrics. Olsson and Bosch [223] presents a systematic approach to determine metrics and value functions for experiments. This is an iterative process that should be refined with usage and aligned with the business strategies and goals. As the metrics become complex, companies allocate of resources for the evolution and ensuring that the experiment metrics are aligned with the company's main KPIs.

10.4.2.2 Privacy concerns regarding user data.

Continuous experimentation relies on the collection and analysis of post-deployed software. However, some issues arise when collecting data, such as the legal and contractual issues or user consent and data sharing.

Strategies: Data sensitivity and the use of data vary largely between different organizations and countries. Data collection should be aligned with the legal requirements for utilization and consent of the users. Data regulations such as the European GDPR [2] create restrictions that might imply in technology and process modifications for compliance. Additionally, some ethical questions regarding the experiment must be evaluated, such as: How are participants guaranteed that their data, which was collected for use in the study, will not be used for some other purpose? What data may be published more broadly, and does that introduce any additional risk to the participants? Web companies, besides compliance with regulations also create ethical checklists to ensure that the experiments follow the companies' policies [225].

10.4.2.3 Lack of sharing user data in business-to-business (B2B) solutions.

Several embedded systems companies operate in a business-to-business domain. In this scenario, there is a difference between user and customer data. Experiments with users might not be possible, they might require deeper involvement between the companies, or there might be a mismatch between the customer and the user value [70].

[2] `https://www.eugdpr.org/`

Strategies: Ecosystems refers to companies co-opting third parties to build and leverage their products and services in such a way that they have more utility value to their customers [226]. In this sense, companies might agree on implementing and sharing data collected inside the ecosystem. Some mobile operating systems (e.g. iOS and Android) collect data and usage statistics to share with app developers. Although most of its use is connected to crash reports, similar strategies can be used to share user data in business-to-business products.

10.4.2.4 Lack of insights obtained from the collected data.

Companies are continuously collecting data from their deployed software. The collected data is mainly used for troubleshooting purposes. However, little insight is provided by the collected data [223]. In the Experimentation Evolution Model [32], web companies evolve from centralized data science teams to small data science teams presented in each product teams. The interviewed embedded systems companies don't have data science teams incorporated in the product development.

Strategies: If the experimentation benefits are not clear, the extra cost of involving data scientists in the product development might be a large step. Different companies started to provided experimentation and data analysis services. Experimentation tools usually incorporate basic statistical analysis, such as statistical significance testing, power analysis, A/A tests and more. Using experimentation and data analysis services to generate basic insights can be used as a short-term solution. Once the benefits of experimentation are clear to the company, investments such as integrating data scientists in the product development or acquiring a complex tool are easier to justify.

10.4.2.5 Long release cycles

Traditionally, embedded systems have a long software release cycle based on up-front defined requirements. Sometimes the software is deployed only once and last for several years [64, 70]. This happens due to several reasons, from both the organizational (structure and decision-making) and business (engineering effort in every cycle, requirements definition and products updates) to the technical perspective (architecture, functionalities available and support for over-the-air updates).

Strategies: From the organizational and business perspective, continuous experimentation aligns with the organizational transition to agile methodologies and the Lean Startup methodology [46]. Continuous experimentation makes use of extreme programming practices such as continuous integration, delivery and deployment to deliver experiments and new software aligned with customer behavior. The Stairway to Heaven [221] conceptual model helps companies to evolve their practices towards continuous deployment of software.

10.4.3 Organizational Challenges

10.4.3.1 Managing multiple stakeholders in the experiment design.

One of the challenges embedded systems companies face is the involvement of multiple stakeholders in an experimental design. Experimentation in embedded systems requires that the involved stakeholders understand the implications of continuous practices in their systems.

Strategies: Embedded systems require the interaction with multiple stakeholders, such as software developers, systems architects, electrical and mechanical engineers, suppliers and subcontractors. Continuous experimentation requires that these stakeholders are aware of the implications in the system design. To overcome some of these challenges, it is proposed a decoupling of the application and the underlying software and also a decoupling in time (software is not integrated at the manufacturing time) [64]. Additionally, if the interaction of the stakeholders happens in a business ecosystems perspective the experiment can be designed to benefit multiple parts [226].

10.4.3.2 Highest Paid Person Opinion - HiPPO.

Some companies are organized in vertical structures, where lower rank developers have fewer possibilities to influence and address customer's needs. Several requirements and architecture specifications are based and determined by higher paid ranks inside the company.

Strategies: This challenge is persistent in several domains and it is not restricted to the embedded systems industries. This challenge is discussed extensively in [2] among other publications. The traditional adopted strategy is to run the first experiments. Usually, experiments continuously disprove beliefs and opinions adopted by the higher paid ranks [2]. However, this requires changes in the organizational and cultural aspect of the company.

10.4.3.3 Tuning experiments is repetitive and requires highly qualified engineers.

One of the interviewed companies runs experiments for parameter optimization. The experiments rely on the system response instead of the customer response. However, running these experiments for tuning and optimization is a repetitive task that consumes R&D time and requires highly qualified engineers to perform them.

Strategies: Existing algorithms in search-based optimization, reinforcement learning and others artificial intelligence algorithms support this kind of optimization strategies. However, both the complexity of these algorithms as well as the introduced technical debt in the existing systems [104] prevent embedded systems companies to use such strategies. Experimentation-as-a-Service solutions allow companies to test Machine Learning algorithms in their system for optimization purposes. Although still in early phases, automated experimentation [65] solutions can help companies to optimize their systems through field experiments.

10.5 Validity Threats

The first threat to the validity of this study refers to the scope of the literature review. The search query was applied to the Scopus indexing library. Both the choice of the search string and the indexing library could miss other research work that can contribute to the literature review. To mitigate this threat the authors performed a backward and forward snowballing [224] process. The snowballing process allowed the authors to identify other cited work in the same area that was not identified by the search query.

An external validity to this is study is the generalization of the challenges to the entire population of embedded systems companies. To mitigate this threat, the authors sample companies producing different products in embedded systems. The authors sampled contacted multiple companies explaining the research goal, and selected only companies that are adopting/running controlled experiments in their development process were included. During the data analysis part, we reviewed all challenges only challenges that had correspondence in more than one company or that could be triangulated with the literature review were included. Challenges that could not be triangulated with other source, and that could be specific to current situation of the company, were not included in this study.

The companies that participated in this study are adopting their first steps towards continuous experimentation and are running their first experiments or trying to scale experimentation practices from a few development teams to the organization. Therefore, most of the presented challenges are faced in these first steps and cannot be generalized to companies or teams that are running experimentation at scale. As the companies evolve their experimentation practices, new challenges will arise from all three perspectives.

10.6 Conclusion

This paper addresses the question of how embedded systems companies can adopt continuous experimentation in their software development process. This question can be divided in two parts: first, the identification of problems and challenges that limit the adoption of continuous experimentation, and second selected strategies adopted by companies to overcome these challenges.

This paper identified twelve key challenges faced by embedded systems and them grouped in three perspectives, the business, the technical and the organizational. The challenges are also presented with suggested strategies to overcome them. The set of strategies can be grouped in three categories, changes that need to take place in how the company handles and analyze the post-deployment collected data, changes in the company development process and changes in the product architecture. The relation between the different strategies and the challenges is seen in Figure 10.1. The paper used a combination of literature review and a multiple company case study to provide a stronger empirical evidence.

Further research is needed to understanding how the system can be architected to support continuous experimentation as a first-class citizen in the development process while still guaranteeing safety and real-time requirements as well as intermittent connectivity. Additionally, continuous experimentation changes how the development process takes place, as it emphasizes in an outcome-driven development and this scenario might lead to impact and changes in the organization.

For future works, the authors are investigating where is the perceived highest return on investment that companies see and plan to invest to overcome the identified challenges and further support of continuous experimentation in their products.

Acknowledgments

This work was partially supported by the Wallenberg Autonomous Systems and Software Program (WASP) and the Software Center.

Chapter 11

Paper G

The HURRIER Process for Experimentation in Business-to-Business Mission-Critical Systems

Mattos, D. I., Bosch, J., Olsson, H. H.

In submission to the Journal of Software: Evolution and Process, 2020

This paper is an extension of the published conference paper: Mattos, D.I., Dakkak, A., Bosch, J. and Olsson, H.H., 2020, June. Experimentation for Business-to-Business Mission-Critical Systems: A Case Study. In Proceedings of the International Conference on Software and System Processes (pp. 95-104).

Abstract

Continuous experimentation (CE) refers to a set of practices used by software companies to rapidly assess the usage, value, and performance of deployed software using data collected from customers and systems in the field using an experimental methodology. However, despite its increasing popularity in developing web-facing applications, CE has not been studied in the development process of business-to-business (B2B) mission-critical systems. By observing the CE practices of different teams, with a case study methodology inside Ericsson, we were able to identify the different practices and techniques used in B2B mission-critical systems and a description and classification of the four possible types of experiments. We present and analyze each of the four types of experiments with examples in the context of the mission-critical Long Term Evolution (4G) product. These examples show the general experimentation process followed by the teams and the use of the different CE practices and techniques. Based on these examples and the empirical data, we derived the HURRIER process to deliver high-quality solutions that the customers value. Finally, we discuss the challenges, opportunities, and lessons learned from applying CE and the HURRIER process in B2B mission-critical systems.

11.1 Introduction

Companies are expected to continuously deliver fast, high-quality software that provides value to customers. For example, in the telecommunication domain, mobile networks are constantly evolving to support new user equipment and improve service quality. Additionally, the deployed software is becoming increasingly complex and has a high degree of interdependence with the operating environment. These aspects make it hard for the development organization to evaluate the delivered value and the quality of the software [2, 28].

With the success stories from the web-facing systems [2, 32, 158], organizations in other domains have also been moving towards continuous experimentation (CE) practices [70, 227, 228]. CE is used not only to validate the value delivered to customers but also to assess quality aspects that cannot be verified during internal development and pre-deployment quality assurance activities [229]. CE has been primarily focused on Software-as-a-Service and web-facing systems, in both research and industry [227]. Despite a few papers that explored the introduction of CE in a business-to-business (B2B) context [40, 71, 230], no publications studied the industrial usage of a CE process in B2B mission-critical systems.

Fowler defines Mission-critical systems as "in the presence of failures or degradation in the system can lead to property damage, reputation damage as well as preventing the main task to be successfully completed" [231]. Fowler [231] also points that such systems are often subject to regulations and standards (e.g. the 3GPP specification [151]). Mobile communication is an integral part of payment and banking systems, medical and transportation devices, and others which, if disrupted, can lead to severe-major failures for different businesses and society [231]. Experimentation in the B2B and mission-critical systems domain have different characteristics compared to most web-facing applications. For example, there is a difference between customer and users, ownership of the product and the data, who has control over new deployments, service level agreements, presence of risk analysis, and the impact of failure and strategies to overcome them, among others. In the mobile communication domain, the mobile operators control which version of the software will be deployed, how and when the deployment will take place. These decisions are based on the risks and benefits of the new software version. Some of the risks can be mitigated by controlled deployments. Therefore any experimentation activity requires in-depth collaboration between the development organization and the customers.

To investigate the use of CE in B2B and mission critical-systems, we conducted a case study in collaboration with Ericsson. Ericsson is a multinational networking and telecommunications company, with an R&D organization of over 24000 people [1]. Ericsson is arguably one of the largest software development companies operating in the B2B domain. By observing the CE practices of different teams, we were able to identify the key activities and derive an experimentation process that addresses the deployment of experiments with customers in the B2B and with mission-critical systems.

This paper provides four main contributions. First, we provide a classification of the different types of experiments, practices, and techniques used in

[1] `https://www.ericsson.com/en/about-us/company-facts`

B2B mission-critical systems. These techniques are discussed in the context of existing research in other domains. We identified four types of experiments that are conducted, business-driven, regression-driven, optimization, and customer support experiments. Second, we present and analyze four examples, one for each of the four types of experiments. These examples show the general experimentation process followed by the team as well as the usage of the different practices and techniques. Third, based on the empirical data we derived the HURRIER process (**H**igh val**U**ed softwa**R**e th**R**ough cont**I**nuous Expe**R**imentation), a process that combines different experimental techniques and practices to deliver high-quality solutions that the customers value. The HURRIER process can help the R&D organization to validate feature functionality, increase coverage, identify and trace stochastic faults and increase the confidence in the developed solutions much faster than without the field experiments. Finally, we discuss the challenges, opportunities, and lessons learned from applying CE and the HURRIER process in B2B mission-critical systems.

This paper is an extension of the paper "Experimentation for Business-to-Business Mission-Critical Systems: A Case Study" [69] presented at the International Conference on Software and Systems Process 2020. This paper provides the following main contributions in addition to the conference paper, based on the two new proposed research questions. First, we provide a classification of the different types of experiments, practices, and techniques used in B2B mission-critical systems (Section 11.4). Second, we provide a revised version of the HURRIER process to include relevant information to complement the different types of experimentation (Section 11.6). Third, in addition to the original example (in regression-driven experiments), we added three new examples for the other types of experimentation (Section 11.5). Finally, we have expanded Section 11.7 to include relevant discussion for the new research questions. In addition to the main contributions in respect to the conference paper, we have revised and expanded the background section and the discussion section.

The rest of this paper is organized as follows. Section 11.2 presents background information in CE and related work in CE in the B2B domain. Section 11.3 describes the research method and validity considerations. Section 11.4 presents a classification of the different types of experiments, practices, and techniques used in B2B mission-critical systems. Section 11.5 presents four examples, in the context of the Long Term Evolution (4G) product, of different types of experiments as well as the practices and techniques. Section 11.6 presents the derived the HURRIER process. Section 11.7 presents a discussion on the results addressing each of the research questions. Finally, Section 11.8 concludes the paper.

11.2 Background and related work

11.2.1 Continuous experimentation

Fitzgerald and Stol [228] provide a review on several different initiatives around the term continuous. They present a holistic view of the various software development activities throughout the entire software life cycle. These activities are divided into three phases: business strategy and planning, development and operations. Continuous experimentation acts as a link between the strategy and operations and the development, where repeated cycles of build, measure, and learn [46] guide the product improvement, evolution, and innovation inside the company.

CE aims at minimizing the risk of developing software that does not deliver value to the customer through continuously identifying, prioritizing, and validating critical product assumptions during all development phases [70]. A significant amount of research has been conducted in the context of continuous experimentation. Auer and Felderer [227] performed a systematic mapping study in CE from 2007 to 2017 and identified a total of 82 publications. In their work, they identified that most of the research with industry participation is in the context of web-facing companies such as Microsoft, Yandex, Facebook, Google and LinkedIn. Additionally, CE is discussed mainly through randomized controlled experiments (A/B testing). However, CE constitutes a group of techniques that goes beyond randomized controlled experiments [41, 232], and that can encompass many other activities and techniques.

Auer et al. [233] provide a taxonomy synthesized from a systematic literature review on the characteristics of experiments in the online domain. These characteristics are divided in five themes, overall, analysis, ideation, execution and design. These characteristics reflect on decisions during a complete A/B testing iteration, such as selection of metrics, scope and hypothesis definition, telemetry and alter conditions, as well as segmentation, number of participants and risk evaluation.

Schermann et al. [41, 229] identify two groups of goals when conducting CE activities. The first group, business-driven experiment, aims to evaluate the delivered value, the development ideas, business changes, and design decisions. The second group, regression-driven experiments, aims to identify and mitigate the impact of software changes in existing behavior, functional and non-functional bugs that evaded or can not be detected in the pre-deployment quality assurance activities, and scalability issues.

A critical aspect of CE in the business-to-business domain is the difference between customers and users. Customers acquire or subscribe to a product or service for the users [70, 71]. In the business-to-customer domain, the customers are also the users and generally acquire or subscribe to the product for themselves. Therefore, in the B2B domain, vendors usually sell products and services to other companies that sell products or services to users. A distinctive factor is that user data, product usage, and user feedback are not readily or easily available for the vendors without prior agreements. This can restrict the data collection, user feedback and even new deployments aimed at product improvement.

Yaman et al. [40] describe the process of introducing continuous experimentation in companies with an established development process using two company cases with pure software products, Ericsson and a digital business consulting company. The study investigates the introduction of experimentation in a cloud service platform. It describes relevant decision points taken (such as the target of the experiment, how to update the experiment design, etc), benefits from the experiment (new insights, reduced development effort etc) and challenges (access to end-users, inexperience with experimentation, length of the process, etc). Rissanen and Münch [71] investigate challenges, benefits and organizational aspects when introducing CE in the B2B domain. They identified that customers play a major role when designing and deploying an experiment.

In the context of mission-critical systems, continuous experimentation has started to gain visibility in the automotive domain. Giaimo and Berger [42] investigate specific software architecture and design criteria to further enable experimentation in automotive systems. Mattos et al. [31], discuss challenges and lessons from the automotive industry when starting to run the first A/B experiments. While some challenges are visible in other domains, such as the number of variants, suppliers, or the low number of users to run, others are specific to the automotive domain and do not generalize to other domains, such as restrictions imposed by the AUTOSAR architecture. However, experimentation in the automotive domain has only recently started. The current industry practices and the research state-of-art require more evidence to generalize to other companies or other mission-critical systems.

11.2.2 Experimentation processes

Several academic publications discuss the experimentation process used when introducing, scaling, and deploying experiments in online systems [5, 7, 27]. In this subsection, we provide an overview of these processes and we compare them with the HURRIER process in the section 11.7.

The presented experimentation process models take into account a two prerequisites for effectively conducting experiments. First, the ability to measure and instrument the software as well as collect this data from field. Second, the ability to deploy or configure the new software to the user based on experimental design conditions. This condition specifies that either the users will have different versions of the software based on the experimental design or that they will have the same software configured in a different way. In addition to this prerequisites, these models assume a continuous software development process with short development time and continuous delivery. The presence of Agile software development, continuous integration and deployment and towards DevOps are seen as necessary prior to a broader adoption of CE [7, 47, 221].

11.2.2.1 The Build-Measure-Learn model

The Lean Startup methodology [46] proposes an approach for companies to continuously and systematically innovate from a startup perspective. The methodology employs a Build-Measure-Learn cycle to ensure that the software development is aligned with the customer's wishes. One of the critical aspects of this Build-Measure-Learn cycle is running scientific experiments to validate customer needs and ensure that the product is aligned with these needs. The build phase reinforces the use of a minimum viable product to steer the product roadmap's direction in a startup environment. The measure phase emphasizes instrumentation needs in the products to measure users' and systems' behavior. Finally, the learn phase uses collected post-deployment data to understand and learn movements in hypothesis metrics. This methodology describes a general experimentation process similar to experiments for learning and innovation.

11.2.2.2 The ESSSDM model

The Early Stage Software Startup Development Model (ESSSDM) [47] proposes an operational support, based on lean principles, for practitioners to investigate multiple ideas in parallel and scale their decision-making process. The model consists of three steps. The first is the generation, in which the startup (or the existing company) generates ideas to expand their product portfolio. The second is the prioritization of the potential ideas in a backlog. The third is the systematic validation funnel using a Build-Measure-Learn loop similar to the Lean Startup methodology. In this step, multiple ideas can be investigated and validated in parallel. The funnel is divided into four stages: the validate problem, the validate solution, the validate the minimum viable product on a small scale, and the validate the minimum viable product on a large scale. In addition, this model supports the use of experiments for learning and innovation in a similar manner as the Build-Measure-Learn model.

