Department of Computer Science
Series of Publications A
Report A-2019-5

Initiating the Transition towards Continuous Experimentation: Empirical Studies with Software Development Teams and Practitioners

Sezin Gizem Yaman

Doctoral dissertation, to be presented for public examination with the permission of the Faculty of Science of the University of Helsinki, in Room 302, Athena Building, University of Helsinki, on the 25th of October, 2019 at 12 o'clock noon.

University of Helsinki
Finland

Supervisors

Tomi Männistö and Fabian Fagerholm, University of Helsinki, Finland

Pre-examiners

Hakan Erdogmus, Carnegie Mellon University Silicon Valley, United States
Eric Knauss, Chalmers University of Technology, University of Gothenburg, Sweden

Opponent

Brian Fitzgerald, University of Limerick, Ireland

Custos

Tomi Männistö, University of Helsinki, Finland

Contact information

Department of Computer Science
P.O. Box 68 (Pietari Kalmin katu 5)
FI-00014 University of Helsinki
Finland

Email address: info@cs.helsinki.fi
URL: http://cs.helsinki.fi/
Telephone: +358 2941 911

Copyright © 2019 Sezin Gizem Yaman
ISSN 1238-8645
ISBN 978-951-51-5542-9 (paperback)
ISBN 978-951-51-5543-6 (PDF)
Computing Reviews (1998) Classification: D.2.m, K.6.0, K.6.3 K.7.2, K.7.4
Helsinki 2019
Unigrafia

Initiating the Transition towards Continuous Experimentation: Empirical Studies with Software Development Teams and Practitioners

Sezin Gizem Yaman

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
Sezin.Yaman@helsinki.fi
http://www.linkedin.com/in/sezin-yaman/

PhD Dissertation, Series of Publications A, Report A-2019-5
Helsinki, September 2019, 81 + 90 pages
ISSN 1238-8645
ISBN 978-951-51-5542-9 (paperback)
ISBN 978-951-51-5543-6 (PDF)

Abstract

Software experiments are presently often used by big technology pioneers, such as Microsoft, Facebook and Google, in order to learn about their users and to guide their research and development activities. Continuous experimentation (CE) is reported to be an integral part of software development in these organisations, however, how they transitioned to the approach is not publicly shared. Therefore, there is a lack of guidance for other organisations that are willing to adopt CE. In the current competitive markets, investing time and money in a new approach might be risky for these organisations, especially if they do not know how to initiate this transition process.

This dissertation focuses on how organisations can initiate the transition towards CE, i.e., an approach to enhance development decisions by running experiments in an iterative and sustainable fashion. The dissertation was designed to acquire descriptive and observational knowledge through empirical studies and was conducted in three main phases. First, we designed and ran multiple-case studies to investigate how CE can be introduced to existing software company development teams, who want to run their first systematic experiments. We extracted descriptive knowledge from the introduction process and composed lessons learned to act as guidelines. In the second phase, we conducted a survey study with practitioners from four Nordic software companies, in order to better understand their attitudes and perception towards experiment-driven development, user involvement and ethics. Examining the results at role-to-role levels gave us an understanding of commonalities and distinctions stemming from different job functions. Furthermore, we identified patterns from the data that describe what trends exist across the dataset with respect to experiment-driven software development. Finally, in the last phase of the study, we conducted a single-case study with a mobile gaming company to investigate how CE functions as an organisational mechanism throughout the development life-cycle.

The findings show that transitioning towards CE is a learning process that can be facilitated well by guidance, utilising existing resources and starting with small experiments with potentially enormous impact. Furthermore, by investigating the point of view of practitioners, we observed that software experiments represent different concepts, for instance, A/B tests and user interviews. We also observed that the role of the practitioner has a big impact not only on how experiments are understood, but also how individuals perceive the ethics involved in the experiments. For example, while managers are more cautious about company-customer relationships, UX designers were found to allow exceptions to user notification during experiments. In addition, we discovered that companies might understand and adopt experiment-driven development differently, for instance, influenced by their business contexts. Lastly, by examining a company's CE practices, we found that experiments can take different forms given the development stage, and the organisational mechanism can be established to fit both the needs of the business domain and organisational goals. One of the biggest challenges of adopting CE, inaccessible real users, can be overcome with alternative methods, such as proxy users, especially early in the development, when experiments are important in determining product value.

Highly competitive markets can put pressure on organisations to avoid risks and costs when adopting a new approach. In this dissertation, we learned that by and large, software organisations and development teams can initiate their transition towards CE in an efficient and economical way. Furthermore, we conclude that the transition is a learning process that improves with practice and has to adapt to the organisational goals and contexts. The influence of human factors, such as the finding that individual perception of experiments and ethics is correlated with job functions indicates that CE is a multi-disciplinary research field, where individuals should be studied as well as experimentation processes. Software engineering research needs further studies to validate the findings in different contexts.

Computing Reviews (1998) Categories and Subject Descriptors:

D.2.m	[Software Engineering]: Miscellaneous
K.6.0	[Management of Computing and Information Systems]: General – economics
K.6.3	[Management of Computing and Information Systems]: Software Management – software process, software development
K.7.2	[The Computing Profession]: Organisations
K.7.4	[The Computing Profession]: Professional Ethics

General Terms:
experiment-driven software development, continuous experimentation, user involvement, software experiments, human factors, ethics

Additional Key Words and Phrases:
empirical software engineering, data-driven decision-making, data-driven software development, agile software development, controlled experiments

Acknowledgements

"...if I'd stay as I was before I was taken prisoner or go through it all again, I'd say for god's sake let me be a prisoner again. ...There is a great deal, a great deal still to come". — Pierre Bezukhov, War and Peace

My adventures in Finland started in 2011 when I came to pursue a master's degree, which extended to doctoral studies. This journey was not an easy one, and I feel tremendous gratitude for many people I have met along the way.

I want to thank my supervisor Prof. Tomi Männistö, for his honest guidance, for believing in me and my research, and for motivating me to follow exciting ideas. Thank you for the enormous assistance and encouragement you provided. I want to thank my second supervisor, Dr. Fabian Fagerholm, who has been an inspiring mentor and compassionate friend, and without whom this dissertation project would not have been possible. I learned a ton from our conversations about the philosophy of science, about human intelligence, and about what makes disciplined, capable research scientists.

I want to thank my examiners, Hakan Erdogmus and Eric Knauss, for the time they spent reviewing my research and for the insights and perspectives they conferred upon me, and which have in turn strengthened this dissertation. I want to thank Casper Lassenius, for the valuable feedback he gave in the doctoral symposiums. I also want to thank my opponent, Brian Fitzgerald, for giving me the opportunity to present my research.

I have many colleagues and friends to thank in University of Helsinki. I must first thank Prof. Tommi Mikkonen, also for believing in me, supporting and advising me especially in the later stages of my PhD. By his encouragement, I pursued an internship at Mozilla Corporation, which proved a fantastic experience. I thank Prof. Dr. Jürgen Münch, for generously supporting my research. Thanks to him, I have better understood the importance of research collaborations, and enjoyed writing articles that have high scientific impact. I want to thank my former colleagues Leah Riungu-Kalliosaari and Myriam Munezero for all their patience and support. Besides collaborating on interesting research subjects, I have learned so much from them about life and strengths of women, while sharing so many memories altogether. I thank my former colleagues and co-authors, including Petri, Simo, Hanna, A-P, Niko, Juha, Francois and Hadi. We had many scientific, intellectual and fun conversations over the last years, it made me gain new perspectives while enjoying my time at the department. I am also grateful to Tiina Väisänen and Pirjo Mulari, and the rest of the administrative team of Kumpula, for all the understanding and help given, countless times.

I would also like to thank the industry professionals I met during our research collaborations, including Mika Aaltola and Christina Palmu from Ericsson; Riku Suomela from Next Games; and David Bryant from Mozilla Corporation. I extend my thanks to the numerous practitioners in these companies who have generously shared their time and experiences. Also, I am so grateful to gain the life-long amazing friends, Nancy and John, during my internship at Mozilla. Thanks guys for making me feel like home at California. Furthermore, I also owe my thanks to Dr. Anton Antonov, whom I met at University of Oxford during my summer school and whom I learned so much about Data Science over the last two years, and hopefully will still learn a lot.

I am especially thankful to Sasu Tarkoma, Petri Myllymäki and Pirjo Moen for all the opportunities and administrative support. This work has been supported by the Department of Computer Science, University of Helsinki, DoCS (Doctoral Programme in Computer Science), HIIT (Helsinki Institute for Information Technology) and Tekes – the Finnish Funding Agency for Technology and Innovation, as part of the N4S Program of DIMECC (Finnish co-creation platform for digital transformations). Furthermore, this work was supported during 2016-2018 by HICT DoCS (Helsinki Doctoral Education Network in Information and Communications Technology) grant and by the Academy of Finland project, xCESE (project number 317657), during the last months. Finally, this work was rewarded in 2017 by the Nokia Foundation scholarship.

There are also close people who were very supportive during my PhD journey. I would like to share my warmest gratitude with Julia, Oulia and Selma, who have been also living their PhD journeys. We shared a lot altogether and had so much fun. I especially want to thank Ville for all the times and experiences we had together, in which I was encouraged to rediscover my life more honestly, while getting closer to the path I strive for. I eagerly look forward to new stories, to fill us with awe, love and joy.

Finally, I want to thank my parents, for the boundless love and patience they have shown me during my PhD journey. I want to thank my little sister, who is a big help in countless ways, wittingly and unwittingly, and from whom I draw strength.

To more adventures!

Helsinki, September 2019
Sezin Gizem Yaman

List of Original Articles

This dissertation is based on the following peer-reviewed original publications. The publications are also referred as Articles I–V in the text. Below the publication list, the contributions of the present author are described. The publications are reproduced with permission from the copyright holders at the end of the dissertation.

Article I Yaman, S. G., Munezero, M., Munch, J., Fagerholm, F., Syd, O., Aaltola, M., & Männistö, T. (2017). Introducing continuous experimentation in large software-intensive product and service organisations. In Journal of Systems and Software, JSS, 133, (pp. 195-211).

Article II Yaman, S. G., Fagerholm, F., Munezero, M., Münch, J., Aaltola, M., Palmu, C., & Männistö, T. (2016). Transitioning towards continuous experimentation in a large software product and service development organisation – a case study. In International Conference on Product-Focused Software Process Improvement (pp. 344-359). Springer, Cham.

Article III Yaman, S., Fagerholm, F., Munezero, M., Mäenpää, H., & Männistö, T. (2017). Notifying and Involving Users in Experimentation: Ethical Perceptions of Software Practitioners. ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2017) (pp. 199-204). IEEE.

Article IV	Yaman, S., Fagerholm, F., Munezero, M., Mikkonen, T., & Männistö,T. (2018). Patterns of User Involvement in Experiment-driven Software Development. In Information and Software Technology, IST, Journal. (In revision)
Article V	Yaman, S., Mikkonen, T., & Suomela, R. (2018). Continuous Experimentation in Mobile Game Development. In 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2018) (pp. 345-352). IEEE.

The author was the main contributor in all of the articles. For Articles I and II, the author carried out the study inspection, design and execution in collaboration with the rest of the authors, while contributing to major parts of the study analysis and reporting. For Articles III and IV, the author developed the initial idea and carried out the main parts of all study phases, from study design to reporting. For Article V, after the second author initiated the study, the author coordinated it and was the main contributor in study design, analysis and reporting. None of the articles have been used in previous dissertations.

Contents

Chapter 1

Introduction

Understanding user needs and behaviour and learning from them are essential parts of software development (Williams and Cockburn, 2003; Lagrosen, 2005; Laage-Hellman et al., 2014). For almost two decades now, agile software methodology has already demonstrated the benefits and acquired success from faster user feedback and user involvement in the development processes (Dingsøyr and Lassenius, 2016). Software engineering research suggests that software experiments can be used as a means for user involvement and to steer the direction of research and development (R&D) (Bosch, 2012; Olsson et al., 2012; Yaman et al., 2016). Experiments can therefore serve as an evidence-based decision-making mechanism for software development organisations.

The benefits of software experiments have been reported and acknowledged both by the academia (Olsson et al., 2012; Fagerholm et al., 2014; Fabijan et al., 2015; Madeyski and Kawalerowicz, 2017) and companies, such as Google (Tang et al., 2010), Facebook (Bakshy et al., 2014) and Microsoft (Kohavi et al., 2013; Fabijan et al., 2017a). Despite the increasing attention to the subject, there is a lack of knowledge on how software organisations transition towards experiment-driven development in their development activities. One particular question is how such a transition can be efficiently enabled for existing organisations that are willing to adopt the approach, that are at the same time constrained by the time and cost to be invested in in today's competitive markets.

This dissertation investigates how existing software development organisations initiate continuous experimentation (CE), that is, a development approach, where software experiments are used to inform R&D decisions in an iterative and sustainable fashion. This requires an understanding of organisational goals and existing resources as well as the practitioners' ongoing commitments and their perceptions on software experiments in the first place. Hence, this dissertation explores how development teams can conduct their first systematic experiments in accordance with their organisational context and goals. In addition, software experimentation requires various other considerations, such as user ethics and privacy. This dissertation also seeks to acquire software practitioner viewpoints on user data collection with respect to ethical issues. The findings of this dissertation contribute to the empirical knowledge on how development teams and software development organisations can facilitate the transition towards experiment-driven software. Eventually, CE can be achieved by establishing an organisational mechanism that captures the learning from the first systematic experiments and scales them up to build the necessarily skills and infrastructure.

1.1 Motivation

Lagrosen (2001) states that: "To give maximum value to the customers it is crucial that the companies have an in-depth understanding of what customers need and want". Agile development has been emphasizing user involvement throughout the software development life-cycle, facilitating feedback and reflection, which would lead to a better understanding of their needs and behaviours (Boehm and Turner, 2003; Dingsøyr et al., 2012). Recent research studies reveal that one of the common ways of collecting larger amount of user feedback is through experiments, where particular ideas, concepts and products can be tested with users (Bosch-Sijtsema and Bosch, 2015).

Software experiments in the form of online controlled experiments, often referred to as A/B testing, have been repeatedly reported by technology forerunners over the last two decades (Kohavi et al., 2013); for instance, Microsoft is reported to run tens of thousands of online controlled experiments a year (Fabijan et al., 2017a). However, online controlled experiments require web-facing product interfaces, where the experiments are designed and ran, which often happens at the post-deployment stage of a software product or service (Mattos et al., 2018). Furthermore, these technology forerunners do not explicitly share how they transitioned to the capability of running software experiments continuously. In other words, existing research indicates that a particular type of experimentation, online controlled experimentation, has been overwhelmingly reported, yet there exists relatively little information on how the transition to experiment-driven development has been or can be achieved.

Approaches where product decisions are guided based on experiments involving users, while varying in detail and implementation, can be termed *experiment-driven software development*. Academia has been increasingly emphasizing the need to transition toward experiment-driven development. Olsson et al. (2012) describe that the next stage of companies transitioning further from agile development towards continuous deployment involves using experimental systems in their R&D. There are several experimentation models suggested, such as the innovation experiment systems model (Bosch, 2012) and the RIGHT model (Fagerholm et al., 2017) for software development, however, these models and related research also do not provide direct insights into how to initiate the transition towards experiment-driven development and how to run the first systematic experiments.

In order to address these research gaps, this research aims to study how existing organisations could initiate the transition towards CE in a systematic way. Such a transition requires an examination of organisational goals on experiment-driven development, as well as and practitioner standpoints, since organisational changes often start at the individual level (Callan, 1993). In this dissertation, along with addressing the practitioner point of view on software experiments, we investigate how the approach can be introduced to development teams, and how CE can work as an organisational mechanism, adapting to the organisational goals and necessities of the software development life-cycle.

Furthermore, as experiments are used to gather data from software product and service users, they are subject to several other considerations, such as ethics and privacy. For instance, in 2014, Facebook ran an experiment with thousands of their users to test how emotional contagion occurs when their News Feed content is manipulated, and they found out that users' mood can be manipulated (Kramer et al., 2014). The study raised serious questions about user privacy and ethical implications, when such experiments are performed without users' informed consent (Agarwal and Dhar, 2014). A similar situation in early 2018 concerned more than 50 million Facebook users, as their information was used to build prediction software by a private company targeting voters in the US presidential election[1], indicating that such user information collected through experiments can lead to very grave implications. Motivated by these considerations, in this dissertation, we also seek to examine software practitioner attitudes and ethical views with respect to user data collection through experiments. Thereby,

[1]https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election/

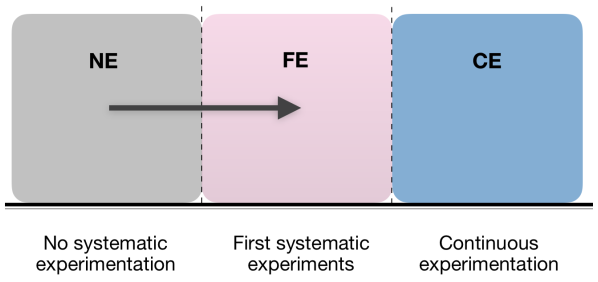

NE
FE
CE
No systematic experimentation
First systematic experiments
Continuous experimentation

Figure 1.1: The scope of this research covers the areas depicted as NE (no systematic experimentation) and FE (first experiments). In particular, the focus of this dissertation is on how the transition from NE to FE can be initiated. Area CE represents having the capability of CE.

we can better comprehend software experiments as targeted at human subjects, given that there is not much existing research.

