On Experimentation in Software-Intensive Systems

David Issa Mattos
On Experimentation in Software-Intensive Systems

DAVID ISSA MATTOS

Copyright ©2021 David Issa Mattos
except where otherwise stated.
All rights reserved.

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 time 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:


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.


[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 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).

In submission to a conference (2021).

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

Abstract v

Acknowledgement vii

List of Publications ix

Personal Contribution xiii

1 Introduction 1

2 Background 5
  2.1 Experiments in software systems ............................ 5
  2.2 Randomized experiments ........................................ 5
  2.3 Other experimental designs ................................... 6
    2.3.1 Factorial and fractional experiments ......................... 6
    2.3.2 Quasi-experiments ............................................ 6
    2.3.3 Crossover experiments ....................................... 7
    2.3.4 Multi-armed bandit experiments .............................. 7
  2.4 Continuous experimentation .................................... 7
  2.5 Experimentation models ........................ 8
    2.5.1 The Build-Measure-Learn model ............................ 8
    2.5.2 The ESSSDM model ....................................... 9
    2.5.3 The QCD model ....................................... 9
    2.5.4 HYPEX model ........................................ 9
    2.5.5 The RIGHT model ..................................... 10
  2.6 Automated experiments ..................................... 10
    2.6.1 Algorithms for online optimization ......................... 11
  2.7 Experimentation in the embedded systems ...................... 11
  2.8 Experimentation in the B2B domain ............................ 12

3 Research approach 15
  3.1 Objectives ................................................... 15
    3.1.1 Objective 1 ............................................. 15
    3.1.2 Objective 2 ........................................... 16
  3.2 Research context ............................................. 17
    3.2.1 Company collaborations .................................... 17
  3.3 Research strategies .......................................... 19
    3.3.1 Field study ........................................... 19
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.1.1</td>
<td>Case studies</td>
<td>19</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Sample studies</td>
<td>20</td>
</tr>
<tr>
<td>3.3.2.1</td>
<td>Literature review</td>
<td>21</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Laboratory experiments</td>
<td>21</td>
</tr>
<tr>
<td>3.3.3.1</td>
<td>Controlled experiments</td>
<td>21</td>
</tr>
<tr>
<td>3.3.3.2</td>
<td>Benchmarking studies</td>
<td>21</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Summary</td>
<td>22</td>
</tr>
<tr>
<td>3.4</td>
<td>Data analysis</td>
<td>22</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Thematic coding</td>
<td>23</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Statistical analysis</td>
<td>24</td>
</tr>
<tr>
<td>3.4.2.1</td>
<td>The frequentist paradigm</td>
<td>24</td>
</tr>
<tr>
<td>3.4.2.2</td>
<td>The Bayesian paradigm</td>
<td>25</td>
</tr>
<tr>
<td>3.5</td>
<td>Validity considerations</td>
<td>25</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Construct validity</td>
<td>26</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Internal validity</td>
<td>26</td>
</tr>
<tr>
<td>3.5.3</td>
<td>External validity</td>
<td>27</td>
</tr>
<tr>
<td>3.5.4</td>
<td>Conclusion validity</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>Contributions of this thesis</td>
<td>29</td>
</tr>
<tr>
<td>4.1</td>
<td>General overview</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>Included publications</td>
<td>32</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Paper A: “Your system gets better every day you use it:</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td><em>towards automated continuous experimentation</em></td>
<td></td>
</tr>
<tr>
<td>4.2.2</td>
<td>Paper B: “Automated Optimization of Software Parameters in a</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Long Term Evolution Radio Base Station”</td>
<td></td>
</tr>
<tr>
<td>4.2.3</td>
<td>Paper C: “An activity and metric model for online controlled</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>experiments”</td>
<td></td>
</tr>
<tr>
<td>4.2.4</td>
<td>Paper D: “Multi-armed bandits in the wild: Pitfalls and strategies</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>in online experiments</td>
<td></td>
</tr>
<tr>
<td>4.2.5</td>
<td>Paper E: “Statistical Models for the Analysis of Optimization</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Algorithms with Benchmark Functions”</td>
<td></td>
</tr>
<tr>
<td>4.2.6</td>
<td>Paper F: “Challenges and Strategies for Undertaking Continuous</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>Experimentation to Embedded Systems: Industry and Research Perspectives”</td>
<td></td>
</tr>
</tbody>
</table>
CONTENTS