11.2.2.3 The QCD model

The QCD model (Quantitative/qualitative Customer-driven Development) [48] explores the continuous validation of customer value instead of relying on up-front requirement specification. The QCD model treats requirements as hypotheses that need customer feedback for validation at the beginning of the development process. All hypotheses are gathered in a hypotheses backlog, where they are prioritized and selected for evaluation. In the validation cycle, the selected hypothesis is evaluated through both quantitative and qualitative feedback. If the hypothesis is not confirmed through the evaluation techniques, it can be refined in another hypothesis for a future iteration or abandoned. This model provides a higher-level experimentation process abstraction. It considers both qualitative and quantitative data analysis methods.

11.2.2.4 HYPEX model

The HYPEX (Hypothesis Experiment Data-Driven Development) model [5] is a development for companies aiming to shorten the feedback loop to customers. Instead of spending engineering efforts on large pieces of non-validated functionality, the HYPEX model reinforces the need for an iterative and incremental approach. The model divides the development process into six steps: (1) Generation of a feature backlog. (2) Feature prioritization and gap specification. (3) Implementation of a minimum viable feature (MVF). (4) Analysis and comparison of the actual behavior with the expected one.(5) Generation of hypotheses to explain the actual behavior of the MVF. (6) Deciding if the feature should be abandoned, iterated once more, or if it should be considered completed.

11.2.2.5 The RIGHT model

The RIGHT (Rapid Iterative value creation Gained through High-frequency Testing) [7,8] is a model for driving experiments in a startup environment. The RIGHT process model uses the Build-Measure-Learn loop to help a startup company to achieve the company's vision through continuous experiments.

Hypotheses that implement business strategies are generated and prioritized, minimum viable features or products are implemented and instrumented, and data are collected. The analysis of the collected data helps the decision-making process in a similar manner to the HYPEX model [5], where decisions to continue iterating, abandoning, or moving on to the next cycle are made. The RIGHT model describes a high-level experimentation process that can be used in innovation and learning experiments.

11.3 Research Method

The purpose of this research is to gain an in-depth understanding of the use of continuous experimentation, including the objectives, the practices, the process, and the current challenges and opportunities when applied to a company that develops mission-critical B2B systems. Based on this purpose, we formulated the following research questions:

- **RQ1**: What are the types of experiments that are conducted in Ericsson and that are relevant in the development of mission-critical B2B systems?
- **RQ2**: What are the current continuous experimentation practices used at Ericsson in the development of mission-critical B2B systems?
- **RQ3**: Can the HURRIER process can be used to drive CE in mission-critical B2B systems at Ericsson?
- **RQ4**: What are the current CE challenges and opportunities in mission-critical B2B systems observed at Ericsson?

11.3.1 The case study

This study was founded on a qualitative case study design for two main reasons. First, it allows the researchers to study and understand the phenomenon in its context in more depth [79]. Second, since CE in mission-critical and B2B systems, to the best of our knowledge, has not been discussed and investigated in research, a case study is an appropriate method for understanding a particular phenomenon in an industrial context [74–77].

We followed the five steps for a software engineering case study using the guidelines proposed by Runeson and Höst [77]: (1) case study design: the objectives are defined and the study is planned, (2) preparation for data collection: procedures and protocols for data collection are defined, (3) collecting evidence: execution with data collection on the study case, (4) analysis of the collected data and (5) reporting of the results.

11.3.1.1 The case company

This research was conducted at Ericsson AB. Ericsson is a multinational networking and telecommunications company that develops, produces, and sells telecommunication equipment, services, software and infrastructure to telecommunication operators in mobile and fixed broadband. Ericsson employs over 95,000 people in around 180 countries. Over the last ten years, Ericsson started the transition from traditional development to agile and towards DevOps. In the previous three years, CE began to get attention and promotion inside Ericsson. Although continuous experimentation is not a well-defined process throughout the company, several teams independently conduct over a thousand field experiments a year in different products and parts of the system. In Ericsson, experiments are used in many use cases ranging from innovation and new feature development to legacy assurance and performance optimization. Although this case study was conducted with a single company, we investigated CE practices in multiple teams, areas, and products spread over six locations in four countries.

11.3.1.2 Data collection

The data collected consists of a mix of different data sources, including transcripts of semi-structured interviews, notes from meetings, emails, documentation, project plans, and presentations. The first author is conducting research on-site and working in close collaboration with Ericsson employees. The second author is employed by Ericsson. He was involved in several of the project meetings and was also responsible for selecting the interview participants for this study.

We utilized a combination of criterion sampling with convenience sampling, where we interviewed the practitioners who were knowledgeable and accepted to participate in this study. We interviewed both participants that were part of the development and design of features that were validated using experimentation, as well as participants that were not involved but had extensive experimentation experience inside Ericsson.

We conducted semi-structured individual and group interviews. The interviews were both on-site and through phone conferences since the 25 practitioners were distributed in six locations in four countries. They had experience ranging from 3 to 25 years working in a range of different roles, as shown in Table 11.1. The interviews were conducted in English, were designed to last approximately one hour, and had an average duration of 55 min with a minimum duration of 41 min and a maximum duration of 1 hour and 8 min. The interviews were conducted between December 2018 and May 2019.

Since part of the interviewees were actively engaged in the deployment of a mission-critical feature that was being deployed and validated with field experiments, we conducted additional four follow-up interviews with ten members, as discussed in section 11.5.2. These interviews were used to evaluate the experience with the experimentation processes in this mission-critical feature. In Table 11.1, the follow-up interviews are identified in the second part of the table.

At least two authors were present in all interviews. In addition, interview guides were created depending and specific questions were asked based on the role of the interviewee. This allowed us to focus on their expertise and knowledge in the particular part of the experimentation process.

All interviewees were asked about their background and experience with customer experiments, live trials, data collection and feedback from both customers and users. Relevant concepts to continuous experimentation already identified in literature such as gradual releases, dark deployments were also asked

Table 11.1: Overview of the interviews

Interview	N. Interviewees	Role	Location site	Years of exp.
A1	1	Operational prd. owner	L1	3
B	1	Test manager	L2	28
C	1	Program manager	L2	23
D	1	Technical specialist	L2	15
E1	6	Developers and testers	L1	3 - 5
F1	1	Prd. guardian	L1	6
G	1	Prd. introduction manager	L2	20
H	1	Prd. introduction manager	L2	19
I	1	Prd. manager	L6	12
J	1	Prin. project manager	L2	23
K	1	Program manager for field analysis	L2	22
L	1	Customer solutions manager	L3	7
M	2	Operational Prd. Owner, Developer	L4	8, 12
N	1	Customer support manager	L2	22
O	1	Release manager	L2	24
P1	4	Technical coord., Field feature tester, CD for customer support, Project manager support	L5	3 - 15
A2	1	Operational prd. owner	L1	3
E2	6	Developers and testers	L2	3 - 5
F2	1	Product guardian	L2	6
P2	2	Technical coord., Field feature tester	L5	3 - 15

whether the subjects had previous experience with these concepts or related ones. We asked more in-depth questions about the process used, project timeline, impediments, lessons learned, perception of the benefits, and the disadvantages for all participants that have conducted entirely or partially projects with an experimentation component. For the interviews with the subjects involved in feature development, additional questions regarding the feature, the impact of the feature in the 4G product, the specific experimentation process used, results, lessons learned, among others. For interviewees in managing positions, we also asked their reflections on experimentation projects they supervised or followed and how the development organization moved towards experimentation. For participants in customer units, we asked questions regarding customer feedback and perception in experimentation projects, data collection and their current and future interest in collaborating in experimentation activities. The additional 10 interviews evaluated the use of experimentation in a mission-critical feature. Therefore for them, we asked specific questions about the project, the results, customer perception and feedback, lessons learned and impediments created by the use of a CE process and its application in a mission-critical feature.

For all interviews, the academic purpose of the study and a statement about participants' anonymity in the analysis and results were explicitly shared before the start of the interview and agreed by the participants. All interviews were recorded and transcribed for the qualitative analysis, both the questions and participants' answers. Since all authors have non-disclosure agreements with Ericsson, the interviewees could utilize internal examples and freely discuss their experiences and practices.

Additionally, we collected data from over 30 documents, including project documentation, feature development plans, solutions, and product presentations for both internal employees and external customers. These additional documents were shared, mentioned, or discussed during the interviews, meetings, or emails and were available to the authors through the internal network. These documents contained detailed information about the development and release process of different features and products, the sequence of steps taken, customer feedback on both the continuous experimentation process and specific feedback used by the development team (e.g., feedback given in the customer feedback channel of the HURRIER process). We utilized these documents as a triangulation source to support the data collected from the interviews, in particular, to help the ordering and timeline of activities to derive both the HURRIER process and answer the other research questions.

The described collected data was the main source of information to answer all research questions. The interview protocols, as well as how they connect to each research question, are available as supplementary material at `https://doi.org/10.5281/zenodo.4943011`.

11.3.1.3 Data analysis

The collected data was added to the qualitative analysis software NVivo, and thematic coding was used as the data analysis method. We utilized the six-phase process proposed by Braun and Clark [82]. We utilized inductive thematic coding, as the themes were first identified and linked to the data, rather than on previous subject coding frames.

The first phase consists of familiarizing with the data. The authors familiarized with the data in several ways, such as participating in the interview process, the transcription process, reading the transcriptions and interview notes. In the second phase, we generated the first set of codes, representing interesting concepts and ideas captured in the interviews or discussed in the additional documents. These codes identified in the interviews the goal of the experiment, perceived advantages and disadvantages, technical limitations, organizational limitations, deployment and experimentation techniques, customer perception, steps taken, deployment prerequisites, showstoppers, etc. In the third phase, we discussed potential themes for the identified codes. In the fourth phase, we reviewed the potential themes and merged similar codes when possible. Then, we classified them as part of the theme-groups: experimentation objective, experimentation practices, experimentation activities, B2B challenges, mission-critical challenges, perceived advantages, customer involvement, among others.

In the fifth phase, we analyzed each theme individually, generating the main

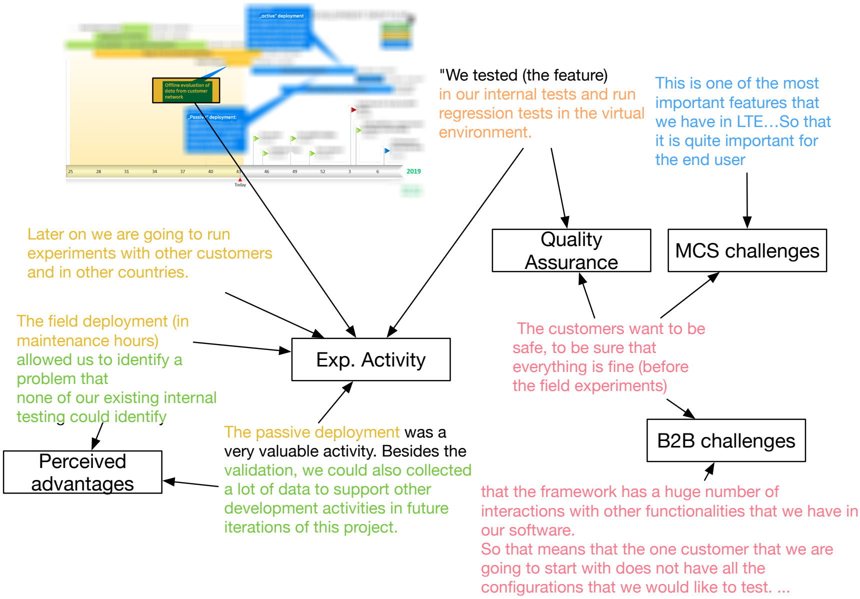

Figure 11.1: An example of the coding process where both interviews and additional data were added in the themes. In colors, we represent the different themes that emerged from the data. The text represents different quotes and color-coded to map how they link to each theme. Some quotes map into more than one theme. The project diagram refers to the plan of feature development discussed in example B in section 11.5.2. The project diagram indicates the use and order of passive deployment before the active deployment (experimentation activity theme). Parts of the project diagram were blurred to hide specific details of the project (at the company request).

results and discussion points to answer each research question. Additionally, in the fifth step, we selected, summarized, and removed company-specific information for the examples presented in this paper. The last phase consists of this publication, where the results are presented, and the research questions explicitly answered in the discussion. Figure 11.1 provides an illustrative view of how different sources of the collected data was combined in themes.

11.3.2 Identification of the HURRIER process

The HURRIER process was identified by observing the CE practices of different projects, teams, and products. Utilizing the thematic coding approach, we were able to identify the key activities performed by each team, the order of these activities, techniques used, pre-conditions, challenges and impediments, results, and lessons learned. We ordered all these activities in time and terms of importance. From these, we iteratively derive an experimentation process that addresses the deployment of experiments for all of the analyzed teams and features involved in this research. All were developed in the context of B2B. While some of them are not mission-critical, over half are considered mission-critical. In section 11.5.2, we illustrate the HURRIER process in one mission-critical project. The HURRIER process itself did not trigger the CE process used in this example, but it is our conceptualization of and a generalization of the observed CE experimentation process.

11.3.3 Validity considerations

Below we provide an overview of the main validity threats we identified and discussed the different strategies used to minimize these threats. We also acknowledge the limitations of this study.

External validity: This case study was conducted within multiple teams, areas, and products in a single software company. To minimize the bias of working with a single team inside the company, the results we identified were based on several teams' experimentation and practices in four countries. The different identified experimentation types and practices are used in specific parts of the organization, depending on the focus of the different teams. Some parts of the organization conduct only a specific type of experiment related to their specific task, such as a tuning experiment, and are not involved in customer support experiments. The practices and types of experiments are general enough to be valid in other domains since they are based on general scientific tasks used in other areas of science. However, these practices might not be identified in other companies since they may require specific conditions and maturity to emerge.

The presented HURRIER process is used in its entirety or a subset of it by the different teams and parts of the organization. We abstracted the activities of the HURRIER process performed by Ericsson in terms of common development activities used in different industries and research, such as continuous integration, passive launch, simulations, laboratory evaluations, gradual rollouts, etc. Although identified in Ericsson, the HURRIER process does not restrict specifically to Ericsson or the telecommunication industry, and therefore can be instantiated, used, or adapted by other software companies striving to introduce CE in mission-critical in the B2B domain. However, due to the nature of this study within a single company, we do not claim generalization of the process to the entire software industry.

Construct validity: The authors of this paper are well familiar with continuous experimentation practices and related research. However, the participants not always utilized the same nomenclature as research in continuous experimentation. To mitigate the threat to construct validity, the second author, who works as a full-time employee at Ericsson, was present in all interviews. When the practitioners asked for clarification or misunderstood a question, the second author explained or rephrased questions and concepts in the technical vocabulary used internally and given examples of well-known internal practices at Ericsson that exemplified the concept or question.

Internal validity refers to whether the unaccounted factors could impact the results of the investigated factors when causal relations are examined [77, 98]. Given the choice of the research method, the data collection, and the analysis, it is impossible to separate the observed results from a complex context where this study takes place or establish causal relationships. Although not causal, the inferences made in this research have been checked with *explanation building* and *addressing rival explanations* as tactics to increase internal validity in case study research [79].

Conclusion validity refers to the particular reasons, methods, and procedures we use to conclude a possible covariation between variables [33, 102]. This research does not analyze covariation between variables. The conclusions presented are based on the thematic coding analysis method presented earlier. We utilize the well-establish thematic coding procedures from Braun, and Clarke [82]. However, the themes and results obtained are not a property of the data, but they emerge from the links and understanding we make from them and cannot be separated from the researchers [82, 84].

11.4 Continuous experimentation and practices

The diversity of the data from the different teams, areas, and products that are actively conducting experimentation indicates many experimentation objectives and practices. Although these objectives and practices are not restricted to Ericsson, or the B2B mission-critical systems, they greatly impact how the experimentation process is planned and conducted.

Next, we present the types of experiments identified during the case study and used within Ericsson. Then, we present the different practices and techniques used in these experiments.

11.4.1 Types of experiments

One of the central aspects differentiating how different teams and part of the organization plan and conduct experiments is the purpose of the experimentation activity. This study identified four main groups: business-driven, regression-driven, optimization/tuning, and customer support experiments. We provide here a definition and specify how those types are used in the process in Section 11.6.

11.4.1.1 Business-driven experiments

These experiments are used to validate and assess business hypotheses by quantifying the value a particular change in the system results in and how it impacts higher-level customer/stakeholders metrics and KPIs. This type of experiment has been subjected to extensive research [7, 41, 223]. In Section 11.5.1, example A analyzes a business-driven experiment in the context of a machine learning feature in the 4G product.

11.4.1.2 Regression-driven experiments

This is a quality assurance technique where the experiment is designed to observe if the new variation or system change harms one or several quality factors and system properties. These experiments are used when laboratory and internal tests cannot assess the impact of the modification because they either cannot accurately replicate all customer's equipment configuration or deployment conditions, such as complex environments, traffic profiles, usage behavior. In Section 11.5.2, example B analyzes a regression-driven experiment in the refactoring of a mission-critical feature in the 4G product.

11.4.1.3 Optimization and tuning experiments

These are experiments done in post-deployment stages, where a system's constants, configuration or calibration parameters are modified to tune the system for a particular operating condition. This type of experiment does not require a new deployment if the necessary metrics and the configuration system are already in place. In this type of experiment, both system internal metrics and business metrics can be used for tuning purposes. In the telecommunication domain, LTE optimization with tuning experiments is a common procedure conducted by both the supplier and the mobile operators [30, 144, 234]. In Section **??**, example C analyzes the use of A/B testing to optimize a feature performance in the 4G product

11.4.1.4 Customer support experiments

These experiments are usually after a negative impact has been identified. Since the negative impact cannot always be traced back to a particular change, as it depends on the deployment conditions, usage behavior, configuration parameters, and other factors, experiments are needed to identify the root cause of the failure or negative change. Since errors in such complex situations can be stochastic, the experiments, in this case, consist of running controlled experiments with the current release of the software and one or multiple previous releases to identify where the error was introduced and which conditions trigger this error. In Section 11.5.4, example D describes the usage of customer support experiments to identify faults in stochastic scenarios in collaboration with customers in the 4G product.

11.4.2 Experimentation practices and techniques

In this subsection, we present the different continuous experimentation practices and techniques identified in the collected data. When available, we contextualize these practices in relevant research in continuous experimentation and present other techniques not discussed before. We categorize the different practices and techniques into four groups. Experimental design and analysis, variation assignment, implementation, and release techniques.

A critical difference regarding web-facing systems and business-to-consumers is the presence of two experiment levels. The first level aims to measure and evaluate the impact on the final users metrics (such as mobile phones). The second level aims to evaluate the impact of changes in the network level, such as radio base station metrics. The experiment's level impacts the choice of experiment design, the variation assignment, the implementation and the release technique.

11.4.2.1 Experiment design and analysis

An activity that impacts the planning and process for continuous experimentation is choosing the experimental design and the analysis method. At Ericsson, we identified the four experiment design types presented below.