1.2 Research Questions

CE is a development approach that requires various skills and infrastructure to run software experiments in an iterative and sustainable fashion. Therefore, time and investment is expected from organisations to build such capability. Figure 1.1 above portrays the scope of this dissertation. Area NE represents the state in which the software teams in the organisations do not have prior experience with systematic experimentation. Area FE represents the state in which the transition to continuous experimentation has begun, and the first systematic experiments are conducted. In particular, the scope of this dissertation is to study the transition from NE to FE. Once the learning from the first systematic experiments is captured, and the CE approach becomes an integral part of the development activities, CE, continuous experimentation, can be achieved. We consider the aforementioned technology forerunners (e.g., Microsoft, Facebook and Google) to reside in the CE area.

In order to study how software development organisations can transition towards CE, first, we seek to observe how software development teams can run their first systematic experiments, given that the teams do not have prior experience with CE. In particular, we are interested in understanding how CE can be introduced to the software development teams systematically to start with. Furthermore, we aim to study the common steps that could be taken to run the first systematic experiments with software development teams; conversely, we also aim to identify the dissimilarities emerging from different contexts. To examine this, we form the first research question as follows:

RQ1: How can continuous experimentation be introduced in software development teams?

Second, we aim to gain a deeper understanding of experiment-driven development and user involvement in experiments from the software practitioner point of views. We intend to investigate existing practitioner development activities with respect to experiment-driven software development. The current state of development and user involvement of software companies can guide how the transition towards CE could be enabled. In particular, existing resources, the tools and methods that practitioners prefer to use in their development activities and how user involvement and ethics in experimentation perceived by the practitioners can provide further insights into understanding the transition. Motivated by these, we ask the second research question:

RQ2: What perspectives and attitudes do practitioners at organisations exhibit with respect to experiment-driven development?

Third, we propose to enquire into how the practice of CE can extend over an organisation and adapt to its context and goals. The necessities of a business domain, organisational needs and goals to adopt the experiment-driven development and typical life-cycle of product development might affect how CE is practised. This lead us to form the following question:

RQ3: How can a continuous experimentation mechanism be established in an organisation, adapting to the business domain and organisational goals?

With these three research questions, we study how existing organisations can initiate the transition towards CE. In particular, RQ1 looks at the first steps that can be taken to conduct systematic experiments with development teams, while RQ2 to aims at getting a deeper understanding of the practitioner points of view in the approach in general. Furthermore, RQ3 intends to comprehend how CE can function as an organisational mechanism, that is, in line with the organisational goals and the needs of the business domain.

1.3 Overview of the Research Process

In order to seek answers to our research questions, we designed the study in three research phases as depicted in Figure 1.2. In the first phase of the study, we sought answers for RQ1, which concerned the introduction of CE to software development teams. In this phase, we designed and conducted multiple-case studies in two software-intensive development organisations. We initiated the CE introduction process by running the first systematic experiments with the development teams. We analysed the results and obtained descriptive knowledge on the introduction of CE, including the challenges faced and the mitigation strategies. The findings were reported in Articles I and II.

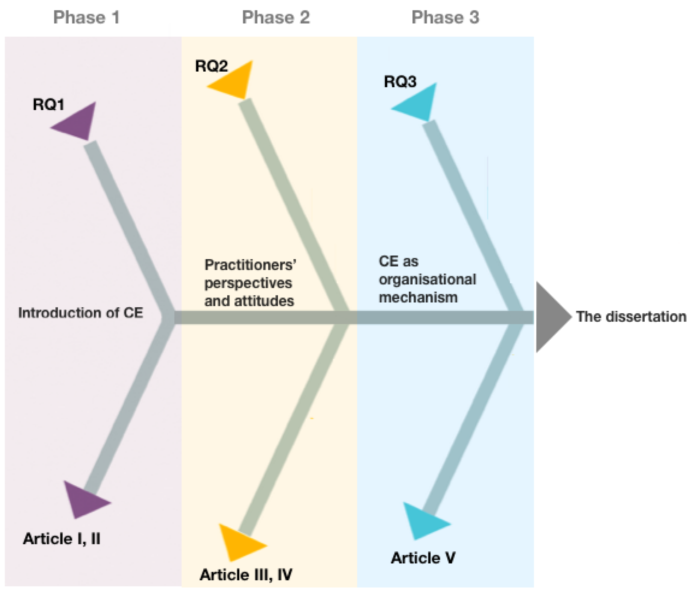


Figure 1.2: The relationship between the research questions and the publications as well as the study focus of each phase.

Furthermore, Phase 2 was designed to address RQ2. In this phase, we designed and ran a questionnaire to examine how practitioners at software development organisations understand and perceive experiment-driven development and user involvement. Both quantitative and qualitative data analysis results revealed research findings obtained from different practitioner roles. Practitioners' perception of ethics was separately reported in Article III, and full survey results

Table 1.1: The relationship between the original articles and the research questions. (X indicates that the article addresses the research question.)

	RQ1	RQ2	RQ3
Article I	X		
Article II	X		
Article III		X	
Article IV		X	
Article V			X

were reported in Article IV. Lastly, in order to address RQ3, we designed Phase 3 of the study, in which we conducted a single-case study with a mobile gaming company, where the CE was an already established part of the software development process. We observed their typical development activities with respect to software experiments and deducted knowledge on how the contextual factors, such as the business domain and organisational goals, affect CE. The findings were reported in Article V.

Table 1.1 further summarises the relationship between the research questions and the publications in which they were addressed, in a tabular form. RQ1 was addressed in Articles I and II, RQ2 was addressed in Articles III and IV, and RQ3 was addressed in Article V.

1.4 Contributions

Each article provided several novel contributions to software engineering research, which can be summarised as follows:

- *Initiating the transition systematically.* Developing the continuous experimentation capability in software development organisations is a learning process, which can be supported by a systematic introduction approach, utilising existing resources. We studied the CE introduction process with development teams from two organisations and captured the common and distinctive steps taken as well as challenges faced. Our findings offer insights for development teams that want to conduct experiments systematically.

- *Understanding existing resources and capabilities.* While studying the adoption of the CE approach, it is important to understand the current resources and capabilities available to experiment with users in the organisations as well as the attitudes with respect to experiments in general. For this goal, we designed and ran a survey, inviting practitioners in three major functional roles: developers, managers and UX designers. We found prominent differences in how software experiments are understood and practised by these roles. Furthermore, we inferred six different patterns in the totality of the responses that describe experimentation and user involvement. Our patterns can be used by practitioners to reveal existing trends in their organisation.

- *Understanding the ethics of experimentation.* We put special emphasis on understanding practitioner perceptions and the attitudes towards ethical issues that must be taken into account when practising experimentation with users as a development strategy. With the survey study, we investigated how practitioners experience the need to notify users when involving them in experimentation as well as the trustworthiness and resources required for experiments. We found out that experiences and daily activities associated with different roles have direct influence on how ethical issues are perceived. Organisations need to address such differences in ethical issues, especially if there is no common understanding on user consent, data collection and processing.

- *CE as an organisational mechanism.* In order to adopt CE, the business context has to be considered, along with the organisational goals. We learned from our case study with a mobile game development company that CE works as an organisational mechanism, adapting to the needs of each development stage and the objectives of the game and closely following the competitive game market. In the case study with the telecommunication network company, assessing the multi-layered structure of users and customers led us carefully target the experimentation on a specific user group. The business context influences both the technicality and the cost of the first experiments. Setting organisational goals to adopt the CE approach that fits the business context is one of the keys to success.

1.5 Dissertation Structure

The rest of the dissertation is structured as follows: In Chapter 2, we look at the previous work from different perspectives, including user involvement in software engineering research and practice, the history of software experiments and how the body of knowledge accumulated over time, existing experimentation systems, models and human factors involved in experiments, such as ethics. In this chapter, we also set the terminology involved in this research. Following that, we elaborate on the research design and outline the rationale behind the different research methods involved in this dissertation, including case studies and the survey study in Chapter 3. Next, in Chapter 4, we move to overview the articles involved in this dissertation and draw a picture of how they are connected in addressing the research questions posited earlier in the dissertation. The dissertation comes to an end with an extensive discussion on the research findings and both scientific and practical contributions in Chapter 5. Furthermore, we also propose open questions, which will hopefully lead to future work, followed by the Conclusion chapter of the dissertation (Chapter 6).

Chapter 2

Previous Work

Software companies seek efficient methods to assess and evaluate user value and the success of their products (Tichy et al., 2015). Collecting product usage data and user feedback to continuously guide decision-making processes is a practice that has gained increasing attention especially in the last decade (Fabijan et al., 2017a). Big technology forerunners like Microsoft (Fabijan et al., 2017a), Google (Tang et al., 2010) and many others are reported to collect data from their users en-masse and are known to run hundreds of experiments to guide R&D decision-making (Kohavi and Longbotham, 2017). With continuous experimentation (CE), companies can evaluate and prioritise their development decisions, even before fully implementing features in their software product or services. Therefore, the cost of implementation would be avoided in case of features or products having no confirmed value (Mattos et al., 2018).

The benefits of experimentation are recognised both by academia and software industry research, however, there is a lack of knowledge on how software organisations can adopt the approach. In fact, Madeyski and Kawalerowicz (2017) argue that few companies are ready to adopt software experimentation, since experimentation is often perceived as difficult to conduct and expensive in terms of both money and time. In addition, we argue that software organisations might not know how to effectively transition towards experiment-driven development in such competitive markets, given the budget and time constraints.

With software experiments, users can be involved in shaping the software product and services by being experiment subjects. Involving users in software development has a long history and has emerged in various sub-disciplines of software engineering. For instance, participatory design has been an important field of research, where software users make effective contributions to reflect their needs and perspectives (Muller et al., 1997). In recent years, software experimentation has been studied as a means for user involvement (Bosch-Sijtsema and Bosch, 2015) and R&D – as an experimentation system have been foreseen as a new stage of agile development (Olsson et al., 2012). Besides, several experimentation models were proposed, and the elements of experiments were studied. Related work on these research sub-disciplines and existing body of knowledge help us better understand the CE approach and its adoption.

In this chapter, we will first look at the general user involvement research in software development and how software experiments facilitate user involvement (Section 2.1). Next, in Section 2.2, we look at the history of software experimentation and how it has evolved along with the technological advancements. In Section 2.3, we focus on existing experimentation models and systems that are suggested by previous academic and industry research and investigate how the CE approach has been defined and what it represents. Following that, in Section 2.4, we look at the body of knowledge on user data collection and ethics. We pay especially attention to the ethical issues involved in user data collection via experiments, as it is a prevalent topic for both technology providers and its users. Lastly, in Section 2.5, we set the terminology that is used in this dissertation.

2.1 User Involvement in Software Development

Organisations have to develop ways to understand users and learn about them to survive and compete in today's fast-changing software development environments (Tichy et al., 2015). It is important to reach out to users and learn from them, while utilising rapid and iterative development cycles rather than waterfall-style interactions between stages such as product development followed by customer feedback processes (Olsson et al., 2012; Fabijan, 2016; Ebert et al., 2016). Many companies transitioned towards agile development, which champions speed and flexibility in development and highlights the importance of customer feedback (Highsmith and Cockburn, 2001). Furthermore, other approaches, such as Lean Startup (Ries, 2011), Customer Development (Blank, 2013), Lean Analytics (Croll and Yoskovitz, 2013) and DevOps (Bass et al., 2015), put emphasis on collecting data from users and continuously learning from them in rapid cycles. Reviewing software engineering research, where users are involved in the development, can help understand how CE facilitates user involvement. User involvement has been widely studied in software engineering research in various approaches and research sub-disciplines such as participatory design, user-centric design, usability engineering, human computer interaction, requirements engineering and information systems (Abelein and Paech, 2015; Fabijan et al., 2015). For instance, in participatory design, direct and active involvement of people in co-design is emphasized, having people actually using the software product most of the time (Hess et al., 2008). While the emphasis in participatory design is on democratic participation and skill enhancement, usability engineering is the process of defining, measuring and improving the usability of products, and it overlaps with user-centric design principles (Kujala, 2003). Gould and Lewis (1985) proposed three main principles of user-centric design in the 80's as: *1)* early focus on users and tasks, *2)* empirical measurement, and *3)* iterative design. Similarly, human computer interaction seeks to improve the usability of human-computer interfaces (Grudin, 1992). These principles are widely followed and have also inspired other development approaches. They could be used as instantiations of or parts of an experiment-driven software engineering approach.

Furthermore, requirements engineering research aims at maximizing the value of the release of software, while accommodating a collaborative approach with users in order to involve their perspectives throughout the development (Barney et al., 2008); therefore, user requirements could be understood better. Information systems research also has a long history with the focus of user involvement and its effects on system success with insights from organisational and social contexts that influence user behaviour (Franz and Robey, 1986; Henfridsson and Lindgren, 2010). Iivari (2004) likewise emphasizes that all user involvement approaches in software engineering research aims at a better understanding of the user by involving them in software development; however, the approaches might differ in *why users should be involved*, *how to involve them* and *how to facilitate the involvement*.

User involvement can denote a broad understanding from active user participation, as being common in participatory design and requirement engineering, to the involvement of the users as information providers through observation, as being more common in fields such as user-centric design and usability engineering (Iivari, 2004). Bosch-Sijtsema and Bosch (2015) define two main categories of user involvement based on user data collection methods used in their studies: *1)* conscious input, and *2)* uninformed user input. In the former category, users are actively and consciously involved in providing input. Interviews, surveys, ethnographic studies are some examples of this category. In the latter category, users can be unaware of an ongoing study to test a feature or a product. Tests and experiments, where user behaviour is observed and data is collected, are some examples of this category. However, some data collection methods, such as simulations and ethnographic studies, can be used to gather both conscious user input and uninformed input based on observation. Bosch-Sijtsema and Bosch (2015) made their user input classification based on their study on data-driven innovation processes in software and software-intensive companies, and the findings provide insights into user involvement in experimentation, which we will further discuss in Section 2.4.

Especially with the transition towards agile software development, intensive and direct user communication has been encouraged, and new concepts, such as prototyping, have gained increased attention for involving users in software development (Kautz, 2010). Agile development can be seen as a reformulation of iterative and incremental software development (Larman and Basili, 2003) with an added emphasis on user involvement. User involvement and value creation were also emphasized with the introduction of Lean software development (Poppendieck and Poppendieck, 2003) that also highlights the importance of eliminating waste. Furthermore, Ries (2011) introduces the 'Build-Measure-Learn' (BML) methodology in his Lean Startup approach where user behaviour is measured using an experimental instrument, and decisions are made based on the learning. The BML loop lies at the core of various experimentation models (see Section 2.3).

Finally, the means of involving users in software development processes has been transforming in attempts to adapt to new development approaches. In recent years, experimentation has been reported to be commonly used to collect user data and feedback. For instance, in their systematic literature review on customer feedback and data collection techniques, Fabijan et al. (2015) report that experiments, such as A/B tests, are a common data collection technique, especially in the Web 2.0 domain and software-as-a-service (SaaS). Yaman et al. (2016) also report in their literature review that experiments and tests are one of the most frequently used ways of user data collection methods, especially since 2009. They also remark that the collected data is most often utilised for decision-making. It is important to remember that software experiments have a long history, which we will look into in the next section; however, experiments being used as a novel means for user involvement in the recent development approaches is becoming increasingly popular, and consequently, the research available is only increasing.

2.2 Software Experimentation

Shadish et al. (2002) say that experiments have strength in providing causal description, that is “describing the consequences attributable to deliberately varying a treatment”. The desire for causal explanation motivates researchers to investigate cause-effect relationships in software engineering research. Hannay et al. (2007) emphasize that in software engineering, exploring these relationships with experiments leads to the development of scientific theories, which is the foundation to science. However, there is a difference between experimentation in software engineering research and experimentation in software engineering in practice.

The software engineering community has emphasized the need for empirical studies to develop or improve processes, methods and tools for software development and maintenance since the 70’s (Basili, 1996; Tichy, 1998; Basili et al., 2002; Kitchenham et al., 2002). Kitchenham et al. (2004) coined the term *evidence-based software engineering* and offered the following definition: “to provide the means by which current best evidence from research can be integrated with practical experience and human values in the decision-making process regarding the development and maintenance of software”. They use an analogy from the field of medicine and how an evidence-based approach could be adapted by software engineering research, however, they also note that software engineering can benefit from this approach with the condition that problems should arise from the nature of software engineering.

Furthermore, Sjøberg et al. (2005), in their survey study on experimentation in software engineering, reveal that the term *experiment* has been inconsistently used by the software engineering community, and it was often used to refer to empirical studies in general. They add that they prefer to use the term *controlled experiments* and offer a definition for it in software engineering: “A randomized experiment or a quasi-experiment in which individuals or teams (the experimental units) conduct one or more software engineering tasks for the sake of comparing different populations, processes, methods, techniques, languages, or tools (the treatment)”. In addition, they note that random experiments, that is, where units are assigned to receive a treatment based on a random process, are not always feasible in the software engineering field; instead, quasi-experiments could be more feasible, where units are not assigned randomly. It is also remarked that only 1.9% of software engineering research has reported uses of controlled experiments from 1993 and 2002, and this could be due to the large effort and resources needed to run big-scale and well-designed experiments.

Evidence-based software engineering and empirical software engineering research aim to answer questions such as what development *technology and skill* (e.g., process, technique, language or tool) are to be used or implemented, which *tasks* are to be carried out in which *environments* by comparing and understanding their effects in software development (Sjøberg et al., 2005; Hannay et al., 2007). For this, experimentation can be used as a research method to measure such effects, and scientific theories can further be developed (Hannay et al., 2007).