4.2.6.1 Summary of the study ........................................ 37
4.2.6.2 Research method ............................................ 37
4.2.6.3 Main results .................................................. 38

4.2.7.1 Summary of the study ........................................ 38
4.2.7.2 Research method ............................................. 39
4.2.7.3 Main results .................................................. 39

4.3 Related publications .................................................. 39
4.3.1 Paper a: “More for less: automated experimentation in software-intensive systems” ........................................... 40
4.3.2 Paper b: “Optimization Experiments in the Continuous Space” ................................................................. 40
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” ........................................ 41
4.3.4 Paper e: “ACE: Easy Deployment of Field Optimization Experiments” ............................................................ 41
4.3.5 Paper g: “Engineering for a Science-Centric Experimentation Platform” .......................................................... 41
4.3.7 Paper j: “Automotive A/B Testing: Challenges and Lessons Learned from Practice” ........................................ 42
4.3.8 Paper n: “Bayesian Paired-Comparison with the bpcs Package” ................................................................. 43
4.3.9 Paper p: “Size matters? Or not: A/B testing with limited sample in automotive embedded software” .......................... 43
4.3.10 Paper s: “On the assessment of benchmark suites for algorithm comparison” .............................................. 43

5 Paper A: Your system gets better every day you use it: towards automated continuous experimentation 45
5.1 Introduction ........................................................... 46
5.2 Background ........................................................... 48
  5.2.1 Controlled experiments ........................................ 48
  5.2.2 Reinforcement Learning in experiments ....................... 48
  5.2.3 Software architectures for adaptation ......................... 49
5.3 Research Method ....................................................... 49
5.4 Architecture qualities ............................................... 50
  5.4.1 External adaptation control .................................... 51
  5.4.2 Data collection as an integral part of the architecture .... 51
  5.4.3 Performance reflection ......................................... 51
  5.4.4 Explicit representation of the learning component .......... 52
  5.4.5 Decentralized adaptation ....................................... 52
  5.4.6 Knowledge exchange ........................................... 52
5.5 Architecture analysis ............................................... 53
5.6 Architecture framework ............................................. 55
  5.6.1 The architecture framework ................................... 55
5.7 Automated experiments in a human-robot interaction problem 57  
5.7.1 Proxemics distance in Human-Robot Interaction 57  
5.7.2 Experimental results 60  
5.8 Conclusion 60  
5.8.1 Research Challenges 62  

6 Paper B: Automated Optimization of Software Parameters in a Long Term Evolution Radio Base Station 65  
6.1 Introduction 66  
6.2 Background 67  
6.2.1 Overview of a Radio Base Station 67  
6.2.2 LTE Optimization Overview 67  
6.2.3 Radio Base Station Parameters and Metrics 68  
6.2.3.1 Metrics 68  
6.2.3.2 Parameters 68  
6.2.4 The Online Optimization Problem 69  
6.3 Experimental setup 71  
6.3.1 The ACE system 71  
6.3.2 Testing bed setup 71  
6.4 Results 73  
6.4.1 Experimental runs 73  
6.4.2 Optimization of multiple parameters 73  
6.4.3 Integration in a deployed RBS 75  
6.4.4 Limitations 75  
6.5 Related work 77  
6.6 Conclusion 78  

7 Paper C: An activity and metric model for online controlled experiments 81  
7.1 Introduction 82  
7.2 Background and related work 83  
7.3 Research Method 85  
7.3.1 Data collection 85  
7.3.2 Data analysis 86  
7.3.3 Validity considerations 86  
7.4 Findings 87  
7.4.1 Customer feedback is an important source of experimentation ideas. 87  
7.4.2 Metrics guide experiments towards long-term goals and help prioritize hypotheses. 88  
7.4.3 Metrics evolve and capture the experiment assumptions. 88  
7.5 The experimentation process framework 89  
7.5.1 The experimentation activity model 89  
7.5.1.1 Experiment development phase. 90  
7.5.1.2 Experiment execution phase. 92  
7.5.1.3 Experiment analysis phase. 93  
7.5.2 The experiment metric model. 94  
7.6 Conclusion 96
# Paper D: Multi-armed bandits in the wild: Pitfalls and strategies in online experiments