Randomized experiments A randomized (and possibly controlled) experiment is a type of scientific method used to investigate cause-effect relationships [2, 16]. This is the most common CE practice and has wide adoption in web-facing systems. It consists of randomly assigning participants to different experimental groups. These groups differentiate by the treatment they are exposed to, while one group is held as a control group. In CE this group of techniques encompasses A/B testing and multi-variate testing. According to multiple interviews and as discussed in Section 11.5, Ericsson uses controlled experiments in regression-driven, business-driven, and customer support experiments when observing the impact of changes in mobile equipment metrics (first level experiment).

Crossover experiments Crossover experimental design is a particular type of design that the same subjects receive a series of treatments over time [39]. In this design each subject serves as its own control since we measure and evaluate within-subjects variance. One of the challenges of crossover designs is when the presence of carryover or learning effects is identified. If not controlled for that, the variance can significantly increase (or decrease), invalidating the design because the subjects change as they are exposed to different treatments. However, if the presence of carryover is known, different groups with different treatment sequences can be created to estimate the carryover effects [39]. One of the advantages of crossover designs is when a limited number of subjects do not present carryover effects. According to empirical data (both interviews and internal documentation), Ericsson utilizes crossover designs when there is a limited number of systems for an experiment. For example, in the second level, experiments evaluate the impact of changes in the network level. Crossover experiments are used in all four types of experiments.

Multi-armed bandits experiments Multi-armed bandit experiments are a particular type of experiment design aiming to minimize cumulative regret by allocating fewer users to under-performing variants. As an example, if A is the current system and B is the system with a modification. Initially, both A and B are allocated with 50% of the users. If A is under-performing B, the design shifts the user allocation to B, which would have more than 50% of the users. This type of design aims to minimize the average number of users exposed to worse variations. Mattos et al. [38] provide an overview and comparisons of multi-armed bandit designs and controlled experiments. At Ericsson, multi-armed bandit experiments are used in optimization and tuning experiments [30]

Quasi-experiments Quasi-experiments are a specific type of experiment that supports causal and counterfactual inference similarly to randomized control experiments, with the key characteristic that it lacks random assignment [33, 37]. The assignment of variations to the subjects occurs by using cut-off criteria to divide the groups. The criteria can be based on natural conditions such as demographic data or other criteria or artificial conditions such as clustering methods based on different characteristics. Since quasi-experiments do not use randomization to minimize selection bias, this can decrease internal validity since additional confounding factors can be introduced during the assignment. However, well-planned transparent designs can minimize internal validity threats. One of the key motivating factors to use quasi-experimental designs compared to randomized designs is when randomization is impractical or unethical. From the interviews, Ericsson relies on quasi-experimental designs to investigate deployments in mobile networks, when the operators are responsible for selecting which parts of the network will receive an update first, or when geographical constraints do not permit complete randomization.

Optimization Experimental optimization is a general group of CE approaches used to optimize a software based on a subject behavior (system or user). The approaches are generally used in optimization and tuning experiments to optimize system constants or calibration parameters. We group these approaches as a single practice because of their common experiment type. However, these techniques are based on very different premises and theories, ranging from Bayesian analysis [53], response surface methodology [16], Taguchi optimal designs [2, 145, 235] to search-based heuristics [51, 147]. These techniques are commonly used in network optimization and can be performed by both the mobile operators as well as Ericsson [30].

11.4.2.2 Variation assignment

The choice of experimental design influences how each variation will be assigned to the users or customers. However, even in a specific design, assignment considerations should be made. The choice of the experimental design described earlier is interconnected with how the variants are assigned in the experiment. Variant assignments also have a large influence on the implementation and release techniques and the data analysis, the causal inference, and the validity of the experiment.

The variation assignment is often not a CE choice but rather a restriction imposed by the combined effect of the type of system, restrictions on how the data can be collected, how and to whom new variations can be deployed. We identified three main assignment choices: complete randomize, cluster-based and manual assignment.

Complete randomization Complete randomization is the variation assignment practice used in controlled experiments and crossover designs. It consists of assigning the system with or without the modification randomly to all units or users. This practice allows the differences between the units to average out as they are randomly sorted, and therefore the observed changes are due to the modifications on the system and not by individual differences between the units. Although this practice has a higher internal validity, it is not always possible to conduct. Complete randomization is common in level 1 experiments (with the final users) inside a single or a similar group of radio base stations. From documentation and interviews, such functionality is implemented in specific features of the product.

Cluster-based randomization Cluster-based randomization is a type of variation assignment technique that divides the population into clusters that are randomized together [20,236,237]. This kind of randomization is common in natural restrictions such as geographical location, as discussed in the interviews.

Automatic assignment Automatic assignment is a type of variation assignment that is based on a criterion that is not necessarily randomization. For instance, multi-armed bandit systems can use different metrics to perform assignments, such as the upper confidence bound [38]. Also, optimization experiments can use a different range of variation assignment heuristics such as the expected improvement [53] or nature-inspired exploration techniques [147].

Manual assignment In B2B, the R&D organization might not have control over the assignment process. In such cases, they rely on the customer to assign how each variation will be assigned to users. In this manual process, the customer utilizes existing metrics to create two comparable groups. This manual process might also be aided by matching tools such as propensity score matching [106].

11.4.2.3 Implementation techniques

As part of the design of the experiment, the R&D organization, in collaboration with the customers/users, might decide on different implementation techniques to deliver the software change. We refer to implementation techniques as to how the modification in the software are implemented to be activated to the customers/users.

Feature toggles Feature toggles are conditional statement blocks in the code that allows enabling and disabling features based on configuration parameters [229, 238]. In the telecommunication domain, feature toggles are extensively used by customers and by the R&D organization to configure and customize specific parts of their network. Besides the configuration flexibility, feature toggles facilitate both the R&D organization and the customers to conduct and launch experiments without needing a new software build or a new deployment.

Software versions The simplest form of deployment of a software modification for experiments is to generate different software versions and deploying them manually. Although such an approach might not be feasible for large scale experiments, in the second level of experiments (the network level), the sample size is small, and this technique is viable. Additionally, customer restrictions in the deployment strategy can restrict other techniques such as feature toggles. As observed in the interviews, this strategy is more common in the first iterations of prototypes with customers.

Traffic routing The technique consists of redirecting requests randomly between different instances of the system (one with the modification and one without) [2,41,229]. This technique is commonly used in controlled experiment designs and in websites and back-end systems where concurrent experiments are low. However, it can provide low experimentation scalability or degradation on the user experience [2]. In the context of 4G product, this technique is not used or recognized among the interviewees, as redirecting traffic to different radio base stations is not practical and can severely impact the whole network. However, this technique can be used in other web-facing products developed by Ericsson.

11.4.2.4 Release techniques

After having the software ready for an experiment, the R&D organization decides together with the customer how this software will be released in the field. These release techniques are often risk minimization techniques aimed at preventing negative effects that have a large impact. If a negative impact is not observed, the scope of the release is increased.

Canary release Canary releases are a practice of deploying a release to only a small percentage of the active customers and monitoring the behavior of the system for negative impact [41,239,240]. Since large negative effect sizes can be observed in smaller samples, this technique effectively minimizes the exposure of a negative change to the general population [16].

Passive deployment or dark launch Dark launch, also known as shadow, dark or passive deployment, is a technique where the new piece of functionality is deployed invisible to the customer [41,239,240]. This means that the silent modification is exposed to real-world data but without acting on the system. The modification can be analyzed to see how it performs in a production environment and compares it to the existing solutions. While not a standard practice, this solution has been explored at Ericsson, as seen in section 11.5.2.

Gradual rollouts Gradual rollouts consist of gradually increasing the number of customers exposed to the new release [241]. This technique is commonly used together with canary releases. When no negative impact is observed, the sample of customers exposed to new release is gradually increased [23,41,240]. This technique is also known as ramp-up and is commonly used in combination with A/B testing. Gradual rollouts are commonly used by internal development and customers in project plans and documentation.

Ring-based releases Ring-based release is a common release technique in experimentation, where the software change is deployed to customers that have previously agreed with the release conditions [242]. The inside rings are exposed to a faster deployment cycle with new features and bug corrections at the expense of less stable builds. The problems identified earlier in the inside rings are corrected, and slower and more stable builds are released to customers in the outside rings, such as the General Availability (GA) features to all customers. Although not referred by this name, ring-based release is often mentioned in documentation and meeting notes.

Time-window releases Time-window release is a technique where the system modification is released only when the application has a lower risk of negatively impacting users. Usually, the period corresponds to the lowest traffic/usage period or in maintenance hours. Similar to canary releases, this technique is also based on the fact that large negative effect sizes can be observed in smaller samples. As the development organization gains more confidence in the system's performance with live but low-risk data, the application can be rolled out gradually to larger time windows. This technique is often combined with other techniques, such as canary releases or gradual rollouts in the whole period. While not a standard practice, this solution has been explored by both customers and at Ericsson, as seen in section 11.5.2.

Table 11.2 summarize the types of experiment and practices identified in the empirical data.

11.5 Examples

This section presents four examples that investigate or utilize the discussed experimentation practices in the context of the Long Term Evolution (4G) system. The identified experimentation activities and techniques presented in Section 11.4, the presented examples lay the foundations of the inductively derived HURRIER process described in Section 11.6.

11.5.1 Example A: Business-driven experiments

This example investigates the development of a new machine-learning feature aimed at replacing an existing solution. The new feature utilizes a machine-learning algorithm to provide faster and better hand-over decisions for the user equipment. Although the team also performs regression-driven and tuning experiments, we focus here on the business-driven experiments as captured in the quote below:

> *““We cannot prove the feature benefit for the customer in the lab, as it is very hard to simulate the customer network and traffic patterns... Also, customers value different aspects of how the feature impacts their network. So this type (business-driven) of experiment is the most valuable for us.””* — Interview M

This feature came from developments in research and was selected by the team to proceed to the pre-study phase. The pre-study phase consisted of simulations to support (and replicate) the research results and determine the instrumentation and data collection.

> *“ “The idea came from research. We had some simulations but nothing concrete to become a product. The question was not only if the idea would work in a live network but also if it is possible to do it in an embedded system. How much data would we need? Would the (ML) models fit the hardware constraints? Some of these questions we could answer before doing a prototype.””* — Interview M

Initially, the prototype supported only the data collection in collaboration with a customer. This data supported the iterative development of the machine learning models and the introduction of the software to the embedded system. After several iterations in the prototype led to a first minimal viable feature that could be deployed in a live network. The first deployment of the feature was done with passive deployment, where both the customer and the development team could verify the behavior of the feature. Given the positive impact observed

Table 11.2: Overview of the types of experiment, practices and techniques

Type of experiment	**Experimental design**	**Variation assignment**	**Implementation**	**Release**
Business-driven	Randomized	Complete randomization	Feature toggles	Canary release
Regression-driven	Crossover	Cluster-based	Software version	Passive deployment
Optimization and tuning	Multi-armed bandit	Automatic assignment	Traffic Routing	Gradual rollouts
Customer support	Quasi-experiments	Manual assignment		Ring-based
				Time-window

in the passive deployment, the prototype was tested in a limited number of radio base stations. The evaluation of this deployment was conducted in two ways, first through a quasi-experiment with a manual assignment and second in a crossover experiment. For the quasi-experiments, the customer manually assigned two comparable zones for the deployment. In one zone, the feature was activated, and in the other, it was in a passive deployment (the feature was available together with the instrumentation but not impacting the network). For the crossover experiment, the active zone was compared against its own metrics baseline, while the passive zone was monitored to see if there were changes in the metrics baseline.

> *""This feature started as a prototype. We had the idea of what we wanted to do. After a few experiment iterations, we collected some field data and analyzed it. Together with good customer feedback, we got strong support for it to become a product.""* — Interview M

With the expansion of the prototype to a product, the R&D team is able to conduct multiple customer validations and optimization experiments to improve the feature and the machine learning models.

11.5.2 Example B: Regression-driven experiments

This example investigates the usage of CE in a project that consists of the refactoring of a framework that implements several functionalities of the 3GPP specification [151]. This framework is used to increase the speed, coverage, and capacity of mobile communication systems. It is considered a mission-critical system in the LTE context, i.e., without the proper function of this framework, critical functionalities of the 4G are compromised with possible mobile traffic disruptions for the affected region. The refactoring procedure aims to increase performance, scalability of the system for new solutions and specification modifications, support for a number of new future user equipment, and open the space for new machine-learning and artificial intelligence solutions. The importance and critical aspect of the framework are captured in the quote:

> *""This is one of the most important features that we have in LTE ... as it has a great impact for the end-user""* — Interview F1

The framework is highly complex as it interacts with over 20 different functionalities in the LTE system. It needs to interact and perform well with over the 5000 different configurations of user equipment available in the 3GPP specification and the additional new 5G systems. Combined with the different traffic profiles and optimizations that mobile operators can have, verifying and validating this system in all different conditions in-house is unfeasible. In terms of cost to create such a testing facility, the evolution of the testing facility to include the continuously increasing number of user equipment and the time to validate the solution. Internal testing of the framework can verify functionality interaction of the framework with new features and how new features can impact the framework. However, it is not possible to achieve high coverage and quantify the improvement of the solution without running field experiments in a customer network. This project involved design teams, development units, and customer units in 4 different countries.

> *" "We want to test it in the field with the customers ... the main reason is that the feature that we are working on is highly dependable on the configuration that is used in the field, and the configuration is different in different countries. It is different between operators in the same country, and it is different between different types of user equipment that we have on the market. And due to that, it is very hard and probably impossible to verify all the functionality with internal testing, or in our lab. In our lab, we only have a limited set of user equipment, a limited set of configurations. So to secure the quality of the product that we are delivering, we want to have the ability to deploy that in some of the customers' networks, before we go full scale.""* — Interview F1

During the pre-study phase, one mobile operator had a network profile that could be used to validate a large part of the refactored framework and had a high interest in the evolution of the system after the framework has been deployed. The operator provided initial field data to aid the initial stages of the development. Following an incremental approach, the first version of the new framework was developed. In parallel, the R&D organization worked on securing that internal verification is set up to cover major use cases and apparent configurations. The internal feedback channel was implemented with the existing Ericsson procedures for quality assurance, including CI reports, simulation status, and laboratory validation tests.

> *""We tested (the feature) in our internal tests and run regression tests in the virtual environment. The field is a second step for validating. The customers want to be safe, to be sure that everything is fine (before the field experiments).""* — Interview E

The customer feedback was implemented together with the local customer unit. It contains both automated data collection of the instrumented software in addition to an ad-hoc manual collection of further performance and diagnostic data, if necessary. Because of the critical aspects of the framework, the single customer validation followed all activities. After passing the customer laboratory evaluation, the passive launch activity allowed benchmarking the responses from the new framework with the existing one. This activity allowed the R&D organization to identify corner cases from the live network traffic, that were not covered during the internal validation and customer laboratory. The frequent feedback and iterations allowed the software to have enough confidence for a restricted launch in the live network.

> *" "The passive deployment was a very valuable activity. Besides the validation, we could also collect a lot of data to support other development activities in future iterations of this project.""* — Interview F2

The restricted launch enabled the framework only during maintenance hours for a week. Maintenance hours correspond to lower traffic and requirements on the network, making it a lower risk scenario in cases of faults. The analysis of the restricted launch was made by comparing key metrics and time series of the maintenance hours of the experiment week against a control week with the old framework. When the new framework reached enough quality level in key metrics, it proceeded to the gradual rollout.

> *" "The field deployment (in maintenance hours) allowed us to identify a problem that none of our existing internal testing could identify""* — Interview F2

The gradual rollout followed the existing customer deployment plans for new software, where the software is deployed in groups, and the performance of each group regarding KPIs is measured before the deployment of the next group occurs. In this stage, quantitative feedback regarding the KPIs ensured that the new framework behaved as designed.

> *" "Later on we are going to run experiments with other customers and in other countries. The reason for that is that the framework has a huge number of interactions with other functionalities that we have in our software. So that means that the one customer that we are going to start with does not have all the configurations that we would like to test. ... If we get positive results from this first customer we want to expand that to the other customers""* — Interview A

In parallel with the first (single) customer validation, the R&D organization contacted other customers for further field experiments. The contacted customers that also showed great interest in the framework had different network profiles to increase the coverage of the solution. Since the first customer validation already validated the most critical aspects of the framework, the R&D organization performed the gradual rollout (after the customer verify the software in their own laboratories). Therefore, these new field experiments with multiple customers were aimed at increasing the coverage and confidence in the framework in special and corner cases.

The field experiments with multiple customers generated enough data and evidence for the R&D organization to proceed towards final documentation and release of the framework in GA for other customers.

11.5.3 Example C: Optimization and tuning experiments

This example discusses a feature developed in the 4G system to improve the configuration parameters of existing features utilizing experiments. The feature consists of an A/B testing framework that performs randomization in the first level (user equipment). The feature can randomize the users in multiple groups, where each group receives a different set of parameters. The feature utilizes existing performance metrics and relies on the configuration parameters already implemented in the system.

> *" "You can see exactly how each set of parameters impact the system at the same time, so you don't have the disadvantage of running the experiment for one week, and a holiday changes the traffic profile and the datasets are no longer comparable""* — Interview D

The parameters to be investigated comes from team-level performance goals, where it is hypothesized the expected impact and the proposed modifications. If the feature is already deployed in the customer network, it does not require a new deployment. However, the customer manually decides upon selecting the network cluster where this feature will activate (second level). Suppose a particular set of parameters outperform in this limited release. In that case, it can be gradually rolled out to other parts of the network, where the aggregated effects can be investigated in a crossover experiment.

However, optimization experiments are not exclusive of feature configurations but are also possible at the network level (second level) and even without the participation of the R&D organization.

> “ *“The customers tune their network over many years. They have a lot of knowledge over the dependencies between the features and configurations they have in their network. However, new customers or customers that do not have the same maturity in tuning the network will benefit from tuning expertise from our R&D teams.””* — Interview D

11.5.4 Example D: Customer support experiments

Customer support units are often responsible for managing the feedback between the mobile operator and the development unit. After the general availability of the software, customers can still find unexpected behavior or metric degradation. If the source of degradation can be traced, software patches are created. The delivery of the software patch is often done with regression-driven experiments using a crossover design experiment by the customer support units.

> “ *“We deliver corrections to customers on issues that they have but since we are not able to fully verify the patch without the specific configurations, profile of the network and customer KPIs, we do regression experiments””* — Interview N

However, customer reports of degradation and faults might not be able to specify the part of the system the failure was observed or even the version of the software where the fault was introduced due to changes and the stochastic behavior in the operational environment. In such cases, the customer support unit conducts “customer support experiments” to identify the source of the fault so the development unit can provide the appropriate software patch. Customer support experiments are used to diagnose faults in the software and configuration issues in the network.

> “ *“In troubleshooting, not everything we get is a software issue. Sometimes it is a configuration fault. And we need to do changes and run experiments on the live network to identify the source (of the fault/behavior)” ”* — Interview N

The customer provides specific requests to the R&D organization. The customer unit acts as the direct customer feedback channel. The request is studied to see if they are able to identify the source of the fault. If the fault can be traced, the development unit provides software corrections experimented in a regression experiment. Otherwise, the customer unit conducts direct experiments with the customer. Customer experiments are usually quasi-experiments, with a manual assignment and with software versions or feature toggles. The use of software versions is to verify which version the fault was introduced, while feature toggles help identify the features.