While the evidence-based software engineering and empirical software engineering fields have been contributing to the body of science, with the technological developments during the last decades, such as web sites, applications and continuous deployment, software experimentation has gained new forms. For instance, with the development of Web 2.0, social network systems and SaaS, companies have begun to collect more and frequent data from their users (Bosch-Sijtsema and Bosch, 2015). Large volumes of data have become available to be applied in personalisation, system improvement, site modification and business intelligence (Srivastava et al., 2000). The adoption of continuous practices, such as continuous integration and continuous deployment, and DevOps, helped companies deliver high-quality of software at an accelerated pace, while getting more and quick feedback from the software development process and users (Bass et al., 2015; Shahin et al., 2017). These changes facilitated the implementation of experimentation platforms for companies to frequently test their development efforts even before fully implementing their products, in order to decrease their R&D costs. This novel form of software experimentation started to be used in the form of controlled experiments, primarily in the Web 2.0 and SaaS fields and often for feature optimization (Bosch-Sijtsema and Bosch, 2015).

In particular, controlled experiments, mostly in the form of A/B tests, are used by web-facing companies often in the post-deployment stage of the development to improve their systems continuously (Mattos et al., 2018). These types of experiments help companies establish a causal relationship between a variation, such as a new feature or a change, in their system and the observed user behaviour (Kohavi et al., 2009). They can be used to perform simple online tests, where different versions of the software are created, for instance, an old version (A – baseline/control group) and a new version with a slightly different layout or colours of a button (B – with intervention/treatment), and deployed to a different set of users (Madeyski and Kawalerowicz, 2017). Following that, user behaviour is recorded, such as counting user clicks, in each group. The aim of this randomized controlled experiment is to detect statistically significant differences between control and treatment groups, which indicates which version of the software is favourable to users (Madeyski and Kawalerowicz, 2017). Consequently, a development decision can be made following the experiment results, for instance, fully implementing and deploying the more favourable software to whole set of users. This approach has already been practiced by numerous companies. Some of them, like Ebay (Davenport, 2009), Google (Eisenberg and Quarto-vonTivadar, 2009; Tang et al., 2010) and Microsoft (Kohavi et al., 2009; Kohavi and Longbotham, 2017), are even known to establish their own experimentation platforms and teams, and they continuously report on their experimentation efforts.

On the other hand, Lindgren and Münch (2015) report that despite the increasing interest in experimentation as a part of software development activities, most of the research in the field mainly focuses on one specific type of experimentation by eminent web-facing companies – *online controlled experimentation.* Despite its popularity among web-facing companies (Fabijan et al., 2017a; Mattos et al., 2018), online controlled experiments are often limited to optimizing narrow aspects of web interfaces (Bosch-Sijtsema and Bosch, 2015), that often takes place in the post-deployment stage of the product life-cycle. We can argue that in the earlier stages of software development, for instance, when developing a new product or feature, there might not be a web-facing environment to run experiments. For this reason, in this research, we are exploring software experiments taking place not only in the post-deployment stage of development life-cycle but overall.

There have also been attempts to enable experimentation in the early stages of software development, especially to test business ideas, models and concepts early on. Companies seek new behaviours to involve users in the development process at the business level even before a product concept is determined (Bosch-Sijtsema and Bosch, 2015). Thomke (2001) emphasizes that gaining early information on products can significantly influence decisions in the early stages of product development. He further lists several rules of experimentation for companies to innovate, the first three of which are: "organize rapid experiments", "fail early and often, but avoid mistakes" and "anticipate and exploit early information". Building the capacity for rapid experimentation in early development means early feedback on the product and user; and utilising learning from rapid experiments can guide early decision-making. Manzi (2012) states that "at Google, only about 10 percent of these [controlled experiments] led to business changes" and Kohavi and Longbotham (2017) state that "at Microsoft, only one third of the experimented ideas were observed to be successful". This might indicate that software experiments do not always make a visible change in business; however, learning will be captured from eliminating bad ideas, and therefore, better experiments can be designed in the future. Nevertheless, this might pose another challenge for the companies that want to design and run their first experiments, as they might not observe the impact of the experiments in their business right away, despite the time and effort put into adopting the approach. Therefore, it is crucial to know how to initiate the transition in an efficient and economical way.

In fact, Madeyski and Kawalerowicz (2017) in their research also address that the professionals at software organisations were observed to be hesitant about taking part in experiments due to an ambivalence. On one hand, they believe that the experimentation is beneficial; however, they fear that the costs of running big-scale and well-designed experiments in terms of time and money could be high. Furthermore, we can argue that existing companies that desire to transition towards experiment-driven development might share similar concerns, especially considering that the tools and skills required for experimentation, such as infrastructure and experience, might not exist yet. Besides, the costs in such investments in highly competitive markets can be very high, if the transition cannot be achieved effectively.

To summarise, experimentation in the field of software engineering is not new and has been investigated for decades as a research method that contributes to the body of science. Software experiments in practice, however, have changed its form, adapting to the newest technological advances, methods and tools. Consequently, rather than being driven by theoretical concerns, they are driven by business goals and aim at answering business questions pragmatically. While reviewing how software experiments in practice have gained new forms, we also identified research gaps and raised arguments concerning the focus of this dissertation. Firstly, we observed that existing software experimentation research heavily reports on a specific type of experiment – online controlled experimentation. In this dissertation, we investigate how organisations can transition to the capability of conducting software experiments when or wherever needed. Secondly, existing research revealed that practitioners might face ambivalence, where on one side, there are benefits of a new approach, but on the other – constraints of time and budget needed to adopt the approach as well as inexperience. This ambivalence is also relevant to this research, as we aim at studying how development teams can conduct their first systematic experiments in an efficient way. Next, we will look at the existing experimentation models and the CE method in more in detail.

2.3 Existing Experimentation Models and Continuous Experimentation

Experimentation models, by and large, have been developed with the aim of capturing and aiding the software experimentation process. Several experimentation models were proposed in the last decade, both by researchers and practitioners. Despite the differences, we observed that most of the experimentation models and systems utilise the 'Build-Measure-Learn' (BML) loops as a means of continuous learning from users through experiments.

To start with, Ries (2011) introduces the BML methodology in his Lean Startup approach, in which BML loops are used to test product assumptions and hypotheses iteratively and to gather user feedback. A BML loop starts by forming one or more falsifiable hypotheses that need to be tested. While the build step focuses on creating a minimum viable product (MVP) to be used for data collection, the measure step focuses on using MVP to collect data. Once the collected data is analysed, a decision can be made to move to a new stage, such as pivot or persevere. Even though the Lean Startup approach does not exclusively address software experiments, it discusses the benefits and influences of experimentation with respect to business considerations, especially geared towards startup companies where the loop can be quite fast. Nevertheless, the Lean Startup approach inspired many researchers and practitioners, and the BML loop and its iterative fashion underlies several experimentation models.

With many organisations moving toward agile development and adopting capabilities for rapid and continuous value delivery to users, Olsson et al. (2012) emphasize that software R&D activities should be experimenting and testing what the customer needs. They suggest that the application of agile methods within an R&D organisation is only one stage on the maturation path, called the Stairway to Heaven model, in which the final stage is R&D as an experimentation system. In that final stage, software development is guided by iteratively conducted experiments and instant user feedback. Bosch (2012) also investigates the experimentation systems and emphasizes the importance of testing product ideas via experiments and learning from users. The Innovation Experiment Systems (IES) model, suggested by Bosch (2012), describes the process in which a hypothesis is formed based on the business need that should be tested against pre-defined metrics.

The Early-Stage Software Startup Development (ESSSD) model further extends the aforementioned models and offers operational and decision-making support on how to move to the next idea to experiment with (Bosch et al., 2013). The Hypothesis Experiment Data-Driven Development (HYPEX) model is proposed by the same authors, Olsson and Bosch (2014), as an alternative development process model to aid companies in shortening their user feedback loops through experimentation with minimum viable features (MVFs).

Similarly, Fagerholm et al. (2014, 2017) propose the RIGHT model for CE, which utilises the BML loop. In the RIGHT model, the BLM blocks are repeated over time and are supported by a technical infrastructure. Within each block, testable assumptions are derived from business strategy, experiments are run with MVFs or MVPs, and the experiment outcomes are used to guide business and product strategy. Differently from other experimentation models, Fagerholm et al. (2017) also propose descriptions for roles, tasks and artefacts involved in CE. Fagerholm et al. (2014) offer a definition for the term of CE: "a software development approach that is based on field experiments with relevant stakeholders, typically customers or users, but potentially also with other stakeholders such as investors, third-party developers, or software ecosystem partners". The same authors later extend this definition of CE as "a systematic approach to experiment-driven, continuous software engineering" (Fagerholm et al., 2017). Complementing the Stairway to Heaven model (Olsson et al., 2012), which suggests that software development should ultimately be driven by continuous and real-time customer feedback as an R&D system, Fagerholm et al. (2014) emphasize the need for continuous observation of user behaviour through field experiments that are derived from business strategies and finding out what users want. Several other research works build on the initial definition of CE such as Rissanen and Münch (2015) who focus on CE in the business-to-business (B2B) domain, and Lindgren and Münch (2015), who presents the results of a survey study on state-of-the-practice. A remarkable result from the Lindgren and Münch (2015) study indicates that despite the wealth of techniques, software organisations use to collect customer feedback, systematic CE with users was found to be rare.

A recent mapping study by Ros and Runeson (2018) also provides a definition for CE – "conducting experiments in iterations" and add that it is "a general term for a wide variety of experiments and the implications of experiments on the whole software engineering process". This mapping study shares several significant findings that are in line with the problem statement of this dissertation. Firstly, they find that CE research is dominated by big companies, such as Microsoft, Google and Facebook, and approximately 70% of the reported work belongs to the large companies. Furthermore, they share the finding that while 70% of the reported experiments are about visual changes on online UI components, only 25% of the experiments are about algorithmic changes, such as search engine ranking, 5% of the experiments report on features or new functionality. Even though Ros and Runeson (2018) do not examine the mapping study results with respect to the type of experiments, we might deduce that a majority of reported experiments are in the form of online controlled experiments, since they are conducted on online UI components. This is also in line with our arguments posed in Section 2.2.

In order to better capture how the key related work provides guidance on the research focus of this dissertation, Table 2.1 summarises the themes that were addressed by the related work that reports on software experiments that have taken place in an iterative fashion. The themes are formed based on aspects of this research, i.e., *transition to continuous experimentation* in an organisation, the software *practitioners' standpoint* and *ethical aspects* of experimentation. In addition, other prominent themes are identified in the key related work, i.e., experimentation with respect to *business aspects*, *(online) controlled experiments* and experimentation *infrastructure* and added to the table.

Transitioning to experiment-driven development or to the CE approach is not investigated by the related work directly, yet a few publications, such as that by Olsson et al. (2012), emphasize the need for transitioning towards experimentation systems, and Madeyski and Kawalerowicz (2017) build a manifesto on agile experimentation that mostly takes the form of A/B tests. Likewise, Fabijan et al. (2017b) report on how a company can evolve towards a data-driven company and describe different phases of the evolution, including technical, organisational and business evolutions. The business aspects of experimentation, on the other hand, such as experimentation being used to prioritise business decisions, were addressed by almost all the publications, except that of Tang et al. (2010), which only concentrates on controlled experiments and technical infrastructure.

Online controlled experiments, as described in Section 2.2 in detail, have increasingly been reported recently, often along with either the business aspects involved, underlying infrastructure, or both. While some of the related work specifically uses the term online controlled experiments, the others only use controlled experiments. This kind of experiments was referred to have mostly taken place at the post-deployment stage of web-facing products, while in the studies by Lindgren and Münch (2015), Rissanen and Münch (2015) and Gutbrod et al. (2017), the experiments were described to have a bigger scope and take different forms, such as interviews, MVPs and A/B tests, at various stages of software development life-cycle.

Table 2.1: The themes that are key aspects of this dissertation and that were commonly addressed by the key previous work relevant to this research, in the field of software engineering (i.e., software experiments that have taken place in an iterative and continuous fashion). (Note: We grouped the same authors' following works together. X indicates that the theme was addressed by the publication, O indicates somewhat addressed, – indicates it was not addressed, * is a mapping study.)

Publication	**Transition to CE**	**Business aspects**	**(Online) controlled experiments**	**Infras-tructure**	**Practitioner standpoint**	**Ethical aspects**
(Tang et al., 2010)	–	–	X	X	–	–
(Ries, 2011)	O	X	–	–	–	–
(Kohavi et al., 2013)	–	X	X	X	–	–
(Olsson et al., 2012), (Bosch et al., 2013), (Olsson and Bosch, 2014)	O	X	–	X	–	–
(Fagerholm et al., 2014), (Fagerholm et al., 2017)	–	X	–	X	O	–
(Bosch-Sijtsema and Bosch, 2015)	–	X	X	–	–	–
(Lindgren and Münch, 2015)	O	X	O	–	–	–
(Rissanen and Münch, 2015)	–	X	O	–	–	–
(Fabijan et al., 2017a), (Fabijan et al., 2017b)	O	X	X	X	O	–
(Kohavi and Longbotham, 2017)	O	X	X	X	–	–
(Gutbrod et al., 2017)	–	X	O	O	–	–
(Madeyski and Kawalerowicz, 2017)	O	X	X	X	–	–
(Mattos et al., 2018)	–	X	X	X	O	X
(Ros and Runeson, 2018)*	O	X	X	X	–	X

While business considerations were referred to by almost all the publications, the role of practitioners in software experimentation was partially addressed by three publications in terms of roles required in experiments such as data analysts and data scientists. However, these publications do not offer a detailed examination of the roles, the required skills and perspectives with respect to CE. Likewise, ethical issues were only addressed by two works. While Mattos et al. (2018) addressed these issues such as how users involved in experiments are guaranteed that their data would not be used for other purposes, in their mapping study, Ros and Runeson (2018) draw attention to the ethics of experimentation and the need to investigate the topic further. Moreover, Mattos et al. (2018) reported on organisational challenges involved in experimentation in embedded systems and also mentioned the skills required.

2.4 Data Collection and Ethics in Software Experimentation

CE is an empirical software development approach that involves experiments targeted at users of software products, who are human subjects, and collecting data from them. Therefore, ethical issues and principles have to be taken into account in the design, execution and analysis of experiments as well as involving users in the experiments. Data collection is subject to several considerations such as data storage, data sharing, informed consent and protection of privacy. For instance, whether the user is notified of an experiment upfront or whether personal information is collected through experiments are important issues to clarify. Understanding different data types, such as personal or identifiable data, and data collection methods, such as conscious or uninformed user input, and reviewing existing regulations and guidelines can help gain insight into the ethics of software experiments.

As we reviewed in the previous sections, in online controlled experiments, the user is typically unaware of their involvement (Bosch-Sijtsema and Bosch, 2015), which is open to debate in terms of privacy and ethical issues. Zhang et al. (2014) state that concerning the personalized recommender systems, there are two main types of user input: *explicit*, which is the direct user input, and *implicit behavioural*, which is the information the user unconsciously left while using the product. Similar classification was done by Bosch-Sijtsema and Bosch (2015) to be *conscious* vs. *uninformed* and also by other related works such as that by Unni and Harmon (2007) to be *push* vs. *pull*. In the case of explicit or conscious input, users are aware and willing to provide their input, such as participating in a user study. On the other hand, implicit or uninformed user input indicates that the behaviour of a user using a product or feature, such as user clicks, is traced and stored for further analysis without active user awareness at the moment of data collection. The latter is a well-known case with online controlled experiments. As more and more organisations are collecting this kind of behavioural data, discussions are arising on privacy and ethics (Bosch-Sijtsema and Bosch, 2015), since it might seem intrusive to the users that their data is tracked without their consent (Zhang et al., 2014).

Zhang et al. (2014) further explain that the idea of privacy is often associated with a sense of control over one's personal information with the instinct of protection. Conscious user input facilitates user control, and uninformed user input may lead to a sense of intrusiveness about their privacy. Furthermore, their study finds that user privacy concerns over their data is a multi-dimensional concept. In addition to the two main types of user input we mentioned above (conscious vs. uninformed), the degree to which the data is identifiable also affects privacy concerns (Zhang et al., 2014). For instance, the social security number of a user is identifiable data, whether it is given consciously or not, whereas age can be unidentifiable data. Therefore, Zhang et al. (2014) suggest system designers should consider these different levels of sensitivity levels such as identifiable vs. unidentifiable.

Mattos et al. (2018) also mention privacy concerns regarding user data collection through experiments and consequent data analysis. They remark that even though data sensitivity and utilisation of data might be different in different organisations and countries, data collection should be aligned with the legal requirements and user consent. Regulations, such as the European General Data Protection Regulation (GDPR),[1] clearly states that protection of personal data is a fundamental right of individuals, therefore, processing personal data is restricted. There are rules about how to process such data, and if they are not followed, fines apply. Article 6 in GDPR talks about the concept of lawfulness, that there should be acceptable grounds for processing user data, and it lists the possible grounds, such as if data processing is necessary for compliance with a legal obligation, or if data processing serves the public interest. However, for sensitive data, such as racial, political or health-related data, only one ground is acceptable: the user has to give explicit consent (Article 9). Furthermore, Article 17 details the rights to erasure, in other words, the right to be forgotten,

[1]https://www.eugdpr.org/

and Article 21 details the right to object, that is, even if consent is given by a subject, it can be taken back.

The mapping study by Ros and Runeson (2018) about CE and A/B tests also addresses the ethics involved in software experiments and poses that ethical experimentation should not be a fringe topic after GDPR regulations. Moreover, they report that only one paper, that is, Article III of this dissertation, was found to be studying the ethics of experiments during the last decade and draw attention to how the software engineering community should prioritise ethics in experimentation. For instance, there are various considerations in the context of experimentation with respect to user data collection, such as different levels of personal data, data storage, data reuse and the capabilities of erasing the data upon request. Thus, clear guidelines should be defined in compliance with the existing legislation.