## 8 Paper D: Multi-armed bandits in the wild: Pitfalls and strategies in online experiments

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Introduction</td>
<td>100</td>
</tr>
<tr>
<td>8.2</td>
<td>Background</td>
<td>100</td>
</tr>
<tr>
<td>8.2.1</td>
<td>A/B experiments</td>
<td>101</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Multi-Armed Bandit</td>
<td>102</td>
</tr>
<tr>
<td>8.3</td>
<td>Research method</td>
<td>103</td>
</tr>
<tr>
<td>8.3.1</td>
<td>Multiple case study</td>
<td>103</td>
</tr>
<tr>
<td>8.3.1.1</td>
<td>Definition and planning</td>
<td>103</td>
</tr>
<tr>
<td>8.3.1.2</td>
<td>Data selection and collection</td>
<td>103</td>
</tr>
<tr>
<td>8.3.1.3</td>
<td>Data analysis</td>
<td>105</td>
</tr>
<tr>
<td>8.3.2</td>
<td>Simulations</td>
<td>105</td>
</tr>
<tr>
<td>8.3.3</td>
<td>Threats to validity</td>
<td>106</td>
</tr>
<tr>
<td>8.3.3.1</td>
<td>Construct validity</td>
<td>106</td>
</tr>
<tr>
<td>8.3.3.2</td>
<td>External validity</td>
<td>106</td>
</tr>
<tr>
<td>8.3.3.3</td>
<td>Internal validity</td>
<td>106</td>
</tr>
<tr>
<td>8.4</td>
<td>Results</td>
<td>107</td>
</tr>
<tr>
<td>8.4.1</td>
<td>Decision errors in naive MAB implementations</td>
<td>107</td>
</tr>
<tr>
<td>8.4.1.1</td>
<td>Pitfall</td>
<td>107</td>
</tr>
<tr>
<td>8.4.1.2</td>
<td>Simulation</td>
<td>107</td>
</tr>
<tr>
<td>8.4.1.3</td>
<td>Strategies</td>
<td>108</td>
</tr>
<tr>
<td>8.4.2</td>
<td>Bad variation lockdown</td>
<td>109</td>
</tr>
<tr>
<td>8.4.2.1</td>
<td>Restriction</td>
<td>109</td>
</tr>
<tr>
<td>8.4.2.2</td>
<td>Strategies</td>
<td>110</td>
</tr>
<tr>
<td>8.4.3</td>
<td>Decision errors due to violations of assumptions</td>
<td>111</td>
</tr>
<tr>
<td>8.4.3.1</td>
<td>Pitfalls</td>
<td>111</td>
</tr>
<tr>
<td>8.4.3.2</td>
<td>Simulation</td>
<td>112</td>
</tr>
<tr>
<td>8.4.3.3</td>
<td>Strategies</td>
<td>114</td>
</tr>
<tr>
<td>8.4.4</td>
<td>Lack of Sample Ratio Mismatch quality check in MAB algorithms</td>
<td>114</td>
</tr>
<tr>
<td>8.4.4.1</td>
<td>Restriction</td>
<td>114</td>
</tr>
<tr>
<td>8.4.4.2</td>
<td>Strategy</td>
<td>115</td>
</tr>
<tr>
<td>8.4.5</td>
<td>Increased complexity in ramp-up procedures in MAB algorithms</td>
<td>115</td>
</tr>
<tr>
<td>8.4.5.1</td>
<td>Restriction</td>
<td>115</td>
</tr>
<tr>
<td>8.4.5.2</td>
<td>Strategy</td>
<td>116</td>
</tr>
<tr>
<td>8.4.6</td>
<td>Increasing regret in experiments due to Simpson’s Paradox in MAB algorithms</td>
<td>116</td>
</tr>
<tr>
<td>8.4.6.1</td>
<td>Pitfall</td>
<td>116</td>
</tr>
<tr>
<td>8.4.6.2</td>
<td>Simulation</td>
<td>117</td>
</tr>
<tr>
<td>8.4.6.3</td>
<td>Strategy</td>
<td>118</td>
</tr>
<tr>
<td>8.4.7</td>
<td>Adaptive allocation based on a single metric</td>
<td>119</td>
</tr>
<tr>
<td>8.4.7.1</td>
<td>Restrictions</td>
<td>119</td>
</tr>
<tr>
<td>8.4.7.2</td>
<td>Strategy</td>
<td>120</td>
</tr>
<tr>
<td>8.5</td>
<td>Discussion</td>
<td>120</td>
</tr>
<tr>
<td>8.5.1</td>
<td>Use cases for multi-armed bandits</td>
<td>121</td>
</tr>
<tr>
<td>8.5.1.1</td>
<td>Content-serving systems</td>
<td>121</td>
</tr>
<tr>
<td>8.5.1.2</td>
<td>Short-term campaigns</td>
<td>123</td>
</tr>
<tr>
<td>8.5.1.3</td>
<td>Targeting experiments</td>
<td>123</td>
</tr>
</tbody>
</table>
CONTENTS