When the source of the fault is identified, and the development unit provides a patch, regression experiments are conducted to verify the behavior of the patch before full deployment to the whole network and other customers.

11.6 The HURRIER Continuous Experimentation Process

The deployment of new software in B2B and mission-critical systems, unlike many web-facing applications, requires extensive verification, risk analysis, and customer approvals. The R&D organization must comply with specifications, pre-established testing procedures, and service level agreements. In these situations, field experiments must be planned and agreed upon in collaboration with the customers to ensure that the R&D organization receives adequate field feedback and minimize the risk imposed on the service provided by the customer. In this section, and based on the empirical data, we present the HURRIER process for conducting experimentation in mission-critical features in the B2B domain. The HURRIER process was identified and formulated based on the current experimentation process and practices of different teams inside Ericsson, as discussed in sections 11.4 and 11.5. The process is used in its entirety or just a subset of its activities, depending on the scope and area of the development project.

The process is composed of a set of generic activities that can be organized in four main areas around two feedback channels. The areas are: (1) the R&D organization, (2) the internal validation, (3) single customer validation, and (4) multiple customer validation. Next, we discuss each of these groups in detail, the set of commonly found activities in these groups, and the feedback channel. Figure 11.2 shows an overview of the HURRIER process. In this process, the square boxes represent activities. The thin arrows indicate the sequence of the activities inside an area. The thick arrows represent feedback data. At any point in this process and based on the feedback data, an activity can be interrupted and returned to the R&D organization, either in the form of new requests and ideas, by adding new use-cases and providing additional data for the pre-study activity, or providing continuous feedback for the incremental development activity.

11.6.1 The R&D organization

The R&D organization is responsible for developing the feature or change that will be deployed. The R&D starts after a development idea is generated. These development ideas can come from different sources such as direct customer request, market needs, competition, innovation, or to meet the internal goals of the R&D organization.

11.6.1.1 Pre-study

The pre-study activity consists of scoping the project and planning its development. Metrics and success criteria are determined, and the expected improvement in both system internal metrics and customer level metrics. The feature is divided into incremental steps that can be rapidly evaluated in the field in collaboration with the customer. In this activity, potential customers are selected to evaluate the first experiments. These customers usually have a significant interest in the specific feature, either because of their request or potential benefits. Customers in this step can have different degrees of interaction, from an observer role to a more active role in the design of field experiments.

> *" "Some customers understand the need for field experiments in their network. Because with experiments they can access earlier the latest functionalities and improvements of our software.""* — Interview C

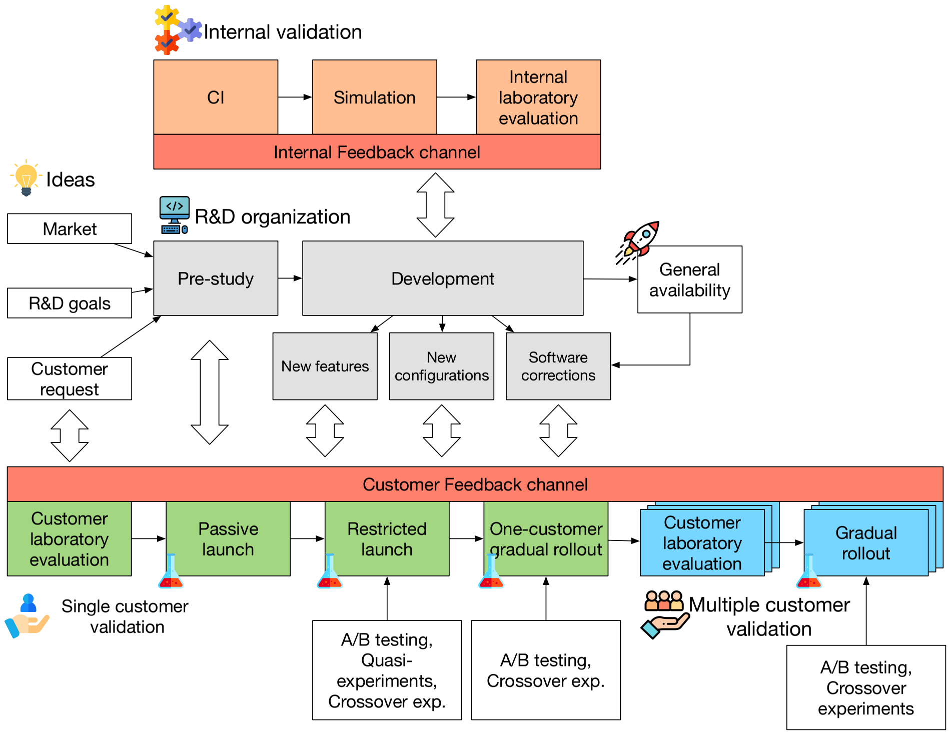

Figure 11.2: The HURRIER Process. The different activities in the process can be organized into four main areas: (1) the R&D organization (in gray), (2) internal validation (in orange), (3) single customer validation (in green), and (4) multiple customer validation (in blue). The internal feedback channel provides continuous feedback from the quality assurance activities, while the customer feedback channel provides feedback from field experiments back to the R&D organization.

11.6.1.2 Incremental development

After the pre-study, the development activities start. We divide the development activities into three main groups, as discussed next.

New feature New features are intended to introduce new functionality in the system. After the pre-studied, the feature is divided to be implemented incrementally and with constant feedback validation from both the internal procedures and feedback from the field in close collaboration with the customer.

New functionality is often designed as prototypes with a very limited scope to have a first field validation before going through new iteration cycles.

> *" "We always start with understanding the problem and trying to solve it in the simplest possible way, and then before we start an expensive simulation or a study how this interact with everything else to fully understand it, we try to build a prototype and test it (in the field), the minimum scope, with tremendous amount of limitations. Possibly, the prototype does not work with this or that feature, but at least we can test if there is a gain or not... We are not more efficient when it comes to building time, but we know in advance that it works (in the real-world). That is the benefit of prototyping and real world experimenting.""* — Interview I

New configurations New configurations consist of a larger category of changes. It includes changes in existing static parameters of features, changes in configuration parameters of features, or in proposing new configuration parameters for the network. This type of activity usually does not require extensive internal laboratory evaluation if these configuration parameters are still in a predefined validated range. Some configuration proposals do not require a new software build or even a new deployment since they can be modified through existing interfaces in the software.

Software corrections Software corrections consist of bug fixes and other modifications to address identified functionality faults or metric degradation. If the fault is deterministic, then new test cases, simulation scenarios, and laboratory procedures are introduced. However, as discussed in Section 11.5.4, faults can present stochastic behavior or be specific to a particular network configuration. In these cases, validation of the software correction is done in collaboration with the customer.

11.6.2 The internal validation

The internal validation consists of quality assurance activities. These activities are performed both before and in parallel with the customer validation and the field experiments. Before the internal development reaches the customer for a field experiment, the R&D organization conducts a series of internal validation procedures to guarantee a minimum quality for first field deployment. The internal validation before the customer field experiments is not aimed at reaching a high degree of coverage as provided by a field evaluation. Instead, this internal validation aims at capturing integration problems, interaction with other features, and other common implementation errors. The first iterations of the internal validation are considered a fast procedure. It targets guaranteeing an acceptable level of quality while minimizing the leading time to deploy with the customer. In parallel to the customer validation activities, quality assurance teams incrementally validate the development.

11.6.2.1 Continuous integration

Continuous integration (CI) is a mandatory internal validation activity. This activity aims to identify integration problems and interaction of the feature under development with a range of other features and many different hardware configurations. This activity is part of all deployment processes at Ericsson, regardless of the presence of field experiments or not. Continuous integration at Ericsson has been discussed extensively in [243].

11.6.2.2 Simulation

The development of some features can be verified using simulators. Ericsson has modeled several characteristics of its products and of the different environments that the products can be deployed. The development of a simulator is an intensive activity that requires extensive validation. Additionally, not all conditions can be easily simulated or have a simulator available, and for those conditions, an internal validation with simulation is not used.

11.6.2.3 Internal laboratory evaluation

Similar to the simulation activity, Ericsson has several laboratory testing environments. The laboratory environments are designed to capture and verify a large number of use cases, including software and hardware integration. These cases give a good indication of how the deployed system will perform under controlled circumstances. However, sometimes the feature under development addresses corner cases or requires the complex interaction of live network traffic. Those cases are often hard and expensive to recreate in an environment laboratory, and in those cases, the feature requires field experiments for validation.

11.6.3 Single customer validation

The internal validation provides software with enough quality and verification to be deployed in the field. However, as discussed previously, internal validation activities cannot cover all the quality aspects of the system. Time and costs constraints impose a limited number of scenarios that can be run in simulation and laboratory evaluation. Additionally, controlled environments lack the high complexity and variability seen in field deployments, such as in traffic patterns, types, and a number of user equipment. They cannot assess customer-specific KPIs (key performance indicators). The first field validation is done in collaboration with a single customer, and it is usually the same one that has been involved since the beginning in the pre-study phase. This customer has a high interest in the development and success of the field experiment and, therefore, collaborates to share field data in qualitative and quantitative feedback.

11.6.3.1 Customer laboratory evaluation

The first activity after the internal verification is the deployment of the software for evaluation in the customer laboratory. The customer laboratory is run by the customer and contains specific configurations to replicate its own network. The customer can verify the software with its internal test procedures and Key Performance Indicators (KPIs) in the laboratory. This step gives confidence for the customer to deploy the software in its own network. This evaluation is also considered a fast procedure since it does not cover all cases.

11.6.3.2 Passive launch

Passive launch, also known as a dark launch or a dark deployment [41, 244], is a CE technique that consists of deploying the new feature or change in parallel with the existing system. The new feature performs its task in the background, and it is executed by the same traffic profile and inputs of the system. However, its output and its main functionality are not exposed to the users. It is used to provide an open loop verification of how the feature would behave in production in systems where response parity is necessary, and the correctness of the response can be evaluated. This activity is not mandatory, and it is often seen as a time-consuming activity for smaller features and changes. However, in mission-critical development, a passive launch can increase the confidence level in the deployment, verify the system response, memory and CPU usage, and the quality of the response with minimal to no risks for the end-users.

11.6.3.3 Restricted launch

Restricted launch corresponds to the deployment of the system in a low-risk scenario that can help validate the development. The restricted scenario can be the selection of systems so that if they fail, then the impact is small on the final users (such as systems with high redundancy and safe fails). Additionally, the restricted scenario can be a restriction in time. The new feature is deployed only in low-risk periods, such as maintenance hours or low traffic hours. If the deployed systems in the restricted launch are compared to other equivalent systems, the customers and the R&D organization usually follow a quasi-experimental design [33]. If the system metrics are compared with the historical metrics or metrics after the restricted launch, the R&D organization and customers plan for a cross-over experimental design [245]. If the tracked KPIs are end-user dependent, the R&D organization and the customers can utilize A/B testing or another randomized factorial design to evaluate the impact of the feature. In this last case, negative movements are further investigated, while statistically non-significant and positive significant movements give confidence for proceeding towards a larger experiment.

11.6.3.4 One customer gradual rollout

After a successful restricted launch, the customer has enough confidence to make the gradual rollout of the feature to the whole network. This rollout can be randomized, from lower to higher risk systems, or use another pre-established procedure. Together with the manufacturer, the mobile operator can decide to run cross-over experiments or A/B experiments between each ramp-up or select two previously known regions with high correlation to evaluate the deployment. At the last stage of the gradual rollout, a full experiment is conducted to evaluate the deployment. This evaluation can be used to verify the value of the deployment and the quality aspects of the deployment. At this stage, the R&D organization can decide to continually develop the feature towards its full scope, running additional single and multiple customer experiments, or abandon the idea and development and move to the next feature.

11.6.4 Multiple customer validation

After one customer validation, the R&D organization aims at iterating on the feature to deliver values to multiple customers.

Suppose the field experiments with the first customer already provide enough coverage and confidence in the solution regarding quality or value. In that case, the R&D organization can decide to mark the feature for general availability (GA), which means that the feature has the adequate quality and therefore ready to be deployed by any customer. However, often a single customer cannot cover all the necessary validation aspects of the feature, and the R&D organization may select a number of additional customers that can increase this coverage for more field experiments.

> *" "We have some experiments with teams where they have the customer involvement all the way (from early development). The problem with that is that we develop features that are supposed to be globally and possible to use for all our customers.""* — Interview B

Since the feature has already gone through field validation and has higher confidence, some steps such as the passive and restricted launch are usually not covered. The customer laboratory and the gradual rollout are similar to the single customer validation. However, the R&D organization may focus on identifying the corner cases to increase the feature coverage and the delivered value by choosing different experimental designs. Additionally, due to the significant differences between each customer network, these experiments are analyzed individually and not combined.

After the multiple customer validation, the feature is fully documented and marked for GA. At this stage, other customers can acquire or deploy the feature in their networks fully or partially, gradually or at once. However, the feedback to the R&D organization takes longer, as the development has moved towards a new idea and other experiments.

11.6.5 The internal and the customer feedback channels

In the B2B context, the software manufacturer cannot always have direct access to field data or user data. The HURRIER process centers all activities around two main feedback channels to ensure the development team has access to all necessary field data.

The internal feedback channel consists of reports and communication between the development teams and the operation teams. This feedback channel provides quantitative data from the quality assurance teams, such as continuous integration status, simulation reports, and laboratory test reports.

The customer feedback channel is an agreed communication channel between customers and the R&D organization. It serves as a central source of feedback that developers can get from the field validation activities with customers. The provided feedback can be both quantitative and qualitative data. The customers can control what information they provide, facilitating compliance with specific regions' regulations, such as GDPR, guaranteeing the anonymity and privacy of their users, and controlling business-sensitive information. If a particular development activity requires sensitive data from users, the customers can agree to run analysis scripts on the data and only provide feedback. The customer feedback channel can be implemented in several ways, from automated data collection of instrumented software, direct contact between developers and customers, or specialized customer units.

11.7 Discussion

This section discusses the research questions and the challenges and lessons learned from retroactively applying the HURRIER process in the presented examples.

11.7.1 RQ1:What are the types of experiments that are conducted in Ericsson and that are relevant in the development of mission-critical B2B systems?

In Section 11.4.1, we discussed four types of experiments that are commonly conducted internally at Ericsson. All four types of experiments are conducted in mission-critical B2B systems in close collaboration with the customers, as exemplified by the examples in Section 11.5.

Business-driven experiments have been extensively researched in the context of web-facing systems [2,7,32,72,227,232,246] and more recently in the context of embedded and automotive systems [31,64,67]. Business-driven experiments are often used interchangeably with A/B testing in the web domain and often, but not limited to, for changes in user interfaces.

The usage of regression-driven experiments has been discussed earlier by Schermann et al. [41,229]. They discuss the usage of regression-driven experiments to detect functional problems that were not captured in internal testing, performance regression, or testing the scalability of a feature. In our work, we reinforce these aspects and present them as an integral part of the release process in the HURRIER framework.

Tuning and optimization experiments are widely used in web-facing systems. However, they are often categorized as business-driven experiments despite having critical differences in the planning and conducting process. This distinction becomes more evident when the customers have higher requirements in the product configuration for their use, such as in mobile networks. In the telecommunication domain, optimization experiments are widely discussed, and [144] and methods to automated this process is a current subject of research [30,234].

Customer support experiments have not been discussed in research literature before and present a new concept in experimentation. However, as companies start to introduce computational intelligence features based on machine learning, the interdependence between the system, the data, and the operational environment will increase and lead to more stochastic faults and degradation that requires customer support experiments to identify the source of the problem.

11.7.2 RQ2: What are the current continuous experimentation practices used at Ericsson in the development of mission-critical B2B systems?

In Section 11.4.1, we discussed several experimentation practices and techniques identified in our data collection and contextualize them with existing research. These practices and techniques were classified into experiment design and analysis, variation assignment, implementation, and release techniques.

In the design of the experiment, research and web-facing companies have often focused on controlled experiments (A/B testing or multi-variate testing) due to the control over the deployed software version and the higher number of users. However, as observed in the examples and the HURRIER process, crossover and quasi-experiments have a significant role in experimentation in B2B and mission-critical systems at Ericsson. In the B2B domain, experiments need to be conducted with close collaboration with the customer. In these cases, the customer often decides on the variation assignment and the sample size. Additionally, features and metrics in the second experiment level (network) can also have geographical restrictions. These restrictions limit the experimentation design to crossover or quasi-experiments and the variation assignment to manual and cluster-based assignment.

Although the variation assignment and release techniques are similar to what is observed in other domains, there are some differences in B2B mission-critical systems. For example, the automation of the release and rollback of software is often undesired and restricted by the customer, that carefully monitors and controls each modification.

The taxonomy presented by Auer et al. [233] describe characteristics of A/B experiments and A/B experimentation platforms focusing on a specific experiment iteration. In contrast, the activities and practices we discuss are presented at a higher level of abstraction. For example, after a team in the case company decides to run a business-driven controlled experiment with complete randomization, with feature toggles in a gradual rollout, it is still necessary to decide on the specific characteristics presented on the taxonomy (variant id, duration, guardrail and success metrics etc.). It is also worth noting that experimentation can be run and conducted ad-hoc, i.e., without an experimentation platform [32]. While Ericsson has a tailored platform and feature for conducting experiments in the first level (the user-level), ad-hoc experiments are suitable for the second level, which requires a joint experimental design with the customer since full randomization is often not possible. The release might involve different techniques depending on the maturity of the feature being experimented with.

11.7.3 RQ3: Can the HURRIER process can be used to drive CE in mission-critical B2B systems at Ericsson?

By observing the CE practices of different teams, we observed the key activities that compose the experimentation process at Ericsson. The empirical data, the set of practices, the concrete examples, documentation, and feature plans led to the development of the HURRIER process model. The HURRIER process represents a superset of the activities that are performed by these different teams. A subset of the activities in the HURRIER framework was used in each of the four types of experiments discussed in section 11.5. The requirements and the availability of tools (such as simulators and test rigs) in both the R&D organization and in the customers determine which set of activities are instantiated.

The key aspects when deciding upon the activities and instantiating the process are the time-length of the activity, the value it delivers when verifying the system. Deployment activities, such as network optimization depend heavily upon the existing conditions of the operator's network. In those cases, extensive laboratory testing and simulation activities deliver little value compared to field experiments. The internal validation is kept to a minimum. The software is validated only in terms of quality, guaranteeing that it does not influence or deteriorate other features.

On the other hand, the development of mission or even safety-critical features requires more extensive and lengthy internal validation, including following specifications and legislation. In this case, simulations and laboratory evaluations play a major role in increasing confidence before the field deployment. However, simulations and laboratory evaluations should be kept to the minimum necessary to guarantee the safety and basic functionality of the feature. Field experiments present a fast way to evaluate the feature and increase coverage in the operating conditions, which is often not feasible, costly, and time-consuming to implement in laboratory conditions. Deploying the mission-critical feature in the field within maintenance hours where traffic is minimal is another way to minimize the risk when the implementation is deployed in the field for the first time. Therefore, it is not only that the feature is deployed to a small subset of the network but also done at different times to reduce the risk further.