Practitioners could potentially turn to the scientific literature or tradition for guidance on ethical concerns about software experimentation. There have been several proposed guidelines for involving human subjects in empirical studies in general, those that are not specific to the software engineering domain. For instance, Vinson and Singer (2008) identified four key principles for conducting empirical studies involving human subjects that could act as fundamental guidelines in software experiments: *1)* Subjects must give informed consent to their participation; this implicitly includes the requirement of notifying users to allow them to give consent. *2)* Before conducting experiments, it is important to assess whether the benefits outweigh the harm, risks and efforts, and whether the user data obtained will really be trustworthy, whether the experiment results can be used for decision-making, and whether the time spent on experiments is worth spending. *3)* Experimenters must take all possible measures to maintain confidentiality. *4)* The experiment should have value in order to motivate subjects to expose themselves to the risks. These guidelines might be useful on a general level, yet they have to be made more specific for user data collection in software experimentation in order to avoid events of failure, where the concerns about user privacy and their trust are on the line as demonstrated by discussions about Facebook and Google's user data collection[2]. These companies have recently been in the public eye for keeping track of their users' data and using it for revenue purposes, not necessarily with their users' informed consent. Given the importance of the topic and in order to address this research gap, Article III of this

[2]https://www.theguardian.com/commentisfree/2018/mar/28/all-the-data-facebook-google-has-on-you-privacy

dissertation specifically focused on the ethical issues involved in experimentation (see Section 4.2).

2.5 Terminology

In Section 2.2, we addressed the inconsistent use of the term *experiment* in software engineering research and practice. While some research uses the term to refer to empirical studies in general, others use it to refer to the research method or as a specific type of experiment. Likewise, several other central terms in the present topic area have been used differently by previous work. In order to clarify how the terms are used in this dissertation, Table 2.2 elements of CE and other related terms including: *continuous experimentation (CE), user, customer* and *user involvement.*

Table 2.2: The terms and elements of CE and their descriptions as used in this dissertation. (Adapted from Article I.)

The term	**Description**
Continuous experimentation (CE)	A software development approach that aims at supporting R&D decisions through iteratively conducting systematic experiments. The term *continuous* represents the iterative nature and sustainability of the approach and *systematic* indicates that business assumptions and hypotheses are methodically tested.
Experiment	Adapted from Munezero et al. (2017), experiment refers to the actual process of testing assumptions and uncertainties in the product or service ideas based on a hypothesis.
Target of experiment	Refers to what drives the experimentation. For instance, this can be new ideas, problems or assumptions with respect to a business strategy that need addressing.
Experiment object	An MVF, MVP or other piece of material that represents critical aspects of the product, the feature or parts of software that will be experimented on. Hypotheses are tested with the experiment objects.
Assumption	A thing, aspect of an idea that is accepted as true or as certain to happen, without proof.
Hypothesis	A proposed, testable explanation for a phenomenon.
Metrics	Metrics or success criteria are defined to capture the values pertaining to the product or feature at a specific time during experiment data collection.
MVF	MVF stands for minimum viable feature and refers to the smallest implementation of a feature that provides essential functionality to the user. This artefact can be used as an experiment object.
MVP	MVP stands for minimum viable product and refers to the minimal set of features that provides the essential value for both the users and owners of the product. This artefact can be used as an experiment object.
User	Adapted from ISO (2017) definition of users, we reserve the term user(s) as a more general term to refer to individual(s) interacting with a software product or service.
Customer	Adapted from ISO (2017), we reserve the term customer for specific kind of user(s) or group(s) who buys the software artefact, obtains a license to use or to rent to other users.
User involvement	Adapted from Yaman et al. (2016), the process by which users consciously or unintentionally provide input into software development.

Chapter 3

Research Design

This research was designed to gather empirical knowledge through studies conducted with industry partners. The empirical studies comprised a multiple-case study, a survey and a single-case study. We aimed at collecting both qualitative and quantitative data and analysed them using multiple techniques, including data triangulation and member checking. This chapter describes the research methodology, including the research phases and research design details within each phase.

This research utilised two main research methods: case studies and survey methodology. The methods were chosen according to the research questions that were sought to be answered as well as the industry partners' interests and availability. The research design included three main phases as depicted in Figure 3.1: In Phase 1, we ran two case studies simultaneously in two case companies to study how CE can be introduced to their development teams. In Phase 1, researchers were on-site and provided feedback into the experimentation process. Following that, Phase 2 included a survey study with four companies to obtain more data towards understanding CE from a practitioner point of view. The survey was designed to inquire into practitioners' experience and attitudes towards experimentation and user involvement, and it was tailored to fit each company's context and structure. The last phase, Phase 3, included a single-case study about how CE can be integrated into a company's goals and context. With this study, we were able to investigate CE as an organisational mechanism.

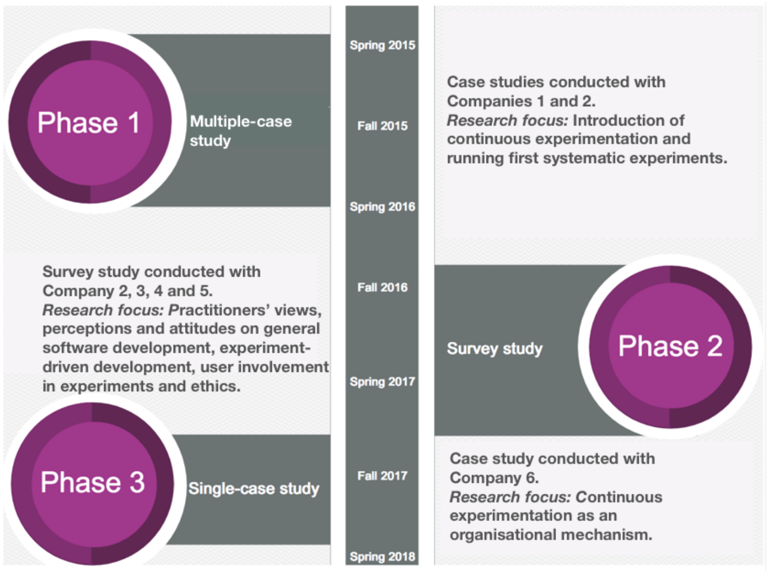

Phase 1
Multiple-case study
Spring 2015
Fall 2015
Spring 2016
Fall 2016
Spring 2017
Fall 2017
Spring 2018
Case studies conducted with Companies 1 and 2.
Research focus: Introduction of continuous experimentation and running first systematic experiments.
Survey study conducted with Company 2, 3, 4 and 5.
Research focus: Practitioners' views, perceptions and attitudes on general software development, experiment-driven development, user involvement in experiments and ethics.
Survey study
Phase 2
Phase 3
Single-case study
Case study conducted with Company 6.
Research focus: Continuous experimentation as an organisational mechanism.

Figure 3.1: Phases of the research together with the timeline.

3.1 Phase 1: Multiple-Case Study

Research for this dissertation began during the time of a research programme called Need For Speed (N4S)[1] in Finland, which involved partnerships between software-intensive companies and research institutions. Several companies involved in the programme were interested in adopting the CE approach. Hence, we planned to collaborate on the introduction of CE, running the first systematic experiments and understanding experimentation involving users within these organisations. Phase 1 (Section 3.1) of this research relies on the industry partnerships with two companies (Company 1 and 2 in Table 3.1) facilitated by the N4S research programme.

This first phase of the dissertation followed a multiple-case study approach, with the process of the introduction of CE as the unit of analysis. Company 1 and 2 were studied as cases at this phase and were both software product and service companies. Both Company 1 and 2 were interested in adopting the approach and were motivated to take part in this study. Companies had prior familiarity with the concept of experiment-driven development; however, they

[1]http://www.n4s.fi/en/

did not conduct experimentation in a systematic way. In particular, the study in this phase focused on specific development teams within each company that wanted to conduct software experiments in a systematic way for the first time.

Using multiple-case studies at this phase allowed us to study the process of the introduction of CE in real business contexts (Yin, 2009) and helped us gain a deeper understanding of how different contextual and human factors and perspectives influence the adoption of the approach. In this phase, researchers participated in the experimentation planning process, providing insights into better experiment design as well as feedback when needed.

Table 3.1: The companies involved in Phase 1.

Company	**Company size and domain**	**Participants**	**Study focus**
1	A large digital services and new business company.	- Development team of 4. - 5 external researchers.	The company's need to improve an existing B2B service led us to build an MVF as an experiment object and test it with beta users.
2	Division of a very large telecommunications network company.	- Development and UX team of 8. - 5 external researchers.	The teams' urgent need to resolve the uncertainties about an ongoing development task led us to design and run series of UI experiments with proxy users.

Table 3.1 demonstrates the profiles of the companies involved in Phase 1. Company 1 is a digital business consulting and services company that specializes in developing new business and digital services. Its business offering includes software development, consulting and service design among others. The focus of the case study with Company 1 was on a service that the company developed for its client. The client further provides this service to its own customers, in a multi-layered B2B customer structure. The demand from the client to improve the service within a tight schedule and budget constraints posed some challenge to Company 1. Fortunately, there was willingness from Company 1 to understand the client and its customers better and to prioritise the development decisions, which facilitated the collaboration on adopting CE.

Company 2 is an international corporation, specializing in providing communication technology and services. One of the products the company was developing was a cloud service platform that was being re-implemented. In particular, there was one feature that was the focus of our case study, as several development decisions had to be made within a schedule, and there were some uncertainties about these decisions. Therefore, adopting CE to support development decisions was of the company's interest.

In terms of data collection, the primary data sources for the multiple-case study included transcripts of audio recordings, minutes and notes of workshops (both on-site and remote), email communication, open-ended, semi-structured and unstructured interviews. In addition, researchers participated in experiment designs that were subject to this research in this phase by giving feedback on experimentation materials such as design and analysis documents prepared by the practitioners. Altogether, four workshops were held with Company 1, and eleven on-site and remote workshops with Company 2 to exchange information about the companies, their products and services and the introduction processes. The content of the workshops are detailed in Article I and II. In these articles, the workshops were mainly referred as meetings, however, due to the interactive and hands-on nature of these events, we refer them as workshops here.

All case data, including the recordings, transcriptions, notes of the workshops, background material, interview design, responses and data analysis were stored in a database. We followed an iterative thematic analysis approach (Braun and Clarke, 2006; Robson, 2011). The method allowed us to identify, analyse and encapsulate themes within the data, as the aim was to comprehend occurring themes when introducing CE to the development teams. As the multiple-case study included two companies in this phase, we also made a comparative study to further understand the similarities and differences in the introduction processes that have emerged from company contexts. In addition, we incorporated *participant observation* and *member checking* techniques to enhance the validity and reliability of the results (Creswell, 2014).

The results from the case studies were disseminated to the company representatives over multiple sessions. In these sessions, how the results guide the companies' journey towards adopting CE was discussed. Phase 1 was reported in Articles I and II of this dissertation. Company representatives also participated in these publications, ensuring the introduction processes were well-described. The summary of the Phase 1, including the data collection and data analysis methods used, as well as the main outcomes, can be seen in Table 3.2.

3.2 Phase 2: Survey Study

In order to gain a deeper understanding of ongoing development activities and to identify the perspectives of software practitioners towards involving users in the development of products and services, as well as software experiments, we designed and conducted a survey as the second phase of the research.

The survey was iteratively designed by three researchers, taking existing survey and questionnaire research, such as Bradburn et al. (2004); Yu and Cooper (1983) into consideration. A conceptual framework based on prior research and the observations from prior studies with companies was developed. The framework consisted of three main areas: *a)* the current state of involving users in software development, *b)* views on experimentation with users and *c)* views of notifying and involving users in experiments. These areas are reflected in the survey structure, with three sections following a background section. The background section was designed to collect demographic data of the participants, ranging from their job functions to years of experience and age-groups. To accommodate for differences in the companies, the survey was constructed as a template based on the conceptual framework, so that certain items were tailored to each company (e.g., using company-specific job position titles).

In particular, before disseminating the survey, practitioner roles were tailored to match the actual titles or job definitions in each company. Other sections of the survey were customised as well when needed. We selected the applicable term of *customer* and *user* involvement in order to refer to the primary user – someone who uses the relevant software in each context/company. For instance, for the employees of Company 4, (see Table 3.3), *user* is the relevant term, as the company has direct access to users, whereas for Company 5, which is a consultancy company, *customer* is the party for whom the products are developed. Two meetings were held with the representatives from each company in order to ensure accuracy of the mappings between company terminology and the concepts in our framework.

The survey was administered to four different software companies operating in Nordic countries, though respondents were distributed over the companies' offices in Europe and the United States. A total of 130 people from different roles, such as developers, managers and UX designers, participated in the survey. Table 3.3 gives a brief description of the companies, size of the target group and the number of responses from each company. The survey included questions formed with various design techniques and elements, such as Likert-type scales (Gliem and Gliem, 2003), and also open text field questions, therefore, collected data

Table 3.2: Summary of the main data collection and analysis methods used at each phase of the research, as well as the main outcomes.

Phase	Research method	Data collection	Data analysis	Outcome
1	Case study	On-site and remote workshops (4 workshops with Company 1, 11 workshops with Company 2), transcripts of audio recordings, notes of the workshops, transcripts of the series of interviews, other materials provided by the companies.	Iterative thematic analysis, participant observation, member checking	Descriptive knowledge gained from introducing CE, including *a)* common and distinct activities followed by the case companies, *b)* decision points together with timelines, *c)* benefits gained during the introduction process and *d)* guidelines based on the learning and challenges faced, together with mitigation strategies. Outcomes reported in Articles I and II.
2	Survey methodology	2 up-front meetings for the survey design and 2 meetings for the dissemination of the results and discussion for Companies 1, 3 and 4; 18 online status meetings and 4 on-site workshops with Company 2. The minutes and notes of the workshops and online status meetings.	Descriptive statistics, inferential statistics, association rule learning, thematic analysis, member checking	Findings from *a)* the current state of involving users in software development, *b)* views on experimentation with users, *c)* views on notifying and involving users in experiments, and *d)* underlying patterns with respect to user involvement in experimentation from the practitioner point of view. Outcomes reported in Articles III and IV.
3	Case study	The minutes and notes interviews with company representatives, notes of phone and email communication during the 6-month period of the study, materials provided by the company executives, notes of 1 workshop	Iterative thematic analysis, member checking	Lessons learned from a company where CE is an integral part of software development, investigating contextual and organisational characteristics associated with CE. Outcomes reported in Article V.

Table 3.3: Companies participated in Phase 2.

Company	**Company size and domain**	**Target group**[2]	**# Responses**
2[1]	A division of a very large telecommunications network company.	231	35
3	A large information security company.	25	8
4	A medium-sized company providing a user interface development toolkit.	135	21
5	A large digital consultancy providing software development services.	397	66
Total		788	**130**

Note 1: Company 2 also participated in Phase 1.
Note 2: Target group represents the population size the survey was sent to in each company.

was in both quantitative and qualitative forms. Data was collected for two weeks in each company during the period from November 2016 to April 2017.

In Phase 2, we followed both qualitative and quantitative data analysis methods. First, we pre-processed the collected data from four companies, so that it could be merged into one whole dataset. As the survey was tailored for each of the four companies, the collected data had to be transformed in a consistent form before merging them together. The transformation was done in accordance with the mapping we had created in collaboration with each company representative. For the data analysis, descriptive statistics, association rule learning (ARL) (Hastie et al., 2009) and thematic analysis were employed depending on the need and purpose of the analysis. Descriptive statistics was used to understand the demographics of the respondents, and to summarise the responses to each survey question that was in quantitative form. In addition, we applied ARL on the full dataset to identify underlying patterns. Furthermore, iterative thematic analysis was employed on the qualitative data collected from the open text field questions, for instance, where we asked participants to describe what a typical software experiment was.

Finally, in order to confirm the findings and to allow our participants to comment on the results and conclusions and possibly use them in their own context, we employed member checking in company-specific feedback sessions with the company representatives and selected participants. In addition, with Company 2, we held online status meetings and a number of workshops where we interactively discussed the findings. In these company-specific sessions, action-able insights were also discussed with company representatives. This phase was reported in Articles III and IV, as also summarised in Table 3.2.

3.3 Phase 3: Single-Case Study

In order to seek answers for our last research question on understanding CE as an organisational mechanism, we designed and conducted a case study with Company 6. The company develops mobile games for their users and has a business-to-consumer (B2C) structure. Table 3.4 demonstrates the profile of the company and the study focus, where the unit of analysis was the experimentation process.

Table 3.4: Company involved in Phase 3.

Company	**Company size and domain**	**Participants**	**Study focus**
6	A medium-sized company developing mobile games.	- 2 company representatives. - 2 external researchers.	Company's established practices on CE and user involvement.

Following the statement by Yin (2009) that "[...] you should choose the case(s) that will most likely illuminate your research questions.", we partnered with Company 6 that is already practising CE as an integral part of their development activities. The case study with Company 6 resembles a revelatory case, which occurs when researchers have the opportunity to empirically observe and analyse a case that was previously not accessible (Yin, 2009). Our aim was to study the company's experiment-driven development and user involvement practices without researcher intervention in order to understand how experiments are conducted continuously, what forms they take and what makes CE company culture. Furthermore, as the context of mobile games is a very competitive domain, we also looked into contextual drivers that could influence experiment-driven development. For this purpose, we performed a series of interviews with company representatives and inquired into their experience with CE and user involvement.