8.5.1.4 Other cases ........................................ 123
8.5.2 Guidelines ................................................ 123
8.6 Conclusion .................................................... 124
8.7 Appendix ...................................................... 124
8.7.1 Multi-Armed Bandit algorithms used in the simulations 124
8.7.1.1 Explore-First algorithm with parameter $N$ ................ 124
8.7.2 The Softmax algorithm .................................... 126
8.7.3 The UCB1 algorithm ........................................ 126
8.7.4 Further extensions .......................................... 127

9 Paper E: Statistical Models for the Analysis of Optimization Algorithms with Benchmark Functions 129
9.1 Introduction .................................................. 130
9.2 Related work .................................................. 131
9.3 Bayesian Data Analysis ....................................... 132
  9.3.1 Bayesian tools ............................................ 133
  9.3.2 Bayesian inference and MCMC ............................ 133
  9.3.3 Posterior and intervals .................................... 133
    9.3.3.1 Equal tail interval .................................. 134
    9.3.3.2 Highest Posterior Density (HPD) Interval .......... 134
    9.3.3.3 Region of Practical Equivalence (ROPE) ......... 134
  9.3.4 Model checking ........................................... 135
    9.3.4.1 Sampling convergence ................................ 135
    9.3.4.2 Choice of priors ..................................... 135
    9.3.4.3 Model comparison .................................... 136
    9.3.4.4 Sensitivity analysis .................................. 137
    9.3.4.5 Posterior predictive checking ...................... 137
    9.3.4.6 Sample size and power analysis ..................... 137
  9.4 The empirical data .......................................... 137
    9.4.1 The algorithms ......................................... 138
    9.4.2 The benchmark functions ................................ 139
    9.4.3 The experimental conditions ............................ 139
    9.4.4 The logged metrics ..................................... 139
    9.4.5 The research questions .................................. 140
  9.5 Statistical Models ........................................... 141
    9.5.1 Compensating the effects of benchmarks .............. 142
    9.5.2 Probability of success .................................. 143
      9.5.2.1 The model .......................................... 143
      9.5.2.2 Model interpretation ............................... 144
      9.5.2.3 Remarks ............................................ 145
    9.5.3 Algorithm relative improvement over Random Search . 146
      9.5.3.1 The model .......................................... 146
      9.5.3.2 Model interpretation ............................... 147
      9.5.3.3 Remarks ............................................ 148
    9.5.4 Ranking comparison ..................................... 148
      9.5.4.1 The model .......................................... 148
      9.5.4.2 Model interpretation ............................... 149
      9.5.4.3 Remarks ............................................ 150
    9.5.5 Number of function evaluations to converge to a solution 151
9.5.5.1 The model ............................................. 152
9.5.5.2 Model interpretation .......................... 153
9.5.5.3 Remarks ............................................. 155
9.5.6 Multiple group comparison of CPU time .......... 155
9.5.6.1 The model ............................................. 156
9.5.6.2 Model interpretation .......................... 156
9.5.6.3 Remarks ............................................. 157
9.5.7 Extending the models ................................ 158
9.6 Conclusion ................................................. 158