The feature should be implemented incrementally, so its value can be evaluated faster. Any evaluation of the delivered value should be left for the field experiments. Suppose the feature's minimum functionality does not deliver the expected value. In that case, the feature can be abandoned without the need to go through all the extensive work of developing, verifying, and even certifying the full feature.

The HURRIER process differs from the already existing experimentation process regarding the level of granularity and type of the activities. The HURRIER adds and reinforces specific activities for deploying software that needs to have high-quality assurance and in the B2B context. However, the proposed HURRIER process does not contradict existing experimentation processes since it is built on top of those models. For example, the execution step of an experiment iteration in the RIGHT model is broken down into several activities such as customer laboratory evaluation, passive launch, restricted launch, and one-customer gradual rollout. Although necessary for the context of mission-critical systems and B2B context, these activities might not be relevant for all startup experiments or web-facing companies that usually have lower risks involved in the deployment and are more in control of the data collection and deployment strategy than the customers. The HYPEX model reinforces the iterative process to increase the value of the delivered software. This is commonly seen in business-driven experiments. While the HURRIER process supports this, it also supports customer support experiments involving continuous iterations to add value.

The HURRIER process emphasizes that to continuously deliver high-quality and validated solutions, two aspects are required. The first is the presence of CI and CD. The second is the exposure of the system to live context and collaboration with customers to obtain feedback. These two aspects provide faster and more valuable feedback than focusing only on in-house validation and predefined testing scenarios.

11.7.4 RQ4: What are the current CE challenges and opportunities in mission-critical B2B systems observed at Ericsson?

Running CE in the B2B domain and mission-critical features presents many opportunities and challenges for both the company and the customers.

One of the challenges in CE at Ericsson is that any new deployment requires explicit approval and consent of the customers. For that, the customer needs to understand the need for the deployment, have a clear vision of how it can impact the system, and the potential benefits. However, successful experiments and transparency between the R&D company and the customer can help to build a trust relationship. A high trust relationship can facilitate running field experiments and evaluating new ideas for which the direct benefit is not yet clear for the customer. CE in the B2B domain also requires close collaboration between companies and customers since the field experiments are run in the customer premises. The collaboration happens from the pre-study to the gradual rollout phase.

Depending on the level of interaction, trust between the R&D team and the customer, and the level of interest in the feature, customers can help shape the development activities and even steer the development of the feature. While this can bring benefits, it also creates some challenges when adapting the feature for general availability, as seen in interview B section 11.6. For instance, customers that have steered the feature since its conception might have some sense of ownership and see some decrease in value when the feature is made to work in conditions beyond their network.

CE practices allow higher coverage and confidence in the development solution in mission-critical features, reducing the number of problem reports after general availability, and minimizing the costs of developing and maintaining an extensive simulation and laboratory solution. However, the responsibility for managing the risks of a field experiment in the B2B context relies on the customers, which requires the company to have a close collaboration and build a high-trust relationship.

CE is not seen only as positive for the case company but also the customers. For the company, CE helps to reduce the time-to-market as features are continuously validated with field data in terms of value and functionality. Other perceived advantages are: (1) the reduced time for a problem report to be addressed by the development team; (2) an increased sense of accomplishment, as the functionality is seen in the field in a shorter period; and (3) the higher sense of autonomy that comes with a higher trust from the customers, as the development teams can propose and test new ideas in the field faster. For the customer, CE has allowed them to test new functionalities, evaluate ideas and understand the impact of those ahead of the release, give them a competitive advantage on their corresponding markets.

11.8 Conclusion

In collaboration with Ericsson, we conducted a case study research method to understand how CE is used in B2B mission-critical systems. The case study investigated the continuous experimentation practices of multiple teams in different locations and countries working on the 4G mission-critical product.

This paper provides four main contributions. First, we classify the different types of experiments, practices, and techniques used in B2B mission-critical systems. Then, these techniques are discussed in the context of existing research in other domains. We identified four types of experiments that are conducted, business-driven, regression-driven, optimization, and customer support experiments. Second, we present and analyze each of the four types of experiments with four examples. The examples show the general experimentation process followed by the team as well as the usage of the different practices and techniques. Third, based on the empirical data we inductively derived the HURRIER process (**H**igh val**U**ed softwa**R**e th**R**ough cont**I**nuous Expe**R**imentation), a process that combines different experimentation techniques and practices to deliver high-quality solutions that the customers value. At Ericsson, subsets of the HURRIER process helped the R&D organization to validate feature functionality, increase coverage, identify and trace stochastic faults and increase the confidence in the developed solutions much faster than without the field experiments. Finally, we discuss the challenges, opportunities, and lessons learned from applying CE and the HURRIER process in B2B mission-critical systems.

CE has the potential to deliver value and higher-quality solutions. However, in the B2B domain, this can only be achieved with high trust and close collaboration with the customers. Therefore, we plan to introduce the HURRIER process and evaluate it in industries in different domains in future work.

Acknowledgments

This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and by the Software Center. The authors would also like to express their gratitude for all the support provided by Ericsson.

Chapter 12

Discussion

We outline in this section how this thesis addresses the two objectives proposed in section 3.1 and discuss each research question in detail.

12.1 Objective 1

Analyze how software companies can support and optimize their systems with automated experiments.

In chapter 5, we have shown a software architecture capable of supporting automated experiments. Architecture qualities considerations such as external adaptation, performance reflection, and an explicit learning component were central for the industrial validation of this architecture in collaboration with Ericsson [30] and Sony Mobile [29]. The software industry aims to integrate experimentation with their existing systems and technologies, and external adaption through experiments is fundamental to keep control of the software complexity and scale experimentation. Chapter 6, discussed the specific use of the proposed architecture to optimize a Long-Term Evolution radio base station. The complexity of the software validation, the use of custom and operator-defined metrics, and other factors such as memory constraints demand an external adaption mechanism for online optimization.

With reinforcement learning algorithms (such as multi-armed bandits) or another heuristic optimization algorithm (such as an evolutionary algorithm), online optimization is often associated with increased technical debt [104]. It often requires explicit learning mechanisms to update the algorithms without losing already learned solutions. Chapter 8 discussed other implementation considerations, especially in the context of multi-armed bandits. For instance, its use in long-term experiments requires thinking about user consistency, time variation (non-stationary bandits), and the ability to perform a hot start with a learning component when the algorithms are updated. Chapter 8 also provides an extensive discussion of the misuse of multi-armed bandits in online experimentation. While there are many valid and successful use cases, the indiscriminate use of multi-armed bandits as a substitute for randomized field experiments can lead to many undesirable consequences, such as reducing the power of the analysis, increased error type I, or lack of tools to detect other experimentation problems.

One area of active development in online optimization is the development of algorithms for searching the parameter space. When formulating the online optimization problem, as a search-based black-box problem, [29, 51, 53], practitioners are presented with a several algorithm families without any systematic procedure to guide their choice. This situation is complicated by the lack of statistical comparison between families of algorithms and by the low quality of the statistical analysis. To address the specific problem of the statistical analysis, we propose on chapter 9 several Bayesian statistical models can be used to provide more interpretable analysis and move beyond the problems associated with null hypothesis testing in the algorithm comparison domain [86–89, 91].

Chapter 7 takes a more holistic view of the automated optimization problem and analyzes the field experimentation process from the Experimentation and Analysis team at Microsoft. We observe that multiple activities conducted within a single experiment iteration can and should be automated to reduce the costs of each iteration, allowing the organization to scale its experimentation. In specific, automation is mostly done in the experiment execution and analysis phases. For instance, pre-quality checks, alerting, post quality checks, ramp-up, randomization, analysis etc. However, hypothesis generation and prioritization, decisions, metric design and the feature coding are manually conducted and there is seen as low return investment areas for automated experiments.

We discuss below the specific research questions regarding the first objective and proposed in chapter 3.

12.1.1 RQ1: What are the characteristics of an automated experimentation architecture?

This research question is discussed in chapter 5. We focus on identifying software architectural qualities from existing systems based on a literature review in terms of the characteristics.

RQ1a: What architectural software qualities support automated experimentation? We identified the following software architectural qualities in the research literature. These qualities are listed in terms of relative importance for the development of an automated experimentation system. External adaptation control, data collection as an integral part of the architecture, performance reflection, explicit representation of the learning component, decentralized adaptation, and knowledge exchange.

RQ1b: What are the existing software architectures that support these qualities? We have identified 21 architectures that are based on 7 different approaches. These architectures are listed in the table 5.1. From those architectures, the FUSION [103] implements the largest number of the qualities according to the relative importance. The FUSION architecture served as inspiration for developing the experimentation framework proposed in chapter 5 and also used and discussed in publications [29, 30, 66].

12.1.2 RQ2: How can we utilize automated experimentation to optimize an existing software-intensive systems?

In chapter 6 we investigate the use of online optimization experiments in the telecommunication domain. The optimization of a LTE radio base station requires tuning a high number of calibration parameters, capturing operator business goals and KPIs that might change. Additionally, since these radio base stations are online and calibration can affect the final user, regret minimization is also required.

By utilizing an implementation of the architecture framework proposed in chapter 5 and the extension of the algorithm proposed in [29] (the mLG-HOO algorithm), we demonstrated the viability of utilizing an external experimentation system capable of regret minimization while calibrating multiple parameters. The proposed approach optimized a common metric (random access success ratio) up to 46.3 % compared to the default parameters in a testbed.

12.1.3 RQ3: What are the main components to run trustworthy online controlled experiments?

In chapter 7, we investigate the experimentation process used in the Analysis and Experimentation team at Microsoft. The studied process focuses on how to scale experimentation with a trustworthy foundation. This process involves two main components, the experimentation activities and the evolution of the metrics. While the experimentation activities allow non-experts to conduct trustworthy experiments effectively, the role and evolution of metrics ensure that the experiments do not suffer from construct validity problems and are aligned with higher-level business goals.

RQ3a: What are the set of activities that are conducted in each experiment iteration? The identified experimentation activity model divides the experimentation process into three stages: development, execution, and analysis. While the development and analysis stages are mostly human-driven (developers, experimentation, and product owners), the execution stage involves a high number of automated activities, from computing quality check, ramp-up, randomization, computing metrics movements, and generating alerts that require human intervention.

RQ3b: What is the role and lifecycle of metrics in the evolution of experiments? Metrics have an important role in the evolution and impact of experiments in software development. The use of company and team-wide metrics capture business long-term goals and are often subject to constant revision and revaluation. While not directly used for decisions, an experiment also requires several additional metrics for verifying data quality, business constraints and supporting local and lower-level diagnostics. These metrics play an important role in ensuring that an experiment is trustworthy. Experimentation metrics follow a specific lifecycle: creation, evolution, maturity and phase-out. New guardrails and overall evaluation metrics creation are often created in parallel to existing ones. Due to the higher risk in influencing decisions, these metrics are first evaluated using historical data, to understand their impact in well-understood experiments and later moved to the online evaluation. In the maturity phase, metrics are updated periodically to ensure that definition, computational time, sensitivity, and synchronization are correct and improved and that they still capture more subjective business goals.

12.1.4 RQ4: How are multi-armed bandit (MAB) algorithms used in online field experiments?

Chapter 8 explores how MAB algorithms are used in practice in online field experiments. Practitioners often discuss MAB as an extension and a better alternative to A/B testing as it minimizes potential regret losses and have a high number of extensions. However, in practical implementations, MAB is also associated with a number of pitfalls and restrictions that should be addressed to lead to valid interpretations and results. The following sub-research questions address these issues

RQ4a: What are the restrictions and pitfalls associated with MAB algorithms applied to software online experiments? In the multiple case study presented in Chapter 8, we identified several restrictions and pitfalls associated with MAB experiments and potential strategies to address them. We identified nine reasons that lead to three main restrictions and pitfalls. For example, näive implementations, violation of assumptions, and MAB use in exploration problems are associated with an increase in type I error. The lack of sample ratio mismatch, the increased regret in the presence of Simpson Paradox and the increased complexity in ramp-up are restrictions when detecting and diagnosing experimentation problems. Finally, adaptive allocation based on a single metric, user consistency, and communicating experiment results increases the complexity of the design compared to traditional A/B experiments. These points are summarized in table 8.2.

RQ4b: What are the decisions involved in the design of MAB-based online experiment? The identified restrictions and pitfalls requires that practitioners evaluate different aspects before choosing MAB-based experiment. In chapter 8, we identified five different aspects to consider before choosing a MAB experiment. The first aspect to consider is the goal of the experiment (learning, innovation, or optimization). The second is the associated costs and risks of making type I and type II errors. The third is how well understood is the problem and well it matches the assumptions of the potential algorithms, The fourth aspect refers to the chosen decision metrics and if they can be grouped in a single metric. Finally, the last aspect refers to the duration of the experiment in terms of time, if it is limited time or long-term experiments, and how it affects user consistency for recurring users. These aspects as well as the different decisions are summarized in table 8.3.

12.1.5 RQ5: How can we improve the conclusion validity on the analysis of optimization algorithms with benchmark functions in different domain-specific research questions?

The success of an automated experimentation system is highly dependant on the availability of optimization algorithms that can be effectively used in industrial and practical applications. With a large number of optimization algorithms available, practitioners rely on research results from algorithm comparison papers and competitions with artificial benchmark functions. One example is the selection and evaluation of Google's Vizier algorithm used for automated experimentation [247]. The authors validate their choice of experimentation algorithm based on the 24 Black-Box Optimization Benchmarking (BBOB) functions [62]. However, algorithms are often poorly compared are subjected to many conclusion validity considerations (as discussed in the data analysis section 3.4.2).

In chapter 9, we provide five Bayesian statistical models to answer common questions regarding the analysis of optimization algorithms as well as addressing many of these conclusion validity threats that arise from misuse of frequentist statistics. All the discussed models include random-effects terms to model the intra-correlation introduced by repeated measures of the benchmark functions. The proposed models are used to evaluate the probability of solving a problem (a binomial model); to evaluate the relative improvement (a linear regression model); to rank algorithms (a Bradley-Terry model), to estimate the number of evaluations to converge to a solution (a Cox's regression model); and to compare multiple algorithms for CPU time (robust regression model). All the data and models are reproducible and available in the online appendix at: `https://davidissamattos.github.io/statscomp/`.

With the proposed models, researchers can ask questions that go beyond differentiating statistical significance between algorithms and evaluate the different impact benchmark functions can have on the obtained results [248]. We expect these models will increase the validity of the conclusions obtained in algorithm comparison and algorithm competition.

12.2 Objective 2

Analyze how non web-facing companies can adopt continuous experimentation as part of their development process.

While experimentation is rather well understood in the web-facing domain, there are many specific challenges to embedded systems and business-to-business systems. These challenges are leading companies to adopt different practices for introducing and adopting experimentation as an integral part of their development process.

In chapter 10, we have explored some of the perceived challenges and proposed solutions in the embedded systems domain. It is interesting to observe, that while many technical and business challenges are unique to these domains, organization challenges share many common aspects with web-facing companies.

In chapter 11, we identified many experimentation techniques that are used and address specific challenges of business-to-business mission-critical systems. In addition, we have developed an experimentation process for these systems. This process emphasizes that field with experiments can not only validate business and development ideas but can also be used to increase software quality, tune systems and identify stochastic failures.

In the context of this second objective, below we the specific research questions proposed in chapter 3.

12.2.1 RQ6: How can the embedded systems industry adopt continuous experimentation in their development process?

To adopt experimentation, the embedded systems industry needs to better understand the challenges that prevent and limits the use of experimentation in the development process, and what are the existing and potential solutions to address these challenges. This two-part process is investigated in detail in the following sub-research questions.

RQ6a: What are the recognized challenges towards continuous experimentation faced by the embedded systems industry? Chapter 10 identifies twelve different challenges categorized in three perspectives, technical, business, and organizational.

From the technical perspective, we identified four main challenges:

[a] Lack of over-the-air (OTA) updates and data collection. Embedded systems companies commonly develop and commercialize embedded systems without integrating OTA data collection and updates. This prevents the development organization to perform iterative experiments and evolve the product after launch.

[b] Lack of experimentation tools that integrate with their existing tooling. The existing tools for conducting experiments in software systems are highly tailored for the web-facing domain. Companies often sell software development kits based on web and mobile technologies (e.g. Javascript, Android, or iOS), where common metrics are automatically collected and ready to be used. However, for other domains, these tools and metrics are not suitable and the lack of existing alternatives hinders the adoption of experimentation.

[c] Expensive testing environments. For companies working in the mission or safety-critical domain, software must be validated in a wide range of testing scenarios, which has a time-associated cost to the development. While the treatment software is not experimental software (as in terms of validation), iterative experiments increase the burden and the costs in testing environments especially if manual validation is mandatory.

[d] Experimentation constraints in real-time and safety-critical systems. The presence of an experimentation system collecting and transmitting data might be subjected to additional certification and regulatory constraints.

Additionally, it can substantially decrease the performance of some systems to an unacceptable level for the product application.

From the business perspective, we identified five main challenges:

[a] Metric definition and validation. Due to the long tradition of conducting experiments, the web-facing domain has already identified many metrics that serve as a proxy for business decisions. Due to the lack of experimentation expertise in the embedded systems domain, the existing metrics are closer to system logs than connected to business goals. Metrics should be evolved and validated, as discussed in chapter 7, to capture business assumptions and goals.

[b] Privacy restrictions. Current regulations on collecting and sharing data add an additional layer of complexity in planning and designing systems that are able to conduct experiments. The granularity of the collected data and the extent to which it can identify an individual affect the design and type of experiments.

[c] Lack of sharing data in B2B solutions. With exceptions [249], most web-facing software is designed by the company in a business-to-consumer setting. The company has controlled of the software and the customer and user are the same. However, it is common for embedded systems companies to have hardware and software components from third parties. In addition, in business-to-business contexts, ownership of the data is distributed between multiple stakeholders. Without agreements, the lack of sharing data can prevent companies from experimenting with parts of the software and compute relevant metrics.

[d] Lack of insights into the collected data. Even though most companies have data collection and instrumentation in place they were designed mainly for troubleshooting purposes and not for understanding customer behavior. Combining with the challenge on metric definition and validation, troubleshooting data lack insights on how to increase the delivered value of the software and support business decisions.

[e] Long release cycles. The lack of OTA, the expensive testing environments, and the number of software variants [31] (which hinders continuous integration and deployment to a large extent) results in longer release cycles. Long release cycles prevent results from experiments from being applied, unsuccessful experiments from being stopped, existing metrics from being validated, and new metrics from being introduced.

From the organizational perspective, we identified three main challenges:

[a] Managing multiple stakeholders during the experiments. While managing multiple stakeholders is a challenge also in the web-facing domain, in the embedded systems experiment owners also need to coordinate collaboration with systems architects, electrical and mechanical engineers.

[b] HiPPO (Highest Paid Person Opinion). Development decisions are imposed top-down without considering existing evidence. These decisions come in the form of requirements and specifications and are often hard to modify in Waterfall and V-model based development processes.