Interviews were used as the basis to understand the experimentation process and as means to understand the retrospective aspects of the game development. In this phase, we did not interfere with the company's experimentation process; we studied the material provided by the company as well as observing their development activities. The dataset consisted of the transcripts of series of interviews with company representatives, notes of phone and email communication during the time of study, material provided by the company executives and notes of one workshop. The interviews aimed to capture the company's development and experimentation processes, as well as the dynamics of mobile game development. During the data analysis stage, additional data was collected through phone and email communication when needed. At the end, a workshop was conducted with the company representatives to capture final reflections and confirm the study results.

The data collection phase of the study took place from November 2017 to March 2018. During the data analysis, we performed iterative thematic analyses on the collected data and member checking, which enabled checking by company representatives that the researchers had understood their operations correctly.

In the end, the findings formed a summary of operations from a company, where CE is an integral part of software development, investigating the contextual and organisational characteristics associated with CE. One company representative actively participated in the reporting stage, therefore, the accuracy of the findings and validity were ensured. The findings of this phase were as reported in Article V.

Table 3.2 summarised the data collection and data analysis methods used in each phase of the research as well as listing the main outcomes from each phase. In summary, Phase 1 included a multiple-case study, Phase 2 was designed and conducted to employ the survey methodology, and Phase 3 included a single-case study. The research outcomes are summarised based on the main findings of each phase, and they are presented in detail in the following chapter.

Chapter 4

Transitioning towards Continuous Experimentation

Continuous experimentation (CE) is getting increasing attention both from the practitioners and researchers as an approach to support development decisions with the empirical results obtained through experiments. Existing research mostly focuses on a specific type and aspects of experimentation, such as online controlled experiments and business aspects; however, how to begin running systematic software experiments is not well-investigated. In this research, we focus on how organisations can initiate the transition towards CE. In light of the research questions and empirical studies designed accordingly, the results of this research were reported in Articles I–V. In this chapter, we overview the findings from the articles involved in this dissertation, following all three phases of the research. First, we look at the findings upon introduction of CE to the development teams at our research partners, which constituted the first phase of the research. Second, we outline the findings from Phase 2 of the study, in which we gathered more data from practitioners at software companies through the survey study. Third, we summarise the findings and learning from Phase 3, in which we investigated how CE can be integrated into an organisation's development activities and work as an organisational mechanism. Lastly, we look at the summary of the findings and investigate how the findings from the each phase of the study relate to each other.

4.1 Introducing Continuous Experimentation

Despite the increasing demand of the CE approach, the lack of guidance on how to systematically and effectively introduce it into an organisation has been the primary motivation of the study reported in Articles I and II. Article I presented the multiple-case study described in Chapter 3. The study investigated how the CE approach can be introduced to existing software-intensive organisations that are willing to adopt the approach. During the case studies, several development teams at the organisations pioneered their first systematic experiments with the guidance of researchers. The introduction process was planned to capture the companies' journeys during the study, including activities involved, decision points during the process, the benefits gained and challenges faced. Comparisons of the companies' introduction processes revealed both common and company-specific results.

Common decisions made during the introduction process by the case companies varied from selecting an experiment target as a UI element to iteratively updating the experiment design multiple times. Despite the common decisions, there were differences emerged from the teams' ongoing development commitments and other contextual factors. For instance, companies chose their experimentation target based on their ongoing prioritization of R&D workloads. In particular, they chose an urgent development question to be answered by the experiments. Moreover, it was evident that starting adopting the approach at the team level, designing the first experiments to be small-scale and having practitioners who can influence development decisions involved in the introduction process were important. Ongoing development activities, strict schedules and budget restrictions also influenced the decision on conducting UI experiments as a starting point – as it was less risky and did not require big changes in technical infrastructure. Through the UI experiments, it was possible to plan and run the first systematic experiments at a low cost.

During the introduction process, development teams were already able to make development decisions based on the empirical data gathered by the first experiments. In fact, one development team realized that they were going to make a completely different decision on deploying a UI element, which was proven by the subjects of the experiment to be the least user-friendly. Disproving a product assumption that was otherwise going to be implemented to the end users was a remarkable lesson for the development teams: product ideas should be tested in a systematic way instead of relying on guess work that would bring little or no value to the users. In this experimentation study, the development teams were able to experience how a small-scope experiment at a low cost can be impactful in eliminating a faulty product idea.

Furthermore, as the outcome of the experiments showed direction for the development, the teams were able to back up the development decisions with experimentation. In other words, they did not need to debate on the course of action taken after the experiments, as the experimentation offered empirical results. At the same time, experiment design was discussed and improved over several sessions together with the researchers to ensure the highest possible validity. For instance, in one of the case companies, the experiment was rerun after an alteration in the design. The need for alteration emerged during the piloting of the experiment, which offered a concrete basis for the discussion sessions to assess the experiment and its design.

Besides the benefits, challenges faced during the introduction process included difficulties in accessing the end users of the software products to involve them in the experiments. This difficulty was primarily caused by the companies' multi-layered B2B stakeholder structure. Company representatives were also hesitant about contacting real users for various reasons, including the bureaucracy involved in reaching out to the end users and uncertainty of the ethical and privacy issues involved in user data collection. However, alternative methods were used to mitigate these challenges in the first experiments. For instance, proxy users were used to overcome the challenge of inaccessible end users. Using proxy users in the first systematic experimentation process helped development teams gain experience, while the learning was used to plan future experiments with real users. We also proved that experiments with proxy users can resolve development uncertainties in a systematic approach, even though they might not offer statistically significant experiment results due to limited population size. Over the workshops and discussion sessions with the practitioners and researchers, it was collectively decided to inquire further into data collection and ethics with respect to software experiments as future work.

Article II elaborated on one specific case company involved in the multiple-case study, in order to better capture the experimentation details and learnings. Article II had a two-fold research focus: *1)* details of the design and execution of the experiment conducted at the case company and *2)* identification of significant themes emerged from the introduction process together with the challenges and potential mitigation strategies to act as guidelines.

Two development teams at the case company and five researchers collaborated on the planning, designing, execution and analysis of the UI experiment that was run twice. Based on the learning from these experiments, the initial steps to transition towards CE were reduced into the following themes:

- *Initial circumstances pose challenges* for the development teams at existing organisations, such as ongoing development commitments and limited resources. It is important to look at the ongoing development from a broad perspective and prioritise the need for experimentation inline with the organisational goals. Essentially, while trying to answer an important development question with the help of experiments, choosing a small scope for the first experiments was found to be useful.

- *Starting with small teams and small-scale experiments are observed to be beneficial*, however, they should be planned attentively. For instance, higher level business goals might not be visible at a team level when planning the first experiments. Involving several teams or individuals in planning the experiment and having champions in the team who are able to influence decision-making and can pioneer the transition were found to be useful strategies. It is also important to understand that starting small does not indicate that first experiments should be small in impact.

- *Identifying an experiment target that will be a good fit for the first experiments can be difficult.* Utilising existing product materials and prioritizing the user needs to be addressed by the experiment are necessary. Having face-to-face workshops together with the development teams and decision-makers (also with researchers in our studies) aid evaluating the user and business needs, ongoing commitments and choosing the right experimentation target collectively.

- *Designing and executing the experiment can be challenging*, considering the inexperience in experimentation and lack of skills required, for instance, to avoid experiment biases and to define experiment metrics and success criteria. This can lead to a significant amount of time to learn running valid experiments, that development teams might be hesitant to allocate. As a mitigation strategy, piloting and rerunning the experiments are advisable. In addition, as there is no comparison point for the first experiments, being flexible with metrics is advised.

- *Collaborating with experts requires time and effort* for the development teams to introduce their product and context. On the other hand, experts help avoid big and risky mistakes and can facilitate an effective transition and learning process.

- *Persistence is required*, as the first experiments are often arduous and might be inefficient or unsuccessful due to inexperience. Experience and learning from each trial, as well as building tools, methods and infrastructure for reusability can increase efficiency. Furthermore, champions in the organisations who promote the transition can help disseminate learning outcomes and keep the motivation for the transition up.

In summary, Articles I and II offered in-depth analysis of two case companies' journeys towards introducing CE. The results revealed that companies were willing to adopt the approach and were open to learning from the introduction process, however, planning their first experiments given the ongoing development activities with short deadlines and lack of experimentation skills impede the process and demotivate development teams. A systematic introduction process with guidance helped companies experience an efficient start. Overall, starting at a small-scale and at a low cost and running the experiment with existing resources created a good response at the organisations, also with the help of champions involved in the process. Choosing an experimentation target so that the experiment can address an important aspect of the ongoing development was effective for the development teams in observing that product assumptions can lead to bad product features, if not tested.

4.2 Practitioners' Perspectives and Attitudes

In order to facilitate an efficient transition to CE at an organisation, it is important to understand the perspectives and attitudes held by practitioners in different roles. Hence, Articles III and IV reported on the user involvement survey, which aimed to *1)* explore how software companies involve users in software development and experimentation, *2)* understand how developer, manager and UX designer roles perceive and involve users in experimentation, *3)* uncover systematic patterns in practitioners' views on user involvement in experimentation. We wanted to put special emphasis on the ethics involved in software experimentation, as little has been published on ethics in software experiments. Therefore, we reported the ethics part of the study separately in Article III, while reporting the full study findings in Article IV.

In particular, Article III described one section of the user involvement survey that was aimed at addressing ethical issues organisations must consider when planning and conducting experiments with users (see Section 3.2 for survey design). The ethics section of the survey asked 130 company employees in different roles, such as developers, managers and UX designers, to reflect on their perceptions and preferences towards user notification and user involvement in software experiments. Out of 130 responses, 71 were developers, 23 were managers and 22 were UX designers. The findings showed that the practitioner role had a big influence on how individuals perceived the notification of users and their involvement in software experiments. For instance, managers were observed to be more cautious with user notification; they strongly indicated that users should always be aware of software experiments they are subject to. On the other hand, UX designers expressed on average that users do not always need to be notified of an experiment they are subject in. Furthermore, UX designers tended to allow exceptions in user notification and involvement; for example, it might be acceptable to inform the users after the experimentation was carried out and not to disclose some of the details of the experiment to the users. Developers also considered practical exceptions in notifying users after the experiments and they believed that users would be willing to take part in experiments. The results also revealed that some ethical aspects are shared and agreed by all the roles, such as "if personal information is collected, users should always be notified". In order to better understand these findings, we performed a full analysis of the survey, where we inquired into practitioners' existing ways of working extensively.

Complementarily, Article IV reported on the full analysis of the survey with 130 practitioner responses of all survey sections. Before being asked about the experimentation practices in detail, practitioners were asked to describe a situation, in which involving users in the development would be useful, but was not possible to do so. Table 4.1 shows the major reasons for users being inaccessible for involvement in software development and experimentation. The reason mentioned most often was the challenge of accessing the end users of the software due to a multi-layered structure of users and customers. Practitioners often work for their companies who deliver the software solution to their customer, who might sell or deliver the product to their own users. Therefore, practitioners might not be allowed to contact the end users and must rely on pre-determined user requirements delivered to them. Other common reasons why users could not be involved in the development included time and budget constraints and lack of process. Especially when there is no clearly defined process to involve users, practitioners might omit or give up on the task of user involvement, given

Table 4.1: Major reasons why users could not be involved in the development activities, according to practitioners.

Theme	**Description**
Multi-layered user/customer structure	Companies might often have *customers* who sell or deliver the software product to their own users; therefore, it might not be possible to access their users. There might also be financial conflicts between different layers. In addition, the customers might think that they are already knowledgeable about what the user wants.
Time and budget constraints	Even if users might be accessible, due to ongoing commitments and tight deadlines, it might be difficult to reach them. The customers might find it costly to allow practitioners to involve users in the process.
Lack of process	There might be no clear process of when and where to involve the users. Heavy bureaucracy, such as getting the right permits to contact users, might also slow down the development.
Consent and privacy	It might be difficult or impossible to get users' consent due to privacy reasons. Alternative solutions, such as test labs, might not be the same as monitoring users on-board an actual flight.
Pre-determined requirements	The user requirements might already be determined in advance, and practitioners are told to follow them.

other pressures such as time. In addition, a group of practitioners also expressed that vague or non-existing rules and procedures with respect to user privacy and consent issues impede the user involvement process.

Furthermore, the results showed that practitioners had different views on what experiments are to begin with. Table 4.2 summarises the two common descriptions given by practitioners when requested to describe what a typical experiment is according to their experience and beliefs. The majority described experiments as UX/UI activities and user studies organised by practitioners from UX/UI teams. The commonly used terms to describe these experiments included: *user studies, scenarios, surveys, interviews and walkthroughs.* On the other hand, 22 respondents described the experiments using the following terms: *hypothesis-based, A/B tests, user analytics, building MVPs and releasing part of a feature or software to (a subset of) users to collect data.* This finding showed us that practitioners understand experimentation differently to begin with.

Moreover, the role analysis revealed different attitudes towards collecting data from users. For example, a significant group of developers opted for wide user data collection, indicating that data should be collected from users, so that it could be explored later when needed. In other words, user data collection does

Table 4.2: Type of experiments, as described by respondents of the user involvement survey.

Type	**Description**	**# people**
UX/UI activities and user studies	Practitioners referred to activities and user studies organised and conducted by job functions, such as UX/UI designers, to describe experiments. The referred activities are: BDD stories, user stories, scenarios, usability tests, surveys, interviews, shadowing sessions, workshops, walkthroughs, talkalouds, mockups	28
Hypothesis-based experiments and analytics	Experiments that are driven by pre-defined hypotheses, measuring user behaviour, collecting and using user data and analytics for experiments. Practitioners referred to the following terms when describing these experiments: A/B tests, prototypes, MVPs or MVFs, partial or limited release, piloting with proxy users	22

not necessarily need to be guided by an up-front plan or strategy but can be exploratory. At the same time, developers reported the least frequent contact with users and ranked the lowest on having sufficient and up-to-date user information. On the other hand, managers and UX designers favoured focused data collection, that is collecting user data when there is a specific question that needs testing. Specifically, UX designers have the most frequent and direct contact with users and are confident about the quality of the information they have. They use various tools and methods to involve users in their job, more often than other roles. The results showed that by proportion, UX designers reported conducting active experimentation the most. In fact, none of the UX designers reported not conducting experiments at all.

The full analysis of the survey also supported our initial analysis that the practitioner role has a direct influence on the understanding of ethics. In particular, we observed that practitioners tend to interpret and rationalise the notification of users in experiments based on their job functions. In other words, understanding ethics in experimentation and user involvement are constructed by job functions and personal experience. For instance, UX designers are the group that allows for exceptions in user notification the most. To them, users do not always need to be notified of experiments, and it might be okay not to disclose all the experiment details. Moreover, they did not think that users have to be convinced of the benefits in advance, nor that experiments would reveal product secrets.

Table 4.3: Existing patterns identified from the user involvement survey and their description.

Pattern	**Description**
Focused data collection	A pattern indicating that data does not always need be collected in case it might be needed later. User behaviour should be measured to know what the software should be like. People who follow this pattern report that they conduct experiments actively, often use log data, have relevant user information and opt for focused data collection.
Wide data collection	A group of respondents are associated with the pattern that data should not only be collected when there is a known need or assumption, instead, rich and detailed data about what users do is always useful. A large group of developers is associated with this pattern, which also includes respondents, who agree that data should always be collected because it might be needed later and who use log data often as a data collection method.
Strict ethical attitude	Regardless of any exception, users should always be notified of an experiment. This pattern also includes respondents, who are likely to think that users have to be convinced of the benefits before taking part in an experiment, experiment results might not be trustworthy, and involving users in experimentation is time-consuming.
Practical ethical consideration	A group of respondents disagree that users should always be notified or need to know that they are involved in an experiment. It is also acceptable to not disclose some experiment details to users. Respondents in this pattern are likely to think that users do not need to be convinced to take part in experiments.
Unrestrained experimentation	A pattern including a group of respondents who opted for wide data collection, and who are associated with practical ethical considerations such as not allowing the disclosure of all the experiment details to the users and disagreeing that users have to be convinced to take part. A subset of these respondents also report their active experimentation practices.
Easy user access	Easy access to user information is associated with not needing permission to contact users, direct user access, having relevant and sufficient user information. People who opt for these statements are also likely to think that users would want to be part of experiments.

Based on our analysis, six patterns emerged from the full dataset of responses on the survey, as summarised in Table 4.3. These patterns were formed to describe the trends across the whole set of responses to the survey. Patterns involved aspects concerning data collection methods and strategies, ethics of experimentation and access to users. These patterns are not mutually exclusive in relation to each other, meaning they are constructed to explain existing trends emerged from the dataset. Findings showed that practitioners can exhibit patterns at two polar opposites – wide data collection vs. focused data collection, and strict ethical attitude vs. practical ethical consideration. The unrestrained experimentation pattern brings wide data collection and practical ethical consideration together and describes the group of practitioners who favour wide data collection yet also allow exceptions in user notification; one of the companies that participated in the survey (Company 5 in Table 3.3) fits the unrestrained experimentation pattern well. This also means that companies might practise and exhibit their experiment-driven development differently, combining different data collection methods and having different attitudes towards software experimentation. Lastly, we identified the easy user access pattern that described the group of respondents who do not need permission for user contact, who have direct access to the users, and who are confident about the quality of the user information they have. These people are also likely to believe that users would want to take part in experiments.

Patterns can be used to detect existing trends and to describe and understand software organisation stances on experiment-driven development. Therefore, the existing ways of working or processes that are undermining experiment-driven development could be determined as well as finding out what skills and tools could enhance experiment-driven development. Such examination can aid a better evaluation of organisational needs and goals for adopting CE.