10 Paper F: Challenges and Strategies for Undertaking Continuous Experimentation to Embedded Systems: Industry and Research Perspectives 161
10.1 Introduction .............................................. 162
10.2 Background .............................................. 163
10.3 Research method ........................................ 165
10.3.1 Literature review .................................. 165
10.3.2 Multiple case study ................................. 166
10.4 Challenges and proposed strategies ................... 167
10.4.1 Technical Challenges ............................... 168
10.4.1.1 Lack of over the air (OTA) updates and data 
          collection, ............................................. 168
10.4.1.2 Lack of experimentation tools that integrate 
          with their existing tooling ........................ 168
10.4.1.3 Expensive testing environments ................ 170
10.4.1.4 Experimentation constraints in real-time and 
          safety-critical systems ............................. 170
10.4.2 Business Challenges ................................. 171
10.4.2.1 Determining good experimentation metrics and 
          metrics validation, .................................. 171
10.4.2.2 Privacy concerns regarding user data ............ 171
10.4.2.3 Lack of sharing user data in business-to-business 
          (B2B) solutions ......................................... 171
10.4.2.4 Lack of insights obtained from the collected data. 172
10.4.2.5 Long release cycles .............................. 172
10.4.3 Organizational Challenges .......................... 173
10.4.3.1 Managing multiple stakeholders in the experiment 
          design, ................................................. 173
10.4.3.2 Highest Paid Person Opinion - HiPPO ................ 173
10.4.3.3 Tuning experiments is repetitive and requires 
          highly qualified engineers .......................... 173
10.5 Validity Threats ......................................... 174
10.6 Conclusion ................................................. 174

11.1 Introduction .............................................. 178
11.2 Background and related work .......................... 179
11.2.1 Continuous experimentation ........................ 179
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.2.2</td>
<td>Experimentation processes</td>
<td>181</td>
</tr>
<tr>
<td>11.2.2.1</td>
<td>The Build-Measure-Learn model</td>
<td>181</td>
</tr>
<tr>
<td>11.2.2.2</td>
<td>The ESSSDM model</td>
<td>182</td>
</tr>
<tr>
<td>11.2.2.3</td>
<td>The QCD model</td>
<td>182</td>
</tr>
<tr>
<td>11.2.2.4</td>
<td>HYPEX model</td>
<td>182</td>
</tr>
<tr>
<td>11.2.2.5</td>
<td>The RIGHT model</td>
<td>182</td>
</tr>
<tr>
<td>11.3</td>
<td>Research Method</td>
<td>183</td>
</tr>
<tr>
<td>11.3.1</td>
<td>The case study</td>
<td>183</td>
</tr>
<tr>
<td>11.3.1.1</td>
<td>The case company</td>
<td>183</td>
</tr>
<tr>
<td>11.3.1.2</td>
<td>Data collection</td>
<td>184</td>
</tr>
<tr>
<td>11.3.1.3</td>
<td>Data analysis</td>
<td>186</td>
</tr>
<tr>
<td>11.3.2</td>
<td>Identification of the HURRIER process</td>
<td>187</td>
</tr>
<tr>
<td>11.3.3</td>
<td>Validity considerations</td>
<td>188</td>
</tr>
<tr>
<td>11.4</td>
<td>Continuous experimentation and practices</td>
<td>189</td>
</tr>
<tr>
<td>11.4.1</td>
<td>Types of experiments</td>
<td>189</td>
</tr>
<tr>
<td>11.4.1.1</td>
<td>Business-driven experiments</td>
<td>189</td>
</tr>
<tr>
<td>11.4.1.2</td>
<td>Regression-driven experiments</td>
<td>189</td>
</tr>
<tr>
<td>11.4.1.3</td>
<td>Optimization and tuning experiments</td>
<td>190</td>
</tr>
<tr>
<td>11.4.1.4</td>
<td>Customer support experiments</td>
<td>190</td>
</tr>
<tr>
<td>11.4.2</td>
<td>Experimentation practices and techniques</td>
<td>190</td>
</tr>
<tr>
<td>11.4.2.1</td>
<td>Experiment design and analysis</td>
<td>190</td>
</tr>
<tr>
<td>11.4.2.2</td>
<td>Variation assignment</td>
<td>192</td>
</tr>
<tr>
<td>11.4.2.3</td>
<td>Implementation techniques</td>
<td>193</td>
</tr>
<tr>
<td>11.4.2.4</td>
<td>Release techniques</td>
<td>194</td>
</tr>
<tr>
<td>11.5</td>
<td>Examples</td>
<td>195</td>
</tr>
<tr>
<td>11.5.1</td>
<td>Example A: Business-driven experiments</td>
<td>195</td>
</tr>
<tr>
<td>11.5.2</td>
<td>Example B: Regression-driven experiments</td>
<td>197</td>
</tr>
<tr>
<td>11.5.3</td>
<td>Example C: Optimization and tuning experiments</td>
<td>199</td>
</tr>
<tr>
<td>11.5.4</td>
<td>Example D: Customer support experiments</td>
<td>200</td>
</tr>
<tr>
<td>11.6</td>
<td>The HURRIER Continuous Experimentation Process</td>
<td>200</td>
</tr>
<tr>
<td>11.6.1</td>
<td>The R&amp;D organization</td>
<td>201</td>
</tr>
<tr>
<td>11.6.1.1</td>
<td>Pre-study</td>
<td>201</td>
</tr>
<tr>
<td>11.6.1.2</td>
<td>Incremental development</td>
<td>203</td>
</tr>
<tr>
<td>11.6.2</td>
<td>The internal validation</td>
<td>203</td>
</tr>
<tr>
<td>11.6.2.1</td>
<td>Continuous integration</td>
<td>204</td>
</tr>
<tr>
<td>11.6.2.2</td>
<td>Simulation</td>
<td>204</td>
</tr>
<tr>
<td>11.6.2.3</td>
<td>Internal laboratory evaluation</td>
<td>204</td>
</tr>
<tr>
<td>11.6.3</td>
<td>Single customer validation</td>
<td>204</td>
</tr>
<tr>
<td>11.6.3.1</td>
<td>Customer laboratory evaluation</td>
<td>205</td>
</tr>
<tr>
<td>11.6.3.2</td>
<td>Passive launch</td>
<td>205</td>
</tr>
<tr>
<td>11.6.3.3</td>
<td>Restricted launch</td>
<td>205</td>
</tr>
<tr>
<td>11.6.3.4</td>
<td>One customer gradual rollout</td>
<td>205</td>
</tr>
<tr>
<td>11.6.4</td>
<td>Multiple customer validation</td>
<td>206</td>
</tr>
<tr>
<td>11.6.5</td>
<td>The internal and the customer feedback channels</td>
<td>206</td>
</tr>
<tr>
<td>11.7</td>
<td>Discussion</td>
<td>207</td>
</tr>
<tr>
<td>11.7.1</td>
<td>RQ1: What are the types of experiments that are conducted in Ericsson and that are relevant in the development of mission-critical B2B systems?</td>
<td>207</td>
</tr>
</tbody>
</table>
11.7.2 RQ2: What are the current continuous experimentation practices used at Ericsson in the development of mission-critical B2B systems? 208