[c] Repetitive tuning experiments repetitive experiments with highly qualified engineers. In one of the case companies, search-based optimization experiments were conducted manually by highly qualified engineers. Together with the lack of integrated tools this significantly increases the cost of each experiment iteration.

The data from chapter 10 was collected in 2017. Since then, we have researched with other companies and identified the set of additional challenges. While specific challenges to the automotive domain are discussed in [31, 42] below we list two additional general challenges.

- Number of variants. Most software companies that run experimentation often design experiments that are hardware agnostic, i.e. changes in the underlying hardware should not influence the experiment outcome. Nevertheless, they still compute metrics specific to the hardware environment to identify faults that might arise. However, the embedded systems domain utilizes a wide range of specialized software for each hardware configuration that depends on the region, supplier, etc. In many cases, such as the automotive [31] and telecommunication domain [69], the high number of software variants creates a barrier to compare nearly equivalent systems. In practice, results are difficult to generalize to other variants and the number of effective samples for an experiment is also reduced.

- Available sample size. In connection to the number of variants and the B2B domain, the embedded systems domain does not have the large user base commonly seen in online field experiments. In these cases, it is required to conduct experiments with higher control of external factors (i.e. controlled experiments), compensate from known sources of variance (to increase metric sensitivity), balance the experimental groups [250] during design (to reduce sample imbalance), and alternative designs (such as paired and crossover).

RQ6b: What are the recommended strategies to facilitate the use of continuous experimentation in the embedded systems domain? The identified solutions and potential strategies can be categorized as development process changes, data handling changes, and architectural changes.

In terms of development process change strategies, continuous experimentation requires adopting agile software development to embrace the changes and adjust to hypotheses that fail to deliver value to customers. Ethical guidelines are also required and should be incorporated in the development process to address both privacy concerns and mitigate experiments that can be prejudicial or discriminatory to certain users. Both the EDAX model [26] as well as the HURRIER model, discussed in chapter 11, provide a framework to address the use of experiments in expensive testing environments and in mission-critical systems.

Data handling strategies address challenges related to how to generate insights from the collected data while complying with exiting data regulations. In terms of data sharing, one solution proposed by companies is the creation of ecosystems that facilitate collecting data and sharing it among partners in different organizations. Companies are also investing and incorporating data scientists in teams to exploit the potential of machine learning applications and increase the understanding of the existing and how to evolve them.

Finally, in terms of architectural change strategies, companies can benefit from the architectural decoupling of the hardware design and the software design. While some of the companies have used this strategy, this process should be conducted with experimentation and data sharing in mind. It can potentially lead to the challenge of the lack of sharing user data between different components of the same application. The ability to perform OTA updates and data collection infrastructure is seen as mandatory for most companies. However, it is not clear for some systems and applications if the potential benefits can be overcome the incurring costs of connectivity.

12.2.2 RQ7: How experimentation can be conducted in mission-critical business-to-business systems?

This general research question and the relevant sub-research questions were discussed in chapter 11.

RQ7a: What are the types of experiments that can be conducted and that are relevant in mission-critical B2B systems? In the case study provided in chapter 11, we have identified four types of experiments that are conducted and relevant to the development of mission-critical B2B systems. The types of experiments are:

[a] Business-driven experiments, which aim at validating business ideas and the impact and value it delivers to customers.

[b] Regression-driven experiments, which aim at understanding functional aspects that were/cannot be validated in internal testing. These experiments are intended to increase the quality aspects of the software.

[c] Tuning and optimization experiments, these experiments aim at identifying configuration aspects that can be tuned to increase the value delivered by existing features, without a re-validation or deployment. This type of experiment has been the focus of most automated experimentation research such as the empirical evaluation described in chapter 6.

[d] Customer support experiments aim at identifying stochastic faults and degradation that occurred in the customer operating environment. These are often conducted by customer support in collaboration with the customers.

RQ7b: What are the current continuous experimentation practices used in mission-critical B2B systems? In chapter 11, we identified four groups of practices and techniques in experimentation: experiment design and analysis, variation assignment, implementation, and release.

While randomized experiments are one of the most used techniques due to their simplicity, and higher internal validity (compared to the other practices), embedded, mission-critical and B2B systems can have additional restrictions that prevent them to use randomized experiments. Besides randomized experiments, we identified crossover, multi-armed bandits, quasi-experiments, and optimization designs. In variation assignment, we identified manual assignment, complete randomization, and cluster-based randomization. In terms of implementation techniques, although, software versions and traffic routing are used techniques, they allow less flexibility in the design and scaling than feature toggles. Canary release, passive deployment, gradual rollout and ring-based releases are common techniques used in both the web and non-web domain. In addition to these, we also identified the use of time-window releases, which restricts the experimentation and release period in lower-risk time windows.

RQ7c: What process can be used to drive CE in mission-critical B2B systems? Chapter 11 presents and describes the HURRIER process model. This model represents a superset of the activities that a team can use to run experiments in collaboration with customers. The HURRIER process is built on top of the models discussed in section 2.5 and can be used for all four types of experiments (business, regression, optimization, and customer support experiments).

The HURRIER model emphasizes the integration of CI, CD, customer feedback, and collaboration, and early exposure of the system to live contexts are necessary to deliver high quality and validation solutions continuously.

RQ7d: What are the current CE challenges and opportunities in mission-critical B2B systems? Among the challenges presented in chapter 10, we identified 11 additional challenges and opportunities for continuous experimentation in mission-critical B2B systems. The major challenge is to build a high-trust relationship with the customers. For CE to be successfully implemented, there is a need if a high degree of collaboration and transparency between the customers and R&D organization.

In terms of opportunities, CE allows the R&D organization to achieve higher coverage and confidence in the developed solution. However, managing the risks in the B2B context still falls in the customer's hands. This emphasizes again the need for a high-trust relationship. Despite the non-widespread use, CE is seen as positive by customers as it reduces the time-to-market of features that they understand and value from the early stages of development

Chapter 13

Conclusion

This thesis studies the topic of experimentation in software-intensive systems from two main objectives.

In the first objective, we analyzed how software companies can support and optimize their systems through automated through the perspectives of the software architecture, the algorithms for the experiment execution, and the experimentation process. We have presented a software architecture for automated experiments in chapter 5 and used this architecture in an industrial context for the automated optimization of a radio-base station in chapter 6. In chapter 7, we identified the different activities of an experimentation process and the lifecycle metrics and impact of metrics in the trustworthiness of an experiment iteration. In chapter 8, we identified common pitfalls and solutions in the adoption of multi-armed bandits in online experiments. MAB-based experiments are one of the base algorithms for automating experiments with regret bounds. Chapter 9 presents statistical models for the validation of optimization algorithms. These models are intended to increase conclusion validity in benchmarking algorithms used in automated experimentation.

In the second objective, we analyze how embedded systems companies can adopt continuous experimentation to validate and deliver value to their customers continuously. This is investigated from the perspectives of the software development process and focuses on the experimentation aspects that are distinct from web-facing companies. In chapter 10, we explore common challenges and potential strategies for the adoption of experimentation in embedded systems. These challenges and solutions were identified both in industrial and research context. In chapter 11, we present different types of experiments and practices used in mission-critical systems. In this last chapter, we also introduce the HURRIER process. This process aims at combining quality assurance techniques common in mission-critical systems with experimentation practices to achieve higher quality, deliver value to customers and final users, identify stochastic faults and optimize systems based on field data.

In the remainder of this chapter, we present potential research and future work in the area of experimentation.

13.1 Future work on experimentation.

While we have contributed to many different aspects of experimentation, there are still many open problems. In this section, we highlight a few possible directions for research on experimentation.

13.1.1 Automating experiments.

While, in this thesis, we have mainly discussed the use of sequential optimization experiments (chapters 5, 6, 8 and 9) and parts of the experimentation execution that can be automated (chapter 7), there are multiple additional characteristics of automating experiments that can be further investigated.

13.1.1.1 Experimentation ontology

Drawing inspiration from Soldatova et al. in biological systems [251, 252], an automated experimentation system refers to a system that is able to carry out cycles of scientific controlled or field experiments. To conduct scientific experiments a system, the system should be able to follow a domain-specific instantiation of a ontology of scientific experiments. While there are no guidelines or ontology for experiments, as there are in other areas, in software systems the goals proposed by Soldatova et al. are applicable:

- to formalize the concepts involved in an experiment and identify the essential metadata for the experiment description and repeatability.
- to provide a controlled vocabulary for the user of the system.
- to organize the information and knowledge in different meta-levels so the system has the capability of updating the knowledge base, plan and execute multiple experiments and access results of the experiments.
- to design a database for the storage of experimental data and track the experiment execution.

While some of these goals have been applied in the design of experimentation systems [21,22,65,66,161], research has still not provide a systematic framework and an ontology for experiments.

13.1.1.2 Hypotheses generation

The current work on automated experimentation has focused on the generation of hypotheses in terms of parameters change. While this is valid and appropriated for optimization of existing features, it has not been applied for more general experiments. While, this hypotheses generation is one of the core challenges for fully automated experiments, it does not necessarily has a high return on investment for companies. The hypotheses for new features usually come from competition and from users [27] and might require a few development and experiment iterations to bring value to the product. The new iterations are complemented with qualitative feedback and increase the number of experimentation cycles. As discussed previously, most software companies are looking at automating their experimentation pipeline to reduce the costs of each experiment iteration in order to run more experiments simultaneous.

13.1.1.3 Counterfactual analysis

Currently, the R&D organization at companies can generate more hypotheses than they can run experiments. While there isn't a strong need for more hypotheses generation there is a need for hypotheses prioritization and better understanding of the potential impacts of a change before an experiment is conducted.

In this scenario, an emerging topic in research is the application of counterfactual analysis based on historical data. Counterfactual analysis aims to answer questions like: *"How would the system have performed if, when the data was collected, we have replaced changed M by change M' without incurring user reactions"* [110].

Analysis of counterfactual questions allows the R&D team to replay historical data of the users (the assumption that the user did not change its behavior due to the system modification) on a modification of the system and observe the potential impact. The results can be used to prioritize modifications that can lead to higher return-on-investment.

To be able to replay historical data, the R&D organization needs to have a model of how the user behaves with the system, such as a structure equation model [253] or a Bayesian network model [110]. Counterfactual analysis are currently used in the area of ad-placement and in search engines [110, 254–256], where companies already have good statistical click models [257], large amount of experiment and historical data.

Research on experimentation can take results and methods used in search engines and ad-placement to provide new tools for prioritization of experiments in different applications.

13.1.2 Small-sample experiments

Experimentation in software systems has started and flourish mostly on the web-domain for a number of reasons, such as the early adoption of an agile mindset, continuous integration, and data collection. However, one of the aspects that were decisive for its success, specially in large-scale companies is the number of users available for conducting field experiments.

However, for many companies, in particular embedded systems companies, the number of connected users is impractical and presents as a great barrier for the adoption of experimentation. In diverse populations, random samples for each experimental group can lead to random imbalance [161] and this problem is augmented in small samples. Random imbalance consists of experimental groups that are not comparable in respect to a set of metrics prior to conducting the experiment. While A/A tests can help identify random imbalance and lead to a new re-randomization, it essentially does not change the problem faced and can significantly increase the time to conduct experiments. For large user bases, online companies have relied in historical A/A tests to identify rash seeds that generates more balanced groups [161].

However, research on different fields have already provided a number of techniques to address group imbalance in experimental designs. These techniques are broadly classified as minimization techniques [258–261]. With minimization, experimental group allocation does not solely rely on chance but rather the groups are designed to reduce any difference in known or suspected covariates that can influence the outcome.

Among these techniques, we can divide minimization in two groups: pre-experiment group allocation and sequential allocation. Pre-experiment group allocation utilizes information from the experimental subjects, such as the known or suspected covariates that can influence the outcomes, and creates two pre-determined groups that minimizes the impact of the covariates [106]. Pre-experiment group allocation is useful when a number of subjects are already available and information on them is already collected.

Sequential allocation aims to achieve minimization as the subjects sequentially come to the experiment. Therefore, instead of utilizing information of the experimental subjects prior to the experiment to divide the groups, it utilizes this information to create the best minimization allocation dynamically. Sequential allocation is useful when we do not have information of which subjects will be part of our experiment and information and group allocation is conducted as new subjects participate [260].

Both minimization techniques can have a high impact on how experiments are conducted within embedded systems. We have been exploring how the Balance Match Weighted technique [106] can be used to generate pre-experiment balanced groups in automotive A/B testings. This technique has been suggested in our previous work [31] and is described in more detail in paper p [250].

13.1.3 Longitudinal experiments

Research on experimentation has mainly focused on how metrics in average are affected by treatments. By focusing on statistical testing, companies inherently assume that these metrics are invariant across time during to simplify the design and analysis process. However, such assumption can lead to overestimating the effects of treatments and which can lead to potential decision errors [38, 262]. To address problems that come from failing to meet this assumption, companies often consider experiments in the average of time periods know to affect the business, e.g. over a week to compensate for the difference of weekdays and weekends, or month to compensate the difference between when customers receive their salary.

However, despite the availability of statistical methods to address such time differences (such as panel data and time series analysis [263]), they are not commonly used in the context of experimentation. Future work on experimentation can benefit from addressing how to formulate dynamic effects problems and incorporate them into experimentation systems. Incorporating seasonality effects, trends and outliers in time events can improve the sensitivity of experimentation metrics, increase the understanding of time-related effects on the development and launch of new features.

13.1.4 Causal inference beyond experiments

Experimental designs are the goal of any organization aiming for causal software development, there are many situations and decision points where we cannot conduct experiments [264].

One of the basic assumptions for causality is the exchangeability assumption. The exchangeability assumption refers to the conditions in which conditional independence relationship between the treatments, potential outcomes and the possible confounding variables [264, 265]. While carefully designed experiments designs fulfill the exchangeability assumption, it is possible to achieve exchangeability and causal inference in observational studies.

Even in non experimental contexts, organizations can identify potential confounds and utilize statistical model to infer causality. We see a potential to use of methods such as differences-in-differences [266] and regression discontinuity [267] to improve causal software development.

Bibliography

[1] CRediT, "CRediT (Contributor Roles Taxonomy)." [Online]. Available: https://www.elsevier.com/authors/policies-and-guidelines/credit-author-statement

[2] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: survey and practical guide," *Data mining and knowledge discovery*, vol. 18, no. 1, pp. 140–181, 2009.

[3] A. Fabijan, H. H. Olsson, and J. Bosch, "Early value argumentation and prediction: An iterative approach to quantifying feature value," in *International Conference on Product-Focused Software Process Improvement.* Springer, 2015, pp. 16–23.

[4] R. Kohavi and S. Thomke, "The surprising power of online experiments," *Harvard Business Review*, vol. 95, no. 5, pp. 74–82, 2017.

[5] H. H. Olsson and J. Bosch, "The hypex model: from opinions to data-driven software development," in *Continuous software engineering.* Springer, 2014, pp. 155–164.

[6] A. Fabijan, H. H. Olsson, and J. Bosch, "Customer feedback and data collection techniques in software r&d: a literature review," in *International Conference of Software Business.* Springer, 2015, pp. 139–153.

[7] 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, 2017.

[8] ——, "Building blocks for continuous experimentation," in *Proceedings of the 1st international workshop on rapid continuous software engineering*, 2014, pp. 26–35.

[9] A. Fabijan, H. H. Olsson, and J. Bosch, "Time to say'good bye': Feature lifecycle," in *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).* IEEE, 2016, pp. 9–16.

[10] A. F. Payne, K. Storbacka, and P. Frow, "Managing the co-creation of value," *Journal of the academy of marketing science*, vol. 36, no. 1, pp. 83–96, 2008.

[11] J. Bosch, "Building products as innovation experiment systems," in *International Conference of Software Business.* Springer, 2012, pp. 27–39.

[12] P. Bosch-Sijtsema and J. Bosch, "User involvement throughout the innovation process in high-tech industries," *Journal of Product Innovation Management*, vol. 32, no. 5, pp. 793–807, 2015.

[13] R. Kohavi, A. Deng, B. Frasca, R. Longbotham, T. Walker, and Y. Xu, "Trustworthy online controlled experiments: Five puzzling outcomes explained," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 786–794.

[14] H. H. Olsson and J. Bosch, "Towards data-driven product development: A multiple case study on post-deployment data usage in software-intensive embedded systems," in *International Conference on Lean Enterprise Software and Systems.* Springer, 2013, pp. 152–164.

[15] ——, "Post-deployment data collection in software-intensive embedded products," in *International Conference of Software Business.* Springer, 2013, pp. 79–89.

[16] D. C. Montgomery, *Design and analysis of experiments.* John wiley & sons, 2017.

[17] F. Auer, R. Ros, L. Kaltenbrunner, P. Runeson, and M. Felderer, "Controlled experimentation in continuous experimentation: Knowledge and challenges," *Information and Software Technology*, p. 106551, 2021.

[18] D. Tang, A. Agarwal, D. O'Brien, and M. Meyer, "Overlapping experiment infrastructure: More, better, faster experimentation," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 17–26.

[19] E. Bakshy, D. Eckles, and M. S. Bernstein, "Designing and deploying online field experiments," in *Proceedings of the 23rd international conference on World wide web*, 2014, pp. 283–292.

[20] H. Gui, Y. Xu, A. Bhasin, and J. Han, "Network a/b testing: From sampling to estimation," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 399–409.

[21] R. Lopez Kaufman, J. Pitchforth, and L. Vermeer, "Democratizing online controlled experiments at booking. com," *arXiv e-prints*, pp. arXiv–1710, 2017.

[22] N. Diamantopoulos, J. Wong, D. I. Mattos, I. Gerostathopoulos, M. Wardrop, T. Mao, and C. McFarland, "Engineering for a science-centric experimentation platform," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*, 2020, pp. 191–200.

[23] Y. Xu, W. Duan, and S. Huang, "Sqr: balancing speed, quality and risk in online experiments." Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 895–904.

[24] F. Yang, A. Ramdas, K. Jamieson, and M. J. Wainwright, "A framework for multi-a (rmed)/b (andit) testing with online fdr control," *arXiv preprint arXiv:1706.05378*, 2017.

[25] A. Datta, M. C. Tschantz, and A. Datta, "Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination," *Proceedings on privacy enhancing technologies*, vol. 2015, no. 1, pp. 92–112, 2015.

[26] J. Bosch and H. H. Olsson, "Data-driven continuous evolution of smart systems," in *2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS).* IEEE, 2016, pp. 28–34.

[27] D. I. Mattos, P. Dmitriev, A. Fabijan, J. Bosch, and H. H. Olsson, "An activity and metric model for online controlled experiments," in *International Conference on Product-Focused Software Process Improvement.* Springer, 2018, pp. 182–198.

[28] D. I. Mattos, J. Bosch, and H. H. Olsson, "Challenges and strategies for undertaking continuous experimentation to embedded systems: Industry and research perspectives." 19th International Conference on Agile Software Development, 2018.

[29] D. I. Mattos, E. Mårtensson, J. Bosch, and H. H. Olsson, "Optimization experiments in the continuous space," in *International Symposium on Search Based Software Engineering.* Springer, 2018, pp. 293–308.