4.3 Continuous Experimentation as an Organisational Mechanism

Phase 3 of the study analysed and reported on a case study conducted with a mobile game development company, in which experimentation works as an organisational mechanism. The findings were reported in Article V. The article describes the company's software development and experimentation practices and investigates the factors that lead to an effective CE mechanism at the company. In this study, the characteristics of the mobile game development were also investigated with respect to experimentation. The mobile gaming market is so competitive that it is almost impossible to create new, truly innovative games via mechanisms that would only rely on planning. This calls for an approach, where the focus is on changing directions toward success, which can be realized by making experimentation involving users have a central role in the development process.

One of the main findings from the study indicated that experiments are important and valuable at all development stages in the case company, however, they take different forms at different stages. For instance, in the early stages of the development, experiments are designed to collect more qualitative data from the subjects, whereas once the product is in the market, the amount of quantitative data is vast. Balancing between different experimentation methods, such as complementing quantitative and qualitative data collection based on the stage of development, appears to be the key to success. In addition, the results showed that regardless of the stage at which experiments take place, they need a concrete goal, i.e., up-front hypothesis and metrics, in order to be successful. Yet, most of the experiments fail to prove the assumptions; in other words, the hypotheses are rejected, and this is considered to be useful. In fact, by rejecting hypotheses through experiments as often as possible, bad product ideas are eliminated at a rapid pace. This is evidenced by the company representatives who report that the most valuable learning come from failed experiments, as it is very critical to eliminate wrong product ideas and assumptions as early as possible. The company has a learning mechanism to capture the findings from the experiments and design new ones accordingly.

Furthermore, the interviews with company representatives revealed that in the context of game development, conducting experiments was seen as *a must* to support development decisions during the evolution of the product, and the development teams are always motivated to design and run experiments. The competitive market steers the direction of development decisions regarding new game features and themes, and CE has been a highly prevalent approach in this field. However, experimentation practices require strong leadership and expertise to schedule the steps at a rapid pace and design systematic experiments in such a competitive environment.

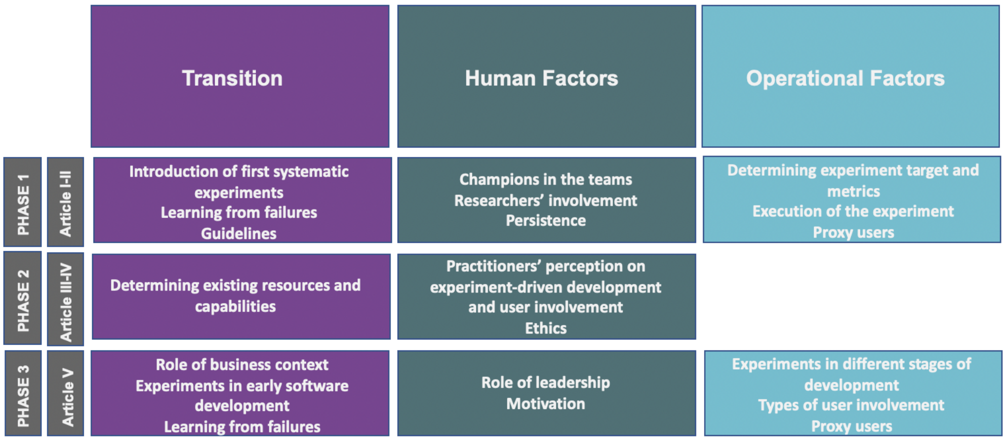
One of the main challenges was inaccessible *real users* to experiment with, especially early in the development. As a coping strategy, the company reported the use of alternative methods such as testing with proxy users. Internal users, such as company employees, can be involved in experiments in the early stages and as the product matures, external users, such as recruited users under non-disclosure agreement or a subset of beta users, can be involved to test the product. Once the game is fully on the market, quantitative data collected through experiments becomes available en masse. At that point, focused experiments with pre-defined metrics, e.g., key performance indicators (KPIs), become very important in evaluating the success of the product, which leads to statistically significant experiment results. Experiments designed to collect qualitative data are required and useful, especially in the early stages of the development. Richness of the qualitative data is important to evaluate, for instance, the desirability and aesthetics of a product feature early on.

Lastly, we discovered that experiments take place throughout the development life-cycle of a mobile game, yet the number of experiments are highest during the early development life-cycle, e.g., during prototyping and preproduction. When the product gets more mature with time, experiments become more specific. However, the experiments that take place early in the development can be more influential in the overall development, as they are more conclusive on how the product is going to be shaped. For instance, in the early stages, main theme of a mobile game can be evaluated and decided via experiments. In addition, the cost of change in later stages is seen as high; therefore, it is very important to effectively design and run experiments early on to eliminate such big changes. On the other hand, experimentation in the earlier stages might offer lower confidence, as they are not performed with real users. That is why speed is very important in order to move to stages where experiments can be more specific and they can be performed with real users. For example, a new game feature can be tested via experiments and development decisions can be made. Then, experiment results will offer more confident results, as they reflect what real users want.

4.4 Summary of the Findings across Study Phases

Each of the three phases of the research were designed to pursue answers to the research questions presented previously. In this chapter, we present the findings from each phase of the research, reported in Articles I–V. The findings from each phase revealed recurring themes in relation to each other. Figure 4.1 compiles the main research findings from the phases and categorises them into three major themes. Even though this dissertation was designed and reported in separate phases (depicted horizontally), these recurring themes offer supportive insights and answers to our research questions (depicted vertically).

The process of *transition* to CE addresses the research findings, including the activities conducted when introducing CE to the development teams, the lessons learned (Articles I and II), determining companies' existing resources and capabilities (Articles III and IV) and the investigation of how CE functions as an integral part of an organisation's development activities (Article V). Furthermore, we identified the research findings from each phase of the research that are related to *human factors*, including having champions in the transition process and practitioners' perspectives of experiment-driven development and ethics. All the articles included in this dissertation have contributed to this theme. Lastly,



		Transition	Human Factors	Operational Factors
PHASE 1	Article I-II	Introduction of first systematic experiments Learning from failures Guidelines	Champions in the teams Researchers' involvement Persistence	Determining experiment target and metrics Execution of the experiment Proxy users
PHASE 2	Article III-IV	Determining existing resources and capabilities	Practitioners' perception on experiment-driven development and user involvement Ethics	
PHASE 3	Article V	Role of business context Experiments in early software development Learning from failures	Role of leadership Motivation	Experiments in different stages of development Types of user involvement Proxy users

Figure 4.1: The phases and the articles of the dissertation in three themes across the categorisation of the important findings.

the final theme, *operational factors*, refers to the research findings related to the elements of experimentation, data collection and user involvement in terms of actions to be taken. These actions included planning and designing the experiment target and determining the metrics for the experiment (Articles I and II) and different types of data collection and user involvement activities (Article V). In Discussion (Chapter 5), while answering our research questions, we will also discuss these findings across all study phases.

Chapter 5

Discussion

The CE approach is foreseen as a novel extension of continuous practices, such as continuous integration and delivery (Olsson et al., 2012; Ros and Runeson, 2018). The approach can be used to aid the R&D decisions, while involving users in the process through iterative and sustainable software experiments. In this three-phase study, we learned that the CE approach can be introduced to the development teams in an economical and efficient way, and human factors, such as practitioner roles and motivation, play an important role in the transition towards CE. In this chapter, we discuss the study findings in depth. First, we summarise the contributions to our research questions and scientific implications. Next, we indicate the practical implications that emerged from the findings. Lastly, we examine the limitations of this research and consider and discuss potential future work.

5.1 Scientific Implications

In the first phase of the study, we investigated how to introduce CE to the development teams at existing software organisations. We described the introduction process at two organisations and identified the operational actions to be taken to run the first systematic experiments. From the findings, several human factors also emerged, such as the role of champions, who drive the introduction process in the development teams. The research aspects associated with human factors were further investigated through the survey study such as how practitioners define software experiments. In the last phase, while investigating how the CE approach is effectively practised in a mobile game company, we obtained knowledge that contributes to the transition process to CE such as the importance of experiments in the early stages of software development and learning from failures. Furthermore, we learned about the operational factors involved in the practice of CE, including how different types of experiments can be planned, and how different types of user involvement can be facilitated. Eventually, we obtained supporting findings to our research questions from all three phases of the study and the findings formed a whole. In this section, we will summarise the answers to our research questions, while also discussing the supporting findings from all phases of the study, as well as the existing research (see Figure 4.1 for the summary of all findings across study phases).

5.1.1 Initiating the Transition Systematically

Existing research to a large extent reports on (online) controlled experiments (Lindgren and Münch, 2015), and further, the research field is mainly dominated by big technology companies (Ros and Runeson, 2018). Not only they do not publicly share their datasets, which makes it difficult to fully understand the details of experiments (Ros and Runeson, 2018), but they also do not explicitly disclose how they transitioned towards the CE approach. This is understandable in terms of confidentiality; in fact, in our study, we were also careful with only sharing the transition process and snippets of experiments, not disclosing any company-sensitive data, since the companies have to protect their business secrets. However, since there is a lack of guidance on how to initiate such a transition, our first research question addressed this research gap. Based on the research findings, we summarise the contributions to the RQ1 as follows, followed by a discussion on it:

RQ1: How can continuous experimentation be introduced in software development teams?

CE can be systematically introduced to software development teams by utilising existing resources, limiting the complexity and starting with small teams and scope, however, not small in impact for the first experiments. The introduction process can be facilitated with guidance and by systematic experimentation, the validity and sustainability of approach can be improved each time an experiment is conducted while capturing learning. Experiments can be conducted at various stages in the development life-cycle, such as even before the product is in the market, and in various forms, such as using proxy users, when the real users do not exist yet. With a systematic introduction process, the benefits of the approach can be observed early on, which is very crucial for the software industry, where budget and time to adopt a new approach are limited.

Case study findings from Phase 1 offered us useful insights on introducing CE to development teams, systematically (see Section 4.1). Starting with a small team size and a small scope, as well as choosing the experiment target, so that the development team can experience the benefits on the spot, were found to be useful. Furthermore, having a champion in the development team, who can carry out the experiment design and execution processes and especially the dissemination of the results into development decision-making, was found to particularly valuable.

Understanding the organisation context, existing resources and capabilities available to experiment with users are important in designing the first experiments as well as to determining the overall direction of the transition process to CE. In our case studies in Phase 1, this required time and effort from company representatives when introducing the company to the researchers and for researchers when introducing systematic CE. Having multiple workshops and discussion sessions were very crucial to evaluate the overall development and ongoing commitments and to determine the experiment target, where the first systematic experiment would be the most beneficial to conduct. Choosing the right target for the first experiment is also important in terms of impact of experiment results. We consider the introduction of CE as a learning process, yet we also learned from our studies that witnessing the immediate benefits of the first experiments highly motivated the practitioners to carry on with the transition. Furthermore, in Phase 2 of the study, we intensively investigated existing organisational resources and capabilities and identified several patterns on experiment-driven development and user involvement in the development in general. These patterns can be utilised by the developments teams to identify their current position with respect to experiment-driven development and how to govern the transition toward CE.

The influence of organisational context on the practice of CE was also evident in Phase 3, where the domain was mobile game development. The dynamics of the game development market, such as high-competitiveness and rapid changes in the demand of game features, steer the direction of the development for game companies. We observed that as the market has a high utilisation of the CE approach, it was *a must* for the case company to adopt the CE approach, as well. Company representatives from this study emphasized that having CE well-established in their development culture is motivated by the needs of the market.

We emphasize that initiating the transition towards CE is a learning process, in which the first experimentation attempts might fail. We have learned from the company representatives in Company 6 that they learn the most from failed experiments and utilise the learning to establish their own guidelines for systematic experimentation. On the other hand, existing research also indicated that one of the reasons why software practitioners cannot transition easily is the fact that the first experiments are likely to fail and this demotivates the teams to make the investment of adopting the approach (Madeyski and Kawalerowicz, 2017). However, we have showed by our research findings that this can be overcome with several strategies, including using proxy users, piloting the experiments to gain experience and being flexible with metrics for the first experiments, as there are no comparison points. Consequently, the ambivalence towards inexperience in experimentation holding development teams back from practising the approach can be dealt with a systematic introduction to the approach. With systematic introduction, existing resources can be utilised, and the process can be economical while being impactful.

Another important finding was made during Phase 3 – experiments that take place early in the development are as valuable as post-deployment experiments, and in fact, they are more conclusive in shaping the software. In the domain of mobile game development in particular, existing research indicates that once a game is fully implemented, it is very expensive to make a change, such as fixing a problem, and this will effect whole project schedule (Aleem et al., 2016). We also know from traditional software engineering research that the cost of a change or a fix significantly increases as the software product matures during the development (Boehm et al., 1981). Therefore, it is important to create the capability of CE in the earlier stages of software development. Even though there is existing research emphasizing the importance of experiments early in the development (e.g., Thomke (2001); Bosch-Sijtsema and Bosch (2015) and Lindgren and Münch (2015)), they did not offer explicit guidelines on how to conduct such experiments. In our introduction processes, we ran the first experiments at early development stages with proxy users and observed the benefits. While in Phase 1, we conducted the first experiments on mature products, in Phase 3, we also studied CE beginning from the product ideation stage.

5.1.2 Human Factors

Examining human factors was relevant to our research from multiple angles. At first, an organisational transition requires a change in practitioner work behaviour, therefore, it is important to study the individuals' perspectives during the change. In Phase 2 of the research, the primary goal was to collect rich data from the practitioners through a survey to better comprehend their attitudes and perceptions. In addition, a number of other human factors also emerged such as having champions in the introduction process and the motivation to keep experiment-driven development up and running during Phases 1 and 3, as can be seen from Figure 4.1. The RQ2 and a summarised answer to the question, followed by an extensive discussion to it are as follows:

RQ2: What perspectives and attitudes do practitioners at organisations exhibit with respect to experiment-driven development?

Practitioners at software organisations tend to describe software experiments differently, and the practitioner role is influential in perceiving the ethics of experimentation. For instance, UX designers tend to see experiments as user studies and might allow exceptions in user notification such as letting the users know of the experiments only afterwards. Software organisations need to evaluate such differences and set their goals and guidelines for adopting experiment-driven development. Having motivated people and decision-makers in the transition process promotes the benefits of experimentation early on and helps plan and manage the process.

We found from the multiple-case study that having champions in the development teams, who are enthusiastic about furthering the approach, was crucial. These champions did not only encourage the rest of the team, but also communicated the study and its results to the rest of the organisation. We saw that having the champions affected the decision-making process in the organisations and enabled a more efficient introduction process, as the experiment results were disseminated and implemented quickly. In addition to this, researchers participated in the introduction process as experts providing guidance when needed, which was also found to be useful and helped eliminate big mistakes. However, we believe that after gaining enough experience by learning from experiment trials and even failures, development teams are expected to be more experienced on their own.

The last phase of our research on CE as an organisational mechanism revealed that the role of leadership is also essential in planning, designing and running the experiments continuously. With the guidance of leaders, teams eventually turn into self-organising teams. We also acknowledged that different organisational contexts, such as mobile game development, have different market dynamics and, therefore, affect how practitioners in the organisations are motivated to adopt the approach. Consequently, we can argue that motivation for transitioning to CE might differ between organisations and development teams, however in any case, persistence is required to make the approach become company culture.

In the second phase of the research, we asked practitioners to describe a typical experiment they have seen or have been involved in their company. The results showed that the term *experimentation* was used inconsistently by practitioners, a finding that was also pointed out by existing research such as Sjøberg et al. (2005) and Ros and Runeson (2018). We found that there are two major perspectives of what experiments are. Majority of the practitioners described typical experiments as UX/UI designers' activities conducted with users. On the other hand, the descriptions provided by the rest of the practitioners showed more awareness of hypothesis-based experimentation and other related methods and techniques such as A/B tests, user analytics and limited time release. The latter description is more inline with the core elements of experiment-driven development, as we reviewed in the previous work. However, we also acknowledged that experiment-driven development can use instantiations or parts of other practices such as usability engineering and user-centric design, and it can cover a broader scope. For instance, a broader description of experimentation was offered by Gutbrod et al.Gutbrod et al. (2017), where they report on their multi-case study with several startup companies, that were run in various forms, including interviews, trade show testing, landing page, A/B testing and MVP testing depending on the specific need for the experiment. When there are differences in perceiving experiments in an company, there might be risks stemming from mismatched understandings. For instance, some practitioners might believe that experiments are/should be conducted only by the UX designers. When they do not consider themselves designing or running experiments, they might be resistant to adopt CE. Therefore, it is advisable for the companies to address such differences and fix the objectives for adopting the approach accordingly.

Furthermore, we observed that the role of practitioner makes a big difference in having an attitude towards experiment-driven development and ethics. UX designers were found to use several tools and methods to interact with the customers and users the most, and they reported knowing their users well compared to other roles. A prominent finding was the significant variation between practitioner roles with respect to ethics. For example, while managers tended to think that users have to be convinced of the benefits before taking part in experiments, UX designers allowed exceptions in user notification more easily – users would not always need to be notified of experiments they take part in, and it may be acceptable to not disclose some experiment details to them.

In terms of participant awareness and notification, it is known from the experiments of other disciplines, such as psychological experiments, that participants' awareness of being experimented on can bias results. It may be defensible to carry out such experiments, provided that the harm done is negligible. In scientific experimentation in general, it is considered obligatory to disclose this to participants afterwards and to allow them to withhold consent to use the data. For this reason, the UX designers' attitude toward user notification and their welcoming attitude towards experimentation can be due to the nature of their job function. They are generally familiar with user studies and different methods, such as prototyping, user surveys and interviews, and how biases are involved in the design of these studies. Organisations need to consider such differences in understanding software experiments and ethics, when initiating the transition to CE. Existing research does not offer extensive findings in terms of practitioner role in experimentation, with the exception of a few studies such as Fagerholm et al. (2017) and Fabijan et al. (2017b). These studies address the need for different roles in experiments.