11.7.3 RQ3: Can the HURRIER process can be used to drive CE in mission-critical B2B systems at Ericsson? 209

11.7.4 RQ4: What are the current CE challenges and opportunities in mission-critical B2B systems observed at Ericsson? 210

11.8 Conclusion 211

12 Discussion 213

12.1 Objective 1 213

12.1.1 RQ1: What are the characteristics of an automated experimentation architecture? 214

12.1.2 RQ2: How can we utilize automated experimentation to optimize an existing software-intensive systems? 215

12.1.3 RQ3: What are the main components to run trustworthy online controlled experiments? 215

12.1.4 RQ4: How are multi-armed bandit (MAB) algorithms used in online field experiments? 216

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? 217

12.2 Objective 2 217

12.2.1 RQ6: How can the embedded systems industry adopt continuous experimentation in their development process? 218

12.2.2 RQ7: How experimentation can be conducted in mission-critical business-to-business systems? 221

13 Conclusion 223

13.1 Future work on experimentation. 224

13.1.1 Automating experiments. 224

13.1.1.1 Experimentation ontology 224

13.1.1.2 Hypotheses generation 224

13.1.1.3 Counterfactual analysis 225

13.1.2 Small-sample experiments 225

13.1.3 Longitudinal experiments 226

13.1.4 Causal inference beyond experiments 226

Bibliography 229
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:

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, $\chi$-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?
3.2. Research context

- **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 Software Program (WASP) ¹ is a research initiative that focuses 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 allow 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

¹https://wasp-sweden.org/research/
²https://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.

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

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
3.4. DATA ANALYSIS

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:

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

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

- $P(d|h)$ is called the likelihood of the data $d$ under the hypothesis $h$.
- $P(h)$ is called the prior.
- $P(h|d)$ is called the posterior. The posterior represents the probability distribution of each parameter estimate (our hypothesis $h$) given our observed data.
- $P(d)$ 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.
3.5. VALIDITY CONSIDERATIONS

[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\textsuperscript{2}-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 $\texttt{bpcs}$ 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 indicate 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/
4.2. INCLUDED PUBLICATIONS

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.
CHAPTER 4. CONTRIBUTIONS OF THIS THESIS

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.
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.


#### 4.2.7.1 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.
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, $\chi^2$-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.


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.