[30] D. I. Mattos, J. Bosch, H. H. Olsson, A. Dakkak, and K. Bergh, "Automated optimization of software parameters in a long term evolution radio base station," in *2019 IEEE International Systems Conference (SysCon).* IEEE, 2019, pp. 1–8.

[31] D. I. Mattos, J. Bosch, H. H. Olsson, A. M. Korshani, and J. Lantz, "Automotive a/b testing: Challenges and lessons learned from practice." 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2020, pp. 101–109.

[32] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The evolution of continuous experimentation in software product development: from data to a data-driven organization at scale," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE).* IEEE, 2017, pp. 770–780.

[33] W. R. Shadish, T. D. Cook, D. T. Campbell *et al.*, *Experimental and quasi-experimental designs for generalized causal inference/William R. Shedish, Thomas D. Cook, Donald T. Campbell.* Boston: Houghton Mifflin,, 2002.

[34] D. I. Sjøberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N.-K. Liborg, and A. C. Rekdal, "A survey of controlled experiments in software engineering," *IEEE transactions on software engineering*, vol. 31, no. 9, pp. 733–753, 2005.

[35] K.-J. Stol and B. Fitzgerald, "The ABC of software engineering research," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 27, no. 3, pp. 1–51, 2018.

[36] Y. Xu, N. Chen, A. Fernandez, O. Sinno, and A. Bhasin, "From infrastructure to culture: A/b testing challenges in large scale social networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 2227–2236.

[37] Y. Xu and N. Chen, "Evaluating mobile apps with a/b and quasi a/b tests." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 313–322.

[38] D. I. Mattos, J. Bosch, and H. H. Olsson, "Multi-armed bandits in the wild: pitfalls and strategies in online experiments," *Information and Software Technology*, vol. 113, pp. 68–81, 2019.

[39] S. Soligon, M. Lixandrão, T. Biazon, V. Angleri, H. Roschel, and C. Libardi, "Lower occlusion pressure during resistance exercise with blood-flow restriction promotes lower pain and perception of exercise compared to higher occlusion pressure when the total training volume is equalized," *Physiology international*, vol. 105, no. 3, pp. 276–284, 2018.

[40] S. G. Yaman, M. Munezero, J. Münch, F. Fagerholm, O. Syd, M. Aaltola, C. Palmu, and T. Männistö, "Introducing continuous experimentation in large software-intensive product and service organisations," *Journal of Systems and Software*, vol. 133, pp. 195–211, 2017.

[41] G. Schermann, J. Cito, P. Leitner, U. Zdun, and H. C. Gall, "We're doing it live: A multi-method empirical study on continuous experimentation," *Information and Software Technology*, vol. 99, pp. 41–57, 2018.

[42] F. Giaimo and C. Berger, "Continuous experimentation for automotive software on the example of a heavy commercial vehicle in daily operation," *arXiv preprint arXiv:2003.03799*, 2020.

[43] 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*, 2017, pp. 1427–1436.

[44] A. Deng and X. Shi, "Data-driven metric development for online controlled experiments: Seven lessons learned," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 77–86.

[45] K. Kevic, B. Murphy, L. Williams, and J. Beckmann, "Characterizing experimentation in continuous deployment: a case study on bing," in

2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP). IEEE, 2017, pp. 123–132.

[46] E. Ries, *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses.* Crown Business Publishing, 2011.

[47] J. Bosch, H. H. Olsson, J. Björk, and J. Ljungblad, "The early stage software startup development model: a framework for operationalizing lean principles in software startups," in *International Conference on Lean Enterprise Software and Systems.* Springer, 2013, pp. 1–15.

[48] H. H. Olsson and J. Bosch, "Towards continuous customer validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation," in *International Conference of Software Business.* Springer, 2015, pp. 154–166.

[49] I. Gerostathopoulos, C. Prehofer, and T. Bures, "Adapting a system with noisy outputs with statistical guarantees," in *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, 2018, pp. 58–68.

[50] I. Gerostathopoulos, C. Prehofer, L. Bulej, T. Bureš, V. Horkỳ, and P. Tuma, "Cost-aware stage-based experimentation: challenges and emerging results," in *2018 IEEE International Conference on Software Architecture Companion (ICSA-C).* IEEE, 2018, pp. 72–75.

[51] G. Tamburrelli and A. Margara, "Towards automated a/b testing." International Symposium on Search Based Software Engineering, 2014, pp. 184–198.

[52] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "X-armed bandits." *Journal of Machine Learning Research*, vol. 12, no. 5, 2011.

[53] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[54] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr., "NiaPy: Python microframework for building nature-inspired algorithms," *Journal of Open Source Software*, vol. 3, 2018.

[55] N. Hansen, "The CMA evolution strategy: a comparing review," in *Towards a New Evolutionary Computation.* Springer, 2006, pp. 75–102.

[56] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions," *SIAM Journal on Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[57] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," 2013.

[58] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1437–1446.

[59] ML4AAD Group of the University of Freiburg, "HpBandSter's package." [Online]. Available: https://automl.github.io/HpBandSter/build/html/index.html

[60] AutoML.org, "SMAC." [Online]. Available: https://www.automl.org/automated-algorithm-design/algorithm-configuration/smac/

[61] J. Rapin and O. Teytaud, "Nevergrad - A gradient-free optimization platform," https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[62] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," *Optimization Methods and Software*, pp. 1–31, 2020.

[63] T. Bartz-Beielstein, C. Doerr, J. Bossek, S. Chandrasekaran, T. Eftimov, A. Fischbach, P. Kerschke, M. Lopez-Ibanez, K. M. Malan, J. H. Moore *et al.*, "Benchmarking in optimization: Best practice and open issues," *arXiv preprint arXiv:2007.03488*, 2020.

[64] J. Bosch and U. Eklund, "Eternal embedded software: Towards innovation experiment systems," in *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. Springer, 2012, pp. 19–31.

[65] D. I. Mattos, J. Bosch, and H. H. Olsson, "Your system gets better every day you use it: towards automated continuous experimentation," in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2017, pp. 256–265.

[66] ——, "More for less: automated experimentation in software-intensive systems," in *International Conference on Product-Focused Software Process Improvement*. Springer, 2017, pp. 146–161.

[67] 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, 2017, pp. 203–210.

[68] 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, 2019, pp. 126–130.

[69] D. I. Mattos, A. Dakkak, J. Bosch, and H. H. Olsson, "Experimentation for business-to-business mission-critical systems: A case study." Proceedings of the International Conference on Software and System Processes, 2020, pp. 95–104.

[70] E. Lindgren and J. Münch, "Raising the odds of success: the current state of experimentation in product development," *Information and Software Technology*, vol. 77, pp. 80–91, 2016.

[71] O. Rissanen and J. Münch, "Continuous experimentation in the b2b domain: a case study," in *2015 IEEE/ACM 2nd International Workshop on Rapid Continuous Software Engineering*. IEEE, 2015, pp. 12–18.

[72] 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, 2017, pp. 18–26.

[73] R. Kohavi, A. Deng, R. Longbotham, and Y. Xu, "Seven rules of thumb for web site experimenters," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1857–1866.

[74] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research." Springer, 2008, pp. 285–311.

[75] J. A. Maxwell, *Qualitative research design: An interactive approach*. Sage publications, 2012, vol. 41.

[76] G. Walsham, "Interpretive case studies in is research: nature and method," *European Journal of information systems*, vol. 4, no. 2, pp. 74–81, 1995.

[77] 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, 2009.

[78] C. Wohlin, "Case study research in software engineering—it is a case, and it is a study, but is it a case study?" *Information and Software Technology*, vol. 133, p. 106514, 2021.

[79] R. K. Yin, *Case study research and applications: Design and methods*. Sage publications, 2017.

[80] C. Robson and K. McCartan, *Real world research*. John Wiley & Sons, 2016.

[81] T. C. Lethbridge, S. E. Sim, and J. Singer, "Studying software engineers: Data collection techniques for software field studies," *Empirical software engineering*, vol. 10, no. 3, pp. 311–341, 2005.

[82] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[83] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, no. 2004, pp. 1–26, 2004.

[84] M. Ely, M. Anzul, R. Vinz, and M. Downing, *On writing qualitative research: Living by words*. Psychology Press, 1997, no. 12.

[85] J. Wakefield, *Bayesian and frequentist regression methods*. Springer Science & Business Media, 2013.

[86] R. L. Wasserstein, A. L. Schirm, and N. A. Lazar, "Moving to a world beyond "p <0.05"," *The American Statistician*, vol. 73, no. sup1, pp. 1–19, 2019.

[87] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2653–2688, 2017.

[88] J. K. Kruschke and T. M. Liddell, "Bayesian data analysis for newcomers," *Psychonomic Bulletin & Review*, vol. 25, no. 1, pp. 155–177, Feb 2018.

[89] J. Gill, "The insignificance of null hypothesis significance testing," *Political Research Quarterly*, vol. 52, no. 3, pp. 647–674, 1999.

[90] H. Haller and S. Krauss, "Misinterpretations of significance: A problem students share with their teachers," *Methods of Psychological Research*, vol. 7, no. 1, pp. 1–20, 2002.

[91] R. L. Wasserstein and N. A. Lazar, "The ASA Statement on p-values: Context, Process, and Purpose," *The American Statistician*, vol. 70, no. 2, pp. 129–133, 2016.

[92] G. Cumming, *Understanding the new statistics: Effect sizes, confidence intervals, and meta-analysis.* Routledge, 2013.

[93] C. A. Furia, R. Feldt, and R. Torkar, "Bayesian Data Analysis in Empirical Software Engineering Research," *IEEE Transactions on Software Engineering*, 2019.

[94] R. Wilcox, *Modern statistics for the social and behavioral sciences: A practical introduction.* CRC press, 2011.

[95] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis.* CRC press, 2013.

[96] D. I. Mattos, J. Bosch, and H. H. Olsson, "Statistical models for the analysis of optimization algorithms with benchmark functions," *arXiv preprint arXiv:2010.03783*, 2020.

[97] L. J. Cronbach and P. E. Meehl, "Construct validity in psychological tests." *Psychological bulletin*, vol. 52, no. 4, p. 281, 1955.

[98] D. T. Campbell, J. C. Stanley, and N. L. Gage, "Experimental and quasi-experimental designs for research." 1963.

[99] B. J. Calder, L. W. Phillips, and A. M. Tybout, "The concept of external validity," *Journal of consumer research*, vol. 9, no. 3, pp. 240–244, 1982.

[100] C. G. Victora, J.-P. Habicht, and J. Bryce, "Evidence-based public health: moving beyond randomized trials," *American journal of public health*, vol. 94, no. 3, pp. 400–405, 2004.

[101] A. Steckler and K. R. McLeroy, "The importance of external validity," 2008.

[102] L. Bickman, *Validity and social experimentation.* Sage, 2000, vol. 1.

[103] A. Elkhodary, N. Esfahani, and S. Malek, "Fusion: a framework for engineering self-tuning self-adaptive software systems," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, 2010, pp. 7–16.

[104] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," *Advances in neural information processing systems*, vol. 28, pp. 2503–2511, 2015.

[105] A. Jansen and J. Bosch, "Software architecture as a set of architectural design decisions," in *5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05).* IEEE, 2005, pp. 109–120.

[106] Z. Xu and J. D. Kalbfleisch, "Propensity score matching in randomized clinical trials," *Biometrics*, vol. 66, no. 3, pp. 813–823, 2010.

[107] H. H. Olsson and J. Bosch, "From opinions to data-driven software r&d: A multi-case study on how to close the'open loop'problem," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications.* IEEE, 2014, pp. 9–16.

[108] T. H. Davenport, "How to design smart business experiments," *Strategic Direction*, 2009.

[109] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "What's your ml test score? a rubric for ml production systems," 2016.

[110] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson, "Counterfactual reasoning and learning systems: The example of computational advertising," *Journal of Machine Learning Research*, vol. 14, no. 65, pp. 3207–3260, 2013. [Online]. Available: http://jmlr.org/papers/v14/bottou13a.html

[111] Y. Brun, G. D. M. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, "Engineering self-adaptive systems through feedback loops," in *Software engineering for self-adaptive systems.* Springer, 2009, pp. 48–70.

[112] C. Gatti, "Reinforcement learning," in *Design of Experiments for Reinforcement Learning.* Springer, 2015, pp. 7–52.

[113] R. S. Sutton, "Sutton & barto book: Reinforcement learning: An introduction," in *A Bradford Book.* MIT Press Cambridge, MA, 1998.

[114] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *European conference on machine learning.* Springer, 2005, pp. 437–448.

[115] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 661–670.

[116] L. Li, S. Chen, J. Kleban, and A. Gupta, "Counterfactual estimation and optimization of click metrics in search engines: A case study," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 929–934.

[117] S. L. Scott, "A modern bayesian look at the multi-armed bandit," *Applied Stochastic Models in Business and Industry*, vol. 26, no. 6, pp. 639–658, 2010.

[118] M. Salehie and L. Tahvildari, "Towards a goal-driven approach to action selection in self-adaptive software," *Software: Practice and Experience*, vol. 42, no. 2, pp. 211–233, 2012.

[119] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[120] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervasive and Mobile Computing*, vol. 17, pp. 184–206, 2015.

[121] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM transactions on autonomous and adaptive systems (TAAS)*, vol. 4, no. 2, pp. 1–42, 2009.

[122] I. A. Computing, ""white paper: An architectural blueprint for autonomic computing," 2005.

[123] J. Dowling and V. Cahill, "Self-managed decentralised systems using k-components and collaborative reinforcement learning," in *Proceedings of the 1st ACM SIGSOFT Workshop on Self-managed Systems*, 2004, pp. 39–43.

[124] D. Fisch, E. Kalkowski, and B. Sick, "Collaborative learning by knowledge exchange," in *Organic Computing—A Paradigm Shift for Complex Systems*. Springer, 2011, pp. 267–280.

[125] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, 2004.

[126] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," in *Future of Software Engineering (FOSE'07)*. IEEE, 2007, pp. 259–268.

[127] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self-adaptive software," *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 3, pp. 54–62, 1999.

[128] F. Kon, M. Román, P. Liu, J. Mao, T. Yamane, L. C. Magalhaes, and R. H. Campbell, "Monitoring, security, and dynamic configuration with the dynamictao reflective orb," in *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2000, pp. 121–143.

[129] L. Capra, W. Emmerich, and C. Mascolo, “Carisma: Context-aware reflective middleware system for mobile applications,” *IEEE Transactions on software engineering*, vol. 29, no. 10, pp. 929–945, 2003.

[130] T. Patikirikorala, A. Colman, J. Han, and L. Wang, “A systematic survey on the design of self-adaptive software systems using control engineering approaches,” in *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2012, pp. 33–42.

[131] D. Menasce, H. Gomaa, J. Sousa *et al.*, “Sassy: A framework for self-architecting service-oriented systems,” *IEEE software*, vol. 28, no. 6, pp. 78–85, 2011.

[132] G. Di Marzo Serugendo, J. Fitzgerald, and A. Romanovsky, “Metaself: an architecture and a development method for dependable self-* systems,” in *Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, pp. 457–461.

[133] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, and R. Mirandola, “Moses: A framework for qos driven runtime adaptation of service-oriented systems,” *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1138–1159, 2011.

[134] S. Hallsteinsen, K. Geihs, N. Paspallis, F. Eliassen, G. Horn, J. Lorenzo, A. Mamelli, and G. A. Papadopoulos, “A development framework and methodology for self-adapting applications in ubiquitous computing environments,” *Journal of Systems and Software*, vol. 85, no. 12, pp. 2840–2859, 2012.

[135] G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White, “A multi-agent systems approach to autonomic computing,” in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*. Citeseer, 2004, pp. 464–471.

[136] C. Ballagny, N. Hameurlain, and F. Barbier, “Mocas: A state-based component model for self-adaptation,” in *2009 Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 2009, pp. 206–215.

[137] M. Reichenbach, R. Seidler, D. Fey, and B. Pfundt, “Generic emergent computing in chip architectures,” in *Organic Computing—A Paradigm Shift for Complex Systems*. Springer, 2011, pp. 179–192.

[138] H. J. Goldsby, P. Sawyer, N. Bencomo, B. H. Cheng, and D. Hughes, “Goal-based modeling of dynamically adaptive system requirements,” in *15Th annual IEEE international conference and workshop on the engineering of computer based systems (ecbs 2008)*. IEEE, 2008, pp. 36–45.

[139] K. Angelopoulos, V. E. S. Souza, and J. Pimentel, “Requirements and architectural approaches to adaptive software systems: A comparative study,” in *2013 8th International Symposium on Software Engineering*

for Adaptive and Self-Managing Systems (SEAMS). IEEE, 2013, pp. 23–32.

[140] T. van Oosterhout and A. Visser, "A visual method for robot proxemics measurements," in *Proceedings of Metrics for Human-Robot Interaction: A Workshop at the Third ACM/IEEE International Conference on Human-Robot Interaction (HRI 2008). Citeseer*, 2008, pp. 61–68.

[141] H. H. Olsson and J. Bosch, "Towards continuous validation of customer value," in *Scientific Workshop Proceedings of the XP2015*, 2015, pp. 1–4.

[142] C. Ebert and C. Jones, "Embedded software: Facts, figures, and future," *Computer*, vol. 42, no. 4, pp. 42–52, 2009.

[143] G. Burtini, J. Loeppky, and R. Lawrence, "A survey of online experiment design with the stochastic multi-armed bandit," *arXiv preprint arXiv:1510.00757*, 2015.

[144] X. Zhang, *LTE optimization engineering handbook.* John Wiley & Sons, 2018.

[145] A. Awada, B. Wegmann, I. Viering, and A. Klein, "Optimizing the radio network parameters of the long term evolution system using taguchi's method," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 8, pp. 3825–3839, 2011.

[146] M. Dottling and I. Viering, "Challenges in mobile network operation: Towards self-optimizing networks," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing.* IEEE, 2009, pp. 3609–3612.

[147] S. Dastoor, U. Dalal, and J. Sarvaiya, "Comparative analysis of optimization techniques for optimizing the radio network parameters of next generation wireless mobile communication," in *2017 Fourteenth International Conference on Wireless and Optical Communications Networks (WOCN).* IEEE, 2017, pp. 1–6.

[148] K. Lieska, E. Laitinen, and J. Lahteenmaki, "Radio coverage optimization with genetic algorithms," in *Ninth IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (Cat. No. 98TH8361)*, vol. 1. IEEE, 1998, pp. 318–322.

[149] I. Cavdar and O. Akcay, "The optimization of cell sizes and base stations power level in cell planning," in *IEEE VTS 53rd Vehicular Technology Conference, Spring 2001. Proceedings (Cat. No. 01CH37202)*, vol. 4. IEEE, 2001, pp. 2344–2348.

[150] H. Zhu and T. Buot, "Multi-parameter optimization in wcdma radio networks," in *2004 IEEE 59th Vehicular Technology Conference. VTC 2004-Spring (IEEE Cat. No. 04CH37514)*, vol. 4. IEEE, 2004, pp. 2370–2374.