The ethical line of inquiry has also been important to study, as only little has been published on software experiments and ethics (Ros and Runeson, 2018). The findings indicate that practitioners see the ethical issues from their own perspective, influenced by their job functions. In fact, we claim that practitioners tend to rationalise what is acceptable in accordance with their job function. Due to unclear rules and policies about user consent, privacy and data collection, practitioners tend to perceive software experiments according to their own experiences and beliefs. We also learned from the case studies that uncertainty about ethical issues slowed down the introduction process. Consequently, we argue that due to lack of clear process of involving users in experimentation in organisations, as well as lack of research and regulations about ethics of experimentation in general, practitioner understanding of the ethics of experimentation might not establish a common ground. Once the future research and the regulations, such as European GDPR, offer more insights, the ethical issues concerning CE can be better assessed.

5.1.3 Operating Experiments and User Involvement

Experimentation can take place in different forms and at different stages of software development. Besides it can be perceived differently by different practitioner roles, it can be practised differently depending on the business domain and organisational needs. Based on our research findings, the RQ3 and summarised answer to it is as follows:

RQ3: How can a continuous experimentation mechanism be established in an organisation, adapting to the business domain and organisational goals?

When transitioning towards experiment-driven development, challenges of inaccessible real users can be overcome by alternative methods such as proxy users. While the experimentation validity will improve with practice and iteratively conducting experiments, establishing an organisational mechanism that will capture learning from the experiments and adapt to both organisational and contextual needs is crucial. Balancing between different experimentation methods and user involvement in accordance with business context and organisational goals is the key to success.

In this research, we observed that company representatives were hesitant about contacting real users for the first experimentation for various reasons – the multi-layered customer structure creates accessibility problems, it might require formal work to contact customers due to permissions, and inexperience with experimentation can damage real users' trust in case of failure. Besides, ethical issues involved in data collection can impede the process. Table 4.1 summarised the main reasons why practitioners could not gain access to users. As a solution, using proxy users in the first systematic experimentation process helped development teams gain experience at a rapid pace, since accessing real users, especially in a multi-layered B2B structure, would have taken longer.

Figure 5.1 represents the different types of user involvement in experiments that were studied in this dissertation. We designed and ran the first systematic experiments, which were conducted with proxy users in our case studies in Phase 1. These experiments were run in the early stages of software development with internal company practitioners in the companies. In general, using proxy users might be especially convenient for experiments, when the product or feature is not in the market and no real users exist yet. We anticipate that while maturing in experimentation, case companies that participated in Phase 1 could soon begin experimenting with a subset of real or recruited users. Furthermore, these

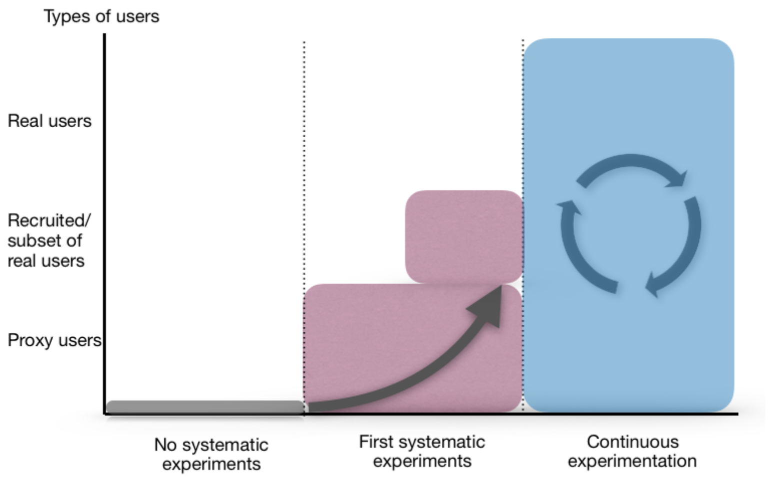

Types of users
Real users
Recruited/ subset of real users
Proxy users
No systematic experiments
First systematic experiments
Continuous experimentation

Figure 5.1: Experimentation with respect to the types of users involved, as studied in this research. The curved arrow represents how the transition to CE was initiated with the proxy users in Phase 1. On the other hand, in Phase 3, an established CE mechanism was studied, where all types of user involvement can occur, depending on the design.

companies can eventually reach the capability of CE, while also building the infrastructure for it, in which they can conduct experiments iteratively. As we learned from the case study with the mobile game development company, continuous experiments can be run with different types of users, depending on the need for the experimentation and the stage of the development. In Company 6, early stage experiments in particular are conducted with proxy users, and while the product matures in production, experiments are started with a bigger scope of users from outside the development teams and the company such as recruited users. However, at the same time, experiments with a subset of users or proxy users are run anytime when needed, for instance, when a new feature has to be tested with a smaller set of users first. As for organisations that just started to adopt experiment-driven development, it might also be possible to start by experimenting with real users. Once the companies have the capability of CE, they can experiment with different degrees of user involvement, depending on the necessity.

In addition to the user involvement in experiments, in Phase 3, we found that in the earlier stage experiments, the data collected through experiments are often qualitative. For instance, qualitative data on user preferences of UI aesthetics can be collected through experiments. Such experiments that are designed to collect qualitative data are important in order to understand the desirability of the product, mobile games in this context, where the real users do not exist yet. We learned from the case study that the case company conducts experiments to collect qualitative data throughout the development life-cycle in order to complement the experiments that collect quantitative data.

Furthermore, one of the challenges impeding the transition to CE was already observed in Phase 1. It was difficult to ensure experiment results with a high degree of validity, as the sample size was not big enough to offer statistically significant results. This is natural when a large number of real users may not be available. Especially when the sample number is limited, random assignment to control and treatment groups might not be possible. Instead, as we discussed earlier, the first experiments can be small in size and scope. Moreover, success criteria and metrics for the first experiments might not be realistic, as there is no comparison point. However, we observed that piloting the experiments and being flexible with the metrics in the beginning helped address this challenge. In general, as the CE mechanism becomes more established in an organisation through capturing learning and adapting to the organisational needs, the validity of experiments is expected to increase.

5.2 Practical Implications

Software practitioners are often concerned by limitations, such as budget and time, in competitive markets. It is especially important to consider the benefits of a new approach at a low cost when initiating a transition. This transition, therefore, requires a careful introduction process. In this dissertation, we showed that experiments can be introduced to software development teams at a low cost, by planning and running small-scale experiment with development teams, yet knowing that the results can be impactful. For instance, in the case study conducted in Phase 1, an experiment that was run using PowerPoint slides with the proxy users revealed that an assumption on a development feature, that was otherwise going to be implemented, was wrong. In other words, thanks to the experiment, a bad product idea was successfully eliminated before it was implemented and deployed to the end users.

In order to initiate the transition towards CE, companies may either seek guidance from experts, if they are not familiar with experiment-driven development, or initiate the process themselves. Our research findings can be applicable to either case. For instance, internal training sessions on systematic experimentation can be arranged, where our guidelines are followed. Furthermore, as discussed previously, it is important to set organisational goals for adopting CE. For instance, business domains, such as game development, require a very rapid pace and experimentation in different forms throughout the development-life cycle. Business structures, such as B2B, require a careful assessment of customer structure and their needs to aim the experiments at the right subject groups. Furthermore, such a structure might require companies to go through bureaucracy to ensure the accessibility of the users and data collection.

Especially keeping in mind that due to the novelty of the research field of CE, there has not been a common understanding on where the practitioners' views reside. The patterns identified from practitioner responses to the survey study can be used by software practitioners to examine their position with respect to experimentation and user involvement. We do not strictly claim which set of patterns constitutes the right mindset for the experiment-driven development approach; on the contrary, we found and discussed that companies can adopt and practise the approach so that it fits their way of working as well as organisational goals.

One of the key practical consideration for practitioners who want to adopt CE in their organisations is to choose the right experiment target for the first systematic experiments. The right experiment can address an important development question at a low cost, and the learning can be scaled up to the organisation. Likewise, an organisational mechanism that will gather the learning from the first experiments and disseminate the results to the relevant parties for development decision-making is needed. In order to do this, new roles in the companies might be necessary such as experimentation scientists.

5.3 Threats to Validity and Limitations

This research did not directly aim at investigating causality and we did not set strong hypotheses when designing the studies. In addition, we did not directly aim at offering normative solutions, but rather extracting and composing descriptive and observational knowledge and lessons learned about adopting CE in companies. Our study design and research findings were exposed to several threats to validity and limitations. We took several precautions during the research design process to mitigate them. Here, we discuss threats to validity, as well as the limitations of the research findings.

Construct validity for this research was taken into consideration from multiple angles. Firstly, we studied the practice of software experiments and its evolution (as reviewed in Section 2.2). Empirical software engineering has adopted experimentation in the field of software development, with the purpose of being able to produce better software products fostered by evidence-based decision-making. Furthermore, experiments in CE practice have significant differences to a natural science experiment. One of the primary differences is that software experiments have pragmatic utility and are driven by business needs. Determining the research terminology (see Section 2.5) and scoping the research accordingly helped overcome such risks that would have emerged from inconsistent use of terms.

Secondly, in the phases of research where we conducted case studies, we were careful to collect and extract data based only on the study design, and we involved company representatives in the analysis stage to ensure that our results and interpretations did not conflict with their understanding. In general, in all of the three phases of research, we tried to ensure *data triangulation* and *member checking* and considered *researcher bias* following the guidelines and recommendations of Yin (2009) and Creswell (2014). We used multiple data resources to determine the consistency of the findings and all researchers systematically participated in the study design, the data analysis and the review of the findings to confirm accuracy. Researchers who were not involved in the initial data analysis of the different phases of research reviewed the results in an effort to eliminate bias.

In particular, concerning Phase 2 of this research, the survey study, we worked closely with company representatives to tailor the survey draft, so that it would fit their company domains and contexts. However, we were careful with maintaining the same goal and structure of the survey, so that the datasets from each company could later be merged. For instance, options for the practitioner roles were added, removed or modified, depending on the actual role descriptions at each company. We believe that discussing the job functions with company representatives helped us better understand descriptions of the roles in different companies. Additionally, in the survey design in Phase 2, we purposefully avoided including an explanation or example of what an experiment, the elements of experiments or implications on ethical issues were. We wanted to observe what the respondents consider to be an experiment, and how they perceive ethical issues by themselves. For this purpose, we had multiple types of questions, for instance, Likert-type scales (Gliem and Gliem, 2003) and open questions, and we also asked respondents to describe what an experiment is in their opinion as well as any challenges they had faced in user involvement. Different forms of collected data helped us cross-validate the findings and capture the similarities and differences in the perceptions held about experiments and users.

In terms of external validity, the focus of this dissertation was the systematic initiation of the transition towards CE. This research ultimately aims at a full transition to CE as an organisational mechanism, in which development teams have the skill-set and infrastructure to run experiments iteratively and continuously, when needed. In order to research the transition to the CE process, we studied the introduction of the approach first. In Phase 1 of this research, the introduction of CE was guided by researchers, who had expertise on the subject. Furthermore, in Phase 3, we also studied a CE mechanism without expert guidance and gained further insight into how CE can be adapted within an organisation. We hope that our research findings are helpful and transferable for development teams and organisations who want to initiate their transition without expert involvement, as well.

In addition, in this dissertation, we started with the introduction process of CE with development teams and disseminated the benefits and lessons learned to the other teams. Our aim was to scale up the approach to the rest of the organisation, while practising and gaining experience. We already observed the benefits of starting with small teams and small scale-experiments in our studies, however, in other contexts or organisations, other introduction approaches might also be possible.

In general, the company domains, structures and cultures certainly influence transitioning towards CE. In our case studies, we worked with the development teams of large organisations in domains such as telecommunications and digital consultancy during the introduction of the CE phase. Afterwards, we investigated CE as an established company culture in the domain of mobile game development. However, dynamics might be different in other domains or contexts. For instance, for start-ups, budget and time constraints might be more restrictive to adopt a new approach. Furthermore, we observed in our case studies that user/customer structure influences the design and execution of the first systematic experiments. In particular, a multi-layered user/customer structure in a B2B domain requires an understanding of the needs of each user level as well as operational issues such as accessibility to the end users. In terms of generalisability, we cannot make strong claims based on three case studies and a survey study, however, the confirmability of the findings between different phases of the study indicates that it would not be surprising if the findings of this dissertation were valid for other companies, as well. We are very much interested in seeing how our findings apply in other contexts.

5.4 Future Work

The research in this dissertation is a step forward towards providing empirical guidance on how to transition to CE from the ground up. In this section, we discuss potential future work complementary to our research findings.

In this research, we showed that the first experiments can take different forms and take place at different stages of the development life-cycle, and any development team might be able to do it. We also addressed the differences in the type of data primarily collected in different experiments and the validity of experiment results. In particular, we noted that experiments that are designed to collect qualitative data are as valuable as those that collect quantitative data from users. One particular question that future work could address is: *"How to combine different types of experiments, that are designed to collect qualitative and quantitative data?"*. More case studies are needed to study the different types and forms of experiments at different stages of software development.

Furthermore, in this research, we investigated how CE can be introduced to the development teams. Planning the first systematic experiments with the teams various aspects to consider, such as the cost of the experimentation, even though it was not an initial objective of the research. In this research, we found out that starting the first experiments economically is important, yet future research should evaluate our findings. In addition, future research could address different transition approaches. For instance, in new settings, the decision for transitioning towards CE might come from the management level, directly driven by business considerations, and might propagate towards development teams.

One of the key findings showed how CE can function as an organisational mechanism, and we found that CE has to adapt to the business domain and organisational goals. Two of our case companies shared a B2B structure, while one company had a B2C structure. However, each case company had different business domains and needs. We emphasized that user needs are different in such layered structures, and correspondingly, the need for experimentation differs. More case studies are needed to better understand the effect of user structure combined with the needs emerging from the domain in experiment-driven development.

Likewise, we emphasized in the research scope of this dissertation that we study the initiation of the transition to CE, yet we do not directly aim at continuity with the first experiments. However, one question is: *"How to start building the infrastructure for CE, while initiating the approach?"*. In the first systematic experiments, development teams do not have to build big technical infrastructure, but as the transition progresses, the number of experiments in parallel, as well experiment data collection, are expected to increase. In order to deal with such complexity, several technical changes might be needed to trace the experiments and user data. In addition, as we already addressed in Chapter 2, in software engineering there is an emergent phenomena emphasising the necessity of a focus on continuous activities, including continuous delivery and DevOps (Fitzgerald and Stol, 2017). How these related activities can work in synchronisation with CE, both from a methodical and technical point of views are yet to be investigated. Future research can study the creation of such continuous systems to involve and support CE.

Finally, we believe that the development and practice of new regulations on user data collection will affect the research on ethical issues involved in software experiments in the future. Furthermore, future research on understanding the end users' point of view on their involvement in experiments would also be valuable. This can be done through studies such as surveys or polls. Thus, the grounds for user data collection and privacy would be better understood by involving them in the studies more directly.

Chapter 6

Conclusions

Software companies have always tried to find more efficient ways to deliver value to their users. Learning from and about the users, such as the way they use software, have become a must, and therefore, an integral part of the whole development process. Agile methodologies and Lean Startup share similar principles of involving users in the development, gathering justifiable data about users in an iterative fashion in order to produce better software. Software experiments are commonly used in software development, especially in the last decade, to collect evidence on the value of the software product and services. The CE approach aims to utilise the strengths of experimentation in software development in order to support design decisions continuously. As the approach is novel and of increasing interest, one of the research areas is how to adopt the approach in organisations. In this dissertation, we examined how this transition to CE can be initiated in software organisations. We conducted empirical studies with development teams and practitioners to obtain descriptive and observational knowledge.

We found that the transition towards CE is a learning process, in which the benefits of the approach with a systematic introduction can be observed early on. We learned from our studies with development teams, that the first experiments can be designed in a small scope yet can be impactful for the organisation. Existing resources can be utilised, and as well as being economical, the first experiments can be efficient. We showed that alternative methods can be used to overcome the challenge of a lack of real users by using proxy users and proved that experiment results can change the direction of development decision-making. In this dissertation, researchers helped introduce CE in the case companies. However, development teams and practitioners in other organisations can utilise the findings to start their own transition process by following our guidelines.

Furthermore, we studied the influence of human factors on experiment-driven development and the transition to it and found that personal beliefs and experience have a strong influence on how software experiments are perceived. For instance, UX designers tend to recognise experiments as UX/UI-related studies and prefer methods for user data collection that are in line with their job function. Such differences in understanding and conducting experiments might need to be addressed in order to set organisational goals for adopting the approach. In addition, we observed that having a champion in the development teams expedited the introduction process and the dissemination of the results as well as feeding the results into decision-making. Witnessing the benefits of a new approach early on also motivates practitioners to embrace the change, and motivation is an important factor when establishing CE as an organisational culture.

Software experiments often collect data from human subjects, therefore, the ethical considerations should be taken seriously into account. We inquired into what practitioners think about user involvement and notification of software experiments. We found that the practitioner role does not only affect how individuals perceive software experiments, but also the ethical issues. Practitioners tend to rationalise what can be acceptable in ethical issues in accordance with their job function. For instance, managers are cautious about the company-customer relationship, and they think that users should always be notified of experiments in advance, whereas UX designers allow for exceptions such as letting them know afterwards. Such differences in ethics indicate that organisations have to work on their regulations and policies concerning user data collection, especially in accordance with data protection regulations their organisations are subject to.