[151] ETSI, "3GPP Technical Specification Release 14 - ETSI TS 136 300," ETSI, Valbonne, France, Tech. Rep. Release 14, 2017.

[152] R. Munos, "From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning," 2014.

[153] J. Krettek, D. Schauten, F. Hoffmann, and T. Bertram, "Evolutionary hardware-in-the-loop optimization of a controller for cascaded hydraulic valves," in *2007 IEEE/ASME international conference on advanced intelligent mechatronics*. IEEE, 2007, pp. 1–6.

[154] A. Krause and C. S. Ong, "Contextual gaussian process bandit optimization." in *Nips*, 2011, pp. 2447–2455.

[155] J. Ling, M. Hutchinson, E. Antono, S. Paradiso, and B. Meredig, "High-dimensional materials and process optimization using data-driven experimental design with well-calibrated uncertainty estimates," *Integrating Materials and Manufacturing Innovation*, vol. 6, no. 3, pp. 207–217, 2017.

[156] S. Huang, T. Han, and N. Ansari, "Big-data-driven network partitioning for ultra-dense radio access networks," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[157] P. Dmitriev, B. Frasca, S. Gupta, R. Kohavi, and G. Vaz, "Pitfalls of long-term online controlled experiments," in *2016 IEEE international conference on big data (big data)*. IEEE, 2016, pp. 1367–1376.

[158] T. Crook, B. Frasca, R. Kohavi, and R. Longbotham, "Seven pitfalls to avoid when running controlled experiments on the web," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 1105–1114.

[159] T. Kluck and L. Vermeer, "Leaky abstraction in online experimentation platforms: a conceptual framework to categorize common challenges," *arXiv preprint arXiv:1710.00397*, 2017.

[160] R. Chen, M. Chen, M. R. Jadav, J. Bae, and D. Matheson, "Faster online experimentation by eliminating traditional a/a validation," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 1635–1641.

[161] 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, 2018, pp. 1–109.

[162] P. Dmitriev and X. Wu, "Measuring metrics," in *Proceedings of the 25th ACM international on conference on information and knowledge management*, 2016, pp. 429–437.

[163] 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*, 2013, pp. 1168–1176.

[164] A. Deng, J. Lu, and J. Litz, "Trustworthy analysis of online a/b tests: Pitfalls, challenges and solutions," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 641–649.

[165] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, 2014, pp. 1–9.

[166] A. Hern, "Why google has 200m reasons to put engineers over designers," *The Guardian*, vol. 5, 2014.

[167] S. L. Scott, "Multi-armed bandit experiments in the online service economy," *Applied Stochastic Models in Business and Industry*, vol. 31, no. 1, pp. 37–45, 2015.

[168] N. Juristo and A. M. Moreno, *Basics of software engineering experimentation.* Springer Science & Business Media, 2013.

[169] S. L. Scott, "Google analytics help page," 2017. [Online]. Available: https://support.google.com/analytics/answer/2844870?hl=en&ref_topic=1745207.

[170] J. W. Creswell, *Educational research: Planning, conducting, and evaluating quantitative.* Prentice Hall Upper Saddle River, NJ, 2002.

[171] C. Stucchio, "Saturday is not tuesday," 2015. [Online]. Available: https://www.chrisstucchio.com/blog/2015/dont_use_bandits.html.

[172] J. White, *Bandit algorithms for website optimization.* " O'Reilly Media, Inc.", 2012.

[173] S. Guha, K. Munagala, and M. Pal, "Multiarmed bandit problems with delayed feedback," *arXiv preprint arXiv:1011.1161*, 2010.

[174] R. Kievit, W. E. Frankenhuis, L. Waldorp, and D. Borsboom, "Simpson's paradox in psychological science: a practical guide," *Frontiers in psychology*, vol. 4, p. 513, 2013.

[175] L. Li, "Offline evaluation and optimization for interactive systems," ser. WSDM '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 413–414. [Online]. Available: https://doi.org/10.1145/2684822.2697040

[176] M. M. Drugan and A. Nowe, "Designing multi-objective multi-armed bandits algorithms: A study," in *The 2013 International Joint Conference on Neural Networks (IJCNN).* IEEE, 2013, pp. 1–8.

[177] X. Wang, Y. Wang, D. Hsu, and Y. Wang, "Exploration in interactive personalized music recommendation: a reinforcement learning approach," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 11, no. 1, pp. 1–22, 2014.

[178] R. Féraud and T. Urvoy, "A stochastic bandit algorithm for scratch games," in *Asian Conference on Machine Learning.* PMLR, 2012, pp. 129–143.

[179] L. Li, W. Chu, J. Langford, and X. Wang, "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 297–306.

[180] K. Jamieson, "Fdr control with adaptive sequential experimental design," 2017.

[181] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *arXiv preprint arXiv:1402.6028*, 2014.

[182] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, and H. China, "Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization," Tech. Rep., 2018.

[183] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[184] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical programming*, vol. 91, no. 2, pp. 201–213, 2002.

[185] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. Resende, and W. R. Stewart, "Designing and reporting on computational experiments with heuristic methods," *Journal of Heuristics*, vol. 1, no. 1, pp. 9–32, 1995.

[186] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization," *Journal of Heuristics*, vol. 15, no. 6, p. 617, 2009.

[187] A. Lacoste, F. Laviolette, and M. Marchand, "Bayesian comparison of machine learning algorithms on single and multiple datasets," in *Artificial Intelligence and Statistics*, 2012, pp. 665–675.

[188] T. Eftimov and P. Korošec, "The Impact of Statistics for Benchmarking in Evolutionary Computation Research," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '18. ACM, 2018, p. 1329–1336.

[189] T. Eftimov and P. Korošec, "A novel statistical approach for comparing meta-heuristic stochastic optimization algorithms according to the distribution of solutions in the search space," *Information Sciences*, vol. 489, pp. 255–273, 2019.

[190] M. Gagliolo and C. Legrand, "Algorithm survival analysis," in *Experimental methods for the analysis of optimization algorithms.* Springer, 2010, pp. 161–184.

[191] M. Chiarandini and Y. Goegebeur, "Mixed models for the analysis of optimization algorithms," in *Experimental Methods for the Analysis of Optimization Algorithms.* Springer, 2010, pp. 225–264.

[192] B. Calvo, O. M. Shir, J. Ceberio, C. Doerr, H. Wang, T. Bäck, and J. A. Lozano, "Bayesian Performance Analysis for Black-Box Optimization Benchmarking," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '19. ACM, 2019, p. 1789–1797.

[193] J. Carrasco, S. García, M. del Mar Rueda, and F. Herrera, "rnpbst: An R package covering non-parametric and Bayesian statistical tests," in *International Conference on Hybrid Artificial Intelligence Systems.* Springer, 2017, pp. 281–292.

[194] G. Corani and A. Benavoli, " Bayesian approach for comparing cross-validated algorithms on multiple data sets," *Machine Learning*, vol. 100, no. 2-3, pp. 285–304, 2015.

[195] A. Benavoli, G. Corani, F. Mangili, M. Zaffalon, and F. Ruggeri, "A Bayesian Wilcoxon signed-rank test based on the Dirichlet process," in *International conference on machine learning*, 2014, pp. 1026–1034.

[196] R. McElreath, *Statistical rethinking: A Bayesian course with examples in R and Stan.* CRC press, 2020.

[197] J. K. Kruschke, "Bayesian estimation supersedes the t-test," *Journal of Experimental Psychology: General*, vol. 142, no. 2, p. 573, 2013.

[198] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell, "Stan: A Probabilistic Programming Language," *Journal of Statistical Software, Articles*, vol. 76, no. 1, pp. 1–32, 2017.

[199] M. D. Hoffman and A. Gelman, "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1593–1623, 2014.

[200] A. Gelman and D. B. Rubin, "Inference from Iterative Simulation Using Multiple Sequences," *Statistical Science*, vol. 7, no. 4, pp. 457 – 472, 1992.

[201] A. Gelman, J. Hwang, and A. Vehtari, "Understanding predictive information criteria for Bayesian models," *Statistics and Computing*, vol. 24, no. 6, pp. 997–1016, 2014.

[202] J. Miller, "What is the probability of replicating a statistically significant effect?" *Psychonomic Bulletin & Review*, vol. 16, no. 4, pp. 617–640, 2009.

[203] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[204] X.-S. Yang and S. Deb, "Cuckoo Search via Lévy flights," in *2009 World congress on nature & biologically inspired computing (NaBIC)*. IEEE, 2009, pp. 210–214.

[205] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[206] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[207] N. Hansen, Y. Akimoto, and P. Baudis, "CMA-ES/pycma on Github," Zenodo, DOI:10.5281/zenodo.2559634, Feb. 2019.

[208] S. Finck and R. Ros, "Real-Parameter Black-Box Optimization Benchmarking 2010: Noiseless Functions Definitions," Tech. Rep. RR-6829, 2009.

[209] T. A. Snijders and R. J. Bosker, *Multilevel analysis: An introduction to basic and advanced multilevel modeling.* Sage, 2011.

[210] A. Agresti, *Categorical data analysis.* John Wiley & Sons, 2003.

[211] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. The method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[212] D. I. Mattos and E. M. S. Ramos, "Bayesian Paired-Comparison with the bpcs Package," *arXiv preprint arXiv:2101.11227*, 2021.

[213] F. Caron and A. Doucet, "Efficient Bayesian Inference for Generalized Bradley–Terry Models," *Journal of Computational and Graphical Statistics*, vol. 21, no. 1, pp. 174–196, 2012.

[214] H. Turner and D. Firth, "Bradley-Terry Models in R: The BradleyTerry2 Package," *Journal of Statistical Software, Articles*, vol. 48, no. 9, pp. 1–21, 2012.

[215] M. Cattelan, "Models for Paired Comparison Data: A Review with Emphasis on Dependent Data," *Statistical Science*, vol. 27, no. 3, pp. 412 – 433, 2012.

[216] H. L. Turner, J. van Etten, D. Firth, and I. Kosmidis, "Modelling rankings in R: the PlackettLuce package," *Computational Statistics*, vol. 35, no. 3, pp. 1027–1057, Sep 2020.

[217] R. R. Davidson, "On extending the Bradley-Terry model to accommodate ties in paired comparison experiments," *Journal of the American Statistical Association*, vol. 65, no. 329, pp. 317–328, 1970.

[218] T. G. Clark, M. J. Bradburn, S. B. Love, and D. G. Altman, "Survival analysis part I: basic concepts and first analyses," *British journal of cancer*, vol. 89, no. 2, pp. 232–238, 2003.

[219] R. Kelter, "Bayesian survival analysis in Stan for improved measuring of uncertainty in parameter estimates," *Measurement: Interdisciplinary Research and Perspectives*, vol. 18, no. 2, pp. 101–109, 2020.

[220] U. Eliasson, R. Heldal, E. Knauss, and P. Pelliccione, "The need of complementing plan-driven requirements engineering with emerging communication: Experiences from volvo car group," in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. IEEE, 2015, pp. 372–381.

[221] H. H. Olsson and J. Bosch, "Climbing the "stairway to heaven": evolving from agile development to continuous deployment of software," in *Continuous software engineering*. Springer, 2014, pp. 15–27.

[222] A. Fabijan, H. H. Olsson, and J. Bosch, "The lack of sharing of customer data in large software organizations: challenges and implications," in *International Conference on Agile Software Development*. Springer, 2016, pp. 39–52.

[223] H. Olsson Holmström, "So much data-so little value: A multi-case study on improving the impact of data-driven development practices." 20th Iberoamerican Conference on Software Engineering (CIbSE 2017);, 2017.

[224] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, "Systematic literature reviews," in *Experimentation in software engineering*. Springer, 2012, pp. 45–54.

[225] B. Zhang, N. Wang, and H. Jin, "Privacy concerns in online recommender systems: influences of control and user data input," in *10th Symposium On Usable Privacy and Security ({SOUPS} 2014)*, 2014, pp. 159–173.

[226] H. H. Olsson and J. Bosch, "From ad hoc to strategic ecosystem management: the "three-layer ecosystem strategy model"(telesm)," *Journal of Software: Evolution and Process*, vol. 29, no. 7, p. e1876, 2017.

[227] F. Auer and M. Felderer, "Current state of research on continuous experimentation: a systematic mapping study," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2018, pp. 335–344.

[228] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: A roadmap and agenda," *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.

[229] G. Schermann, J. Cito, and P. Leitner, "Continuous experimentation: challenges, implementation techniques, and current research," *Ieee Software*, vol. 35, no. 2, pp. 26–31, 2018.

[230] S. G. Yaman, F. Fagerholm, M. Munezero, J. Münch, M. Aaltola, C. Palmu, and T. Männistö, "Transitioning towards continuous experimentation in a large software product and service development organisation–a case study," in *International Conference on Product-Focused Software Process Improvement*. Springer, 2016, pp. 344–359.

[231] K. Fowler, "Mission-critical and safety-critical development," *IEEE Instrumentation & Measurement Magazine*, vol. 7, no. 4, pp. 52–59, 2004.

[232] A. Fabijan, P. Dmitriev, C. McFarland, L. Vermeer, H. Holmström 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, 2018.

[233] F. Auer, C. S. Lee, and M. Felderer, "Continuous experiment definition characteristics." 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2020, pp. 186–190.

[234] A. S. M. Zain, M. F. A. Malek, M. Elshaikh, and N. Omar, "Optimization of resource allocation using taguchi's method for lte-advanced network," in *2014 2nd International Conference on Electronic Design (ICED)*. IEEE, 2014, pp. 281–286.

[235] S. K. Karna, R. Sahai *et al.*, "An overview on taguchi method," *International journal of engineering and mathematical sciences*, vol. 1, no. 1, pp. 1–7, 2012.

[236] J. A. Middleton and P. M. Aronow, "Unbiased estimation of the average treatment effect in cluster-randomized experiments," *Statistics, Politics and Policy*, vol. 6, no. 1-2, pp. 39–75, 2015.

[237] S. W. Raudenbush, "Statistical analysis and optimal design for cluster randomized trials." *Psychological methods*, vol. 2, no. 2, p. 173, 1997.

[238] M. T. Rahman, L.-P. Querel, P. C. Rigby, and B. Adams, "Feature toggles: practitioner practices and a case study." Proceedings of the 13th International Conference on Mining Software Repositories, 2016, pp. 201–211.

[239] S. Neely and S. Stolt, "Continuous delivery? easy! just change everything (well, maybe it is not that easy)." 2013 Agile Conference, 2013, pp. 121–128.

[240] D. G. Feitelson, E. Frachtenberg, and K. L. Beck, "Development and deployment at facebook," *IEEE Internet Computing*, vol. 17, no. 4, pp. 8–17, 2013.

[241] E. Pakarinen, T. Harakkamäki, and T. Mikkonen, "Gradual deployment in practice: Experiences from an industrial case study." 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2020, pp. 237–241.

[242] T. Xia, S. Bhardwaj, P. Dmitriev, and A. Fabijan, "Safe velocity: a practical guide to software deployment at scale using controlled rollout," 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, 2019, pp. 11–20.

[243] D. Ståhl and J. Bosch, "Cinders: The continuous integration and delivery architecture framework," *Information and Software Technology*, vol. 83, pp. 76–93, 2017.

[244] P. Rodríguez, A. Haghighatkhah, L. E. Lwakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J. M. Verner, and M. Oivo, "Continuous deployment of software intensive products and services: A systematic mapping study," *Journal of Systems and Software*, vol. 123, pp. 263–291, 2017.

[245] D. E. Johnson, "Crossover experiments," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 5, pp. 620–625, 2010.

[246] A. Fabijan, P. Dmitriev, H. H. Olsson, J. Bosch, L. Vermeer, and D. Lewis, "Three key checklists and remedies for trustworthy analysis of online controlled experiments at scale," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 1–10.

[247] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1487–1495.

[248] D. I. Mattos, L. Ruud, J. Bosch, and H. H. Olsson, "On the assessment of benchmark suites for algorithm comparison," *arXiv preprint arXiv:2104.07381*, 2021.

[249] R. Sveningson, D. I. Mattos, and J. Bosch, "Continuous experimentation for software organizations with low control of roadmap and a large distance to users: an exploratory case study," in *International Conference on Product-Focused Software Process Improvement*. Springer, 2019, pp. 528–544.

[250] Y. Liu, D. I. Mattos, J. Bosch, H. H. Olsson, and J. Lantz, "Online a/b experiments design with limited sample," in *in submission*, 2021, pp. 1–10.

[251] L. N. Soldatova, A. Clare, A. Sparkes, and R. D. King, "An ontology for a robot scientist," *Bioinformatics*, vol. 22, no. 14, pp. e464–e471, 2006.

[252] L. N. Soldatova and R. D. King, "An ontology of scientific experiments," *Journal of the Royal Society Interface*, vol. 3, no. 11, pp. 795–803, 2006.

[253] R. H. Hoyle, *Handbook of structural equation modeling*. Guilford press, 2012.

[254] T. Joachims and A. Swaminathan, "Counterfactual evaluation and learning for search, recommendation and ad placement," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 1199–1201.

[255] A. Swaminathan and T. Joachims, "Counterfactual risk minimization: Learning from logged bandit feedback," in *International Conference on Machine Learning*. PMLR, 2015, pp. 814–823.

[256] ——, "The self-normalized estimator for counterfactual learning," in *advances in neural information processing systems*. Citeseer, 2015, pp. 3231–3239.

[257] A. Chuklin, I. Markov, and M. d. Rijke, "Click models for web search," *Synthesis lectures on information concepts, retrieval, and services*, vol. 7, no. 3, pp. 1–115, 2015.

[258] T. Treasure and K. D. MacRae, "Minimisation: the platinum standard for trials?: randomisation doesn't guarantee similarity of groups; minimisation does," 1998.

[259] D. R. Taves, "Minimization: a new method of assigning patients to treatment and control groups," *Clinical Pharmacology & Therapeutics*, vol. 15, no. 5, pp. 443–453, 1974.

[260] S. J. Pocock and R. Simon, "Sequential treatment assignment with balancing for prognostic factors in the controlled clinical trial," *Biometrics*, pp. 103–115, 1975.

[261] L. Freedman and S. J. White, "On the use of pocock and simon's method for balancing treatment numbers over prognostic factors in the controlled clinical trial," *Biometrics*, pp. 691–694, 1976.

[262] E. Forsell, J. Beckley, S. Ejdemyr, V. Hannan, A. Rhines, M. Tingley, M. Wardrop, and J. Wong, "Success stories from a democratized experimentation platform," *arXiv preprint arXiv:2012.10403*, 2020.

[263] S. J. Taylor and B. Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[264] M. A. Hernán, "Beyond exchangeability: the other conditions for causal inference in medical research," 2012.

[265] O. Saarela, D. A. Stephens, and E. E. Moodie, "The role of exchangeability in causal inference," *arXiv preprint arXiv:2006.01799*, 2020.

[266] M. Lechner *et al.*, *The estimation of causal effects by difference-in-difference methods*. Now, 2011.

[267] D. L. Thistlethwaite and D. T. Campbell, "Regression-discontinuity analysis: An alternative to the ex post facto experiment." *Journal of Educational psychology*, vol. 51, no. 6, p. 309, 1960.