Studying an organisation's established CE practices showed that the CE functions as an organisational mechanism that adapts to the business domain and organisational goals. Business domain and the goals for adapting CE affect both the technicality and the cost of first experiments. In the field of mobile game development, for instance, where the competition is very high, experiments are vital in order to move quickly, while constantly testing product assumptions. Experiments can take many different forms, ranging from qualitative user studies to A/B tests, depending on the development stage, yet experiments take place throughout the development. Organisations can develop such a mechanism gradually, assessing their business structure and using alternative methods for user involvement, while improving the validity of the experiments each time. Furthermore, we learned that having a shared organisational goal for experimentation reflects on the practitioners, and motivation becomes inherited.

This dissertation provides several novel contributions to software engineering research and practice. We found that despite the existing research on a specific type of software experiment and dominance of big companies, it is possible to run different forms of experiments depending on the stage of software development and the need for experimentation. Furthermore, we discovered that even when real users are not available, software experiments involving users can be designed and ran. We also investigated the ethics involved in software experiments and found that when there are no established regulations, ethical issues are left to practitioners' own interpretation. Above all, we observed that CE should be introduced systematically, and it should be treated as a learning process, in which development teams can observe how empirical evidence might change the direction of development decisions. An organisational mechanism should be built to support such a transition.

References

Abelein, U. and Paech, B. (2015). Understanding the influence of user participation and involvement on system success–a systematic mapping study. *Empirical Software Engineering*, 20(1):28–81.

Agarwal, R. and Dhar, V. (2014). Big data, data science, and analytics: The opportunity and challenge for is research.

Aleem, S., Capretz, L. F., and Ahmed, F. (2016). Game development software engineering process life cycle: a systematic review. *Journal of Software Engineering Research and Development*, 4(1):6.

Bakshy, E., Eckles, D., and Bernstein, M. S. (2014). Designing and deploying online field experiments. In *Proceedings of the 23rd international conference on World wide web*, pages 283–292. ACM.

Barney, S., Aurum, A., and Wohlin, C. (2008). A product management challenge: Creating software product value through requirements selection. *Journal of Systems Architecture*, 54(6):576–593.

Basili, V. R. (1996). The role of experimentation in software engineering: past, current, and future. In *Software Engineering, 1996., Proceedings of the 18th International Conference on*, pages 442–449. IEEE.

Basili, V. R., McGarry, F. E., Pajerski, R., and Zelkowitz, M. V. (2002). Lessons learned from 25 years of process improvement: The rise and fall of the nasa software engineering laboratory. In *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, pages 69–79, New York, NY, USA. ACM.

Bass, L., Weber, I., and Zhu, L. (2015). *DevOps: A software architect's perspective.* Addison-Wesley Professional.

Blank, S. (2013). *The four steps to the epiphany: successful strategies for products that win.* BookBaby.

Boehm, B. and Turner, R. (2003). *Balancing agility and discipline: A guide for the perplexed.* Addison-Wesley Professional.

Boehm, B. W. et al. (1981). *Software engineering economics*, volume 197. Prentice-hall Englewood Cliffs (NJ).

Bosch, J. (2012). Building products as innovation experiment systems. In Cusumano, M. A., Iyer, B., and Venkatraman, N., editors, *Software Business*, pages 27–39, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bosch, J., Olsson, H. H., Björk, J., and Ljungblad, J. (2013). The early stage software startup development model: a framework for operationalizing lean principles in software startups. In *Lean Enterprise Software and Systems*, pages 1–15. Springer.

Bosch-Sijtsema, P. and Bosch, J. (2015). User involvement throughout the innovation process in high-tech industries. *Journal of Product Innovation Management*, 32(5):793–807.

Bradburn, N. M., Sudman, S., and Wansink, B. (2004). *Asking questions: the definitive guide to questionnaire design–for market research, political polls, and social and health questionnaires.* John Wiley & Sons.

Braun, V. and Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101.

Callan, V. J. (1993). Individual and organizational strategies for coping with organizational change. *Work & Stress*, 7(1):63–75.

Creswell, J. W. (2014). *Research design: Qualitative, quantitative , & mixed methods approaches.* SAGE Publications, Inc, 4th edition.

Croll, A. and Yoskovitz, B. (2013). *Lean analytics: Use data to build a better startup faster.* " O'Reilly Media, Inc.".

Davenport, T. H. (2009). How to design smart business experiments. *Strategic Direction*, 25(8).

Diestel, R. (2005). *Graph Theory.* Springer, Berlin, 3rd edition.

Dingsøyr, T. and Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*, 77:56–60.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development.

Ebert, C., Gallardo, G., Hernantes, J., and Serrano, N. (2016). Devops. *Ieee Software*, 33(3):94–100.

Eisenberg, B. and Quarto-vonTivadar, J. (2009). *Always be testing: the complete guide to Google website optimizer*. John Wiley & Sons.

Fabijan, A. (2016). *Developing the right features: the role and impact of customer and product data in software product development*. Malmö university, Faculty of Technology and Society.

Fabijan, A., Dmitriev, P., Olsson, H. H., and Bosch, J. (2017a). The benefits of controlled experimentation at scale. In *Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on*, pages 18–26. IEEE.

Fabijan, A., Dmitriev, P., Olsson, H. H., and Bosch, J. (2017b). The evolution of continuous experimentation in software product development: from data to a data-driven organization at scale. In *Proceedings of the 39th International Conference on Software Engineering*, pages 770–780. IEEE Press.

Fabijan, A., Olsson, H. H., and Bosch, J. (2015). Customer feedback and data collection techniques in software r&d: a literature review. In *International Conference of Software Business*, pages 139–153. Springer.

Fagerholm, F., Guinea, A. S., Mäenpää, H., and Münch, J. (2014). Building blocks for continuous experimentation. In *Proceedings of the 1st international workshop on rapid continuous software engineering*, pages 26–35. ACM.

Fagerholm, F., Guinea, A. S., Mäenpää, H., and Münch, J. (2017). The right model for continuous experimentation. *Journal of Systems and Software*, 123:292–305.

Fitzgerald, B. and Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123:176–189.

Franz, C. R. and Robey, D. (1986). Organizational context, user involvement, and the usefulness of information systems. *Decision sciences*, 17(3):329–356.

Gliem, J. A. and Gliem, R. R. (2003). Calculating, interpreting, and reporting cronbach's alpha reliability coefficient for likert-type scales. In *Proceedings of Midwest Research to Practice Conference in Adult, Continuing, and Community Education*.

Gould, J. D. and Lewis, C. (1985). Designing for usability: key principles and what designers think. *Communications of the ACM*, 28(3):300–311.

Grudin, J. (1992). Utility and usability: research issues and development contexts. *Interacting with computers*, 4(2):209–217.

Gutbrod, M., Münch, J., and Tichy, M. (2017). How do software startups approach experimentation? empirical results from a qualitative interview study. In *International Conference on Product-Focused Software Process Improvement*, pages 297–304. Springer.

Hannay, J. E., Sjoberg, D. I., and Dyba, T. (2007). A systematic review of theory use in software engineering experiments. *IEEE transactions on Software Engineering*, 33(2):87–107.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

Henfridsson, O. and Lindgren, R. (2010). User involvement in developing mobile and temporarily interconnected systems. *Information Systems Journal*, 20(2):119–135.

Hess, J., Offenberg, S., and Pipek, V. (2008). Community driven development as participation?: involving user communities in a software design process. In *Proceedings of the Tenth Anniversary Conference on Participatory Design 2008*, pages 31–40. Indiana University.

Highsmith, J. and Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, 34(9):120–127.

Iivari, N. (2004). Exploring the rhetoric on representing the user: discourses on user involvement in software development. *ICIS 2004 Proceedings*, page 52.

ISO (2017). Iso/iec 25010:2011, systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models.

Järvinen, J. and Mikkonen, T. (2017). Need for speed–towards real-time business. In *International Conference on Product-Focused Software Process Improvement*, pages 621–624. Springer.

Kautz, K. (2010). Participatory design activities and agile software development. In *IFIP Working Conference on Human Benefit through the Diffusion of Information Systems Design Science Research*, pages 303–316. Springer.

Kitchenham, B. A., Dyba, T., and Jorgensen, M. (2004). Evidence-based software engineering. In *Proceedings of the 26th international conference on software engineering*, pages 273–281. IEEE Computer Society.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K., and Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on software engineering*, 28(8):721–734.

Kohavi, R., Deng, A., Frasca, B., Walker, T., Xu, Y., and Pohlmann, N. (2013). Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1168–1176. ACM.

Kohavi, R. and Longbotham, R. (2017). *Online Controlled Experiments and A/B Testing*, pages 922–929. Springer US, Boston, MA.

Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. M. (2009). Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery*, 18(1):140–181.

Kramer, A. D., Guillory, J. E., and Hancock, J. T. (2014). Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, 111(24):8788–8790.

Kujala, S. (2003). User involvement: a review of the benefits and challenges. *Behaviour & information technology*, 22(1):1–16.

Laage-Hellman, J., Lind, F., and Perna, A. (2014). Customer involvement in product development: An industrial network perspective. *Journal of Business-to-Business Marketing*, 21(4):257–276.

Lagrosen, S. (2001). Strengthening the weakest link of tqm–from customer focus to customer understanding. *The TQM Magazine*, 13(5):348–354.

Lagrosen, S. (2005). Customer involvement in new product development: A relationship marketing perspective. *European Journal of Innovation Management*, 8(4):424–436.

Larman, C. and Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56.

Lindgren, E. and Münch, J. (2015). Software development as an experiment system: a qualitative survey on the state of the practice. In *International Conference on Agile Software Development*, pages 117–128. Springer.

Madeyski, L. and Kawalerowicz, M. (2017). Software engineering needs agile experimentation: A new practice and supporting tool. In Madeyski, L., Śmiałek, M., Hnatkowska, B., and Huzar, Z., editors, *Software Engineering: Challenges and Solutions*, pages 149–162, Cham. Springer International Publishing.

Manzi, J. (2012). *Uncontrolled: The surprising payoff of trial-and-error for business, politics, and society.* Basic Books (AZ).

Mattos, D. I., Bosch, J., and Olsson, H. H. (2018). Challenges and strategies for undertaking continuous experimentation to embedded systems: Industry and research perspectives. In Garbajosa, J., Wang, X., and Aguiar, A., editors, *Agile Processes in Software Engineering and Extreme Programming*, pages 277–292, Cham. Springer International Publishing.

Mikkonen, T., Lassenius, C., Männistö, T., Oivo, M., and Järvinen, J. (2017). Continuous and collaborative technology transfer: Software engineering research with real-time industry impact. *Information and Software Technology.*

Muller, M. J., Haslwanter, J. H., and Dayton, T. (1997). Participatory practices in the software lifecycle. In *Handbook of Human-Computer Interaction (Second Edition)*, pages 255–297. Elsevier.

Munezero, M., Yaman, S. G., Fagerholm, F., Kettunen, P., Mäenpää, H., Mäkinen, S., Tiihonen, J., Riungu-Kalliosaari, L., Tuovinen, A.-P., Oivo, M., et al. (2017). *Continuous experimentation cookbook: an introduction to systematic experimentation for software-intensive businesses.* DIMECC result publications; 3-(DIMECC publications series no. 15).

Olsson, H. H., Alahyari, H., and Bosch, J. (2012). Climbing the “stairway to heaven” – a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, pages 392–399.

Olsson, H. H. and Bosch, J. (2014). The hypex model: from opinions to data-driven software development. In *Continuous software engineering*, pages 155–164. Springer.

Poppendieck, M. and Poppendieck, T. (2003). *Lean software development: an agile toolkit.* Addison-Wesley.

Ries, E. (2011). *The lean startup: How today’s entrepreneurs use continuous innovation to create radically successful businesses.* Crown Books.

Rissanen, O. and Münch, J. (2015). Continuous experimentation in the b2b domain: a case study. In *Proceedings of the Second International Workshop on Rapid Continuous Software Engineering*, pages 12–18. IEEE Press.

Robson, C. (2011). Real world research: A resource for users of social research methods in applied settings 3rd edition.

Ros, R. and Runeson, P. (2018). Continuous experimentation and A/B testing: a mapping study. In *Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering, RCoSE@ICSE 2018, Gothenburg, Sweden, May 29, 2018*, pages 35–41.

Shadish, W. R., Cook, T. D., and Campbell, D. (2002). *Experimental and quasi-experimental designs for generalized causal inference.* Houghton Mifflin Boston.

Shahin, M., Babar, M. A., and Zhu, L. (2017). Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5:3909–3943.

Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N.-K., and Rekdal, A. C. (2005). A survey of controlled experiments in software engineering. *IEEE transactions on software engineering*, 31(9):733–753.

Srivastava, J., Cooley, R., Deshpande, M., and Tan, P.-N. (2000). Web usage mining: Discovery and applications of usage patterns from web data. *Acm Sigkdd Explorations Newsletter*, 1(2):12–23.

Tan, P.-N., Steinbach, M., and Kumar, V. (2005). Association analysis: basic concepts and algorithms. *Introduction to Data mining*, pages 327–414.

Tang, D., Agarwal, A., O'Brien, D., and Meyer, M. (2010). Overlapping experiment infrastructure: More, better, faster experimentation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 17–26. ACM.

Thomke, S. (2001). Enlightened experimentation: The new imperative for innovation. *Harvard Business Review*, 79(2):66–75.

Tichy, M., Bosch, J., Goedicke, M., and Fitzgerald, B. (2015). 2nd international workshop on rapid continuous software engineering (rcose 2015). In *Proceedings of the 37th International Conference on Software Engineering-Volume 2*, pages 993–994. IEEE Press.

Tichy, W. F. (1998). Should computer scientists experiment more? *Computer*, 31(5):32–40.

Unni, R. and Harmon, R. (2007). Perceived effectiveness of push vs. pull mobile location based advertising. *Journal of Interactive advertising*, 7(2):28–40.

Vinson, N. G. and Singer, J. (2008). A practical guide to ethical research involving humans. In *Guide to Advanced Empirical Software Engineering*, pages 229–256. Springer.

Williams, L. and Cockburn, A. (2003). Guest editors' introduction: Agile software development: It's about feedback and change. *Computer*, 36(6):39–43.

Yaman, S. G., Sauvola, T., Riungu-Kalliosaari, L., Hokkanen, L., Kuvaja, P., Oivo, M., and Männistö, T. (2016). Customer involvement in continuous deployment: A systematic literature review. In Daneva, M. and Pastor, O., editors, *Requirements Engineering: Foundation for Software Quality*, pages 249–265, Cham. Springer International Publishing.

Yin, R. (2009). *Case study research: design and methods.* SAGE Publications, Inc., 4th edition.

Yu, J. and Cooper, H. (1983). A quantitative review of research design effects on response rates to questionnaires. *Journal of Marketing research*, pages 36–44.

Zhang, B., Wang, N., and Jin, H. (2014). Privacy concerns in online recommender systems: influences of control and user data input. In *Symposium on Usable Privacy and Security (SOUPS)*, pages 159–173.

A-2015-6	Y. Ding: Collaborative Traffic Offloading for Mobile Systems. 223 pp. (Ph.D. Thesis)
A-2015-7	F. Fagerholm: Software Developer Experience: Case Studies in Lean-Agile and Open Source Environments. 118+68 pp. (Ph.D. Thesis)
A-2016-1	T. Ahonen: Cover Song Identification using Compression-based Distance Measures. 122+25 pp. (Ph.D. Thesis)
A-2016-2	O. Gross: World Associations as a Language Model for Generative and Creative Tasks. 60+10+54 pp. (Ph.D. Thesis)
A-2016-3	J. Määttä: Model Selection Methods for Linear Regression and Phylogenetic Reconstruction. 44+73 pp. (Ph.D. Thesis)
A-2016-4	J. Toivanen: Methods and Models in Linguistic and Musical Computational Creativity. 56+8+79 pp. (Ph.D. Thesis)
A-2016-5	K. Athukorala: Information Search as Adaptive Interaction. 122 pp. (Ph.D. Thesis)
A-2016-6	J.-K. Kangas: Combinatorial Algorithms with Applications in Learning Graphical Models. 66+90 pp. (Ph.D. Thesis)
A-2017-1	Y. Zou: On Model Selection for Bayesian Networks and Sparse Logistic Regression. 58+61 pp. (Ph.D. Thesis)
A-2017-2	Y.-T. Hsieh: Exploring Hand-Based Haptic Interfaces for Mobile Interaction Design. 79+120 pp. (Ph.D. Thesis)
A-2017-3	D. Valenzuela: Algorithms and Data Structures for Sequence Analysis in the Pan-Genomic Era. 74+78 pp. (Ph.D. Thesis)
A-2017-4	A. Hellas: Retention in Introductory Programming. 68+88 pp. (Ph.D. Thesis)
A-2017-5	M. Du: Natural Language Processing System for Business Intelligence. 78+72 pp. (Ph.D. Thesis)
A-2017-6	A. Kuosmanen: Third-Generation RNA-Sequencing Analysis: Graph Alignment and Transcript Assembly with Long Reads. 64+69 pp. (Ph.D. Thesis)
A-2018-1	M. Nelimarkka: Performative Hybrid Interaction: Understanding Planned Events across Collocated and Mediated Interaction Spheres. 64+82 pp. (Ph.D. Thesis)
A-2018-2	E. Peltonen: Crowdsensed Mobile Data Analytics. 100+91 pp. (Ph.D. Thesis)
A-2018-3	O. Barral: Implicit Interaction with Textual Information using Physiological Signals. 72+145 pp. (Ph.D. Thesis)
A-2018-4	I. Kosunen: Exploring the Dynamics of the Biocybernetic Loop in Physiological Computing. 91+161 pp. (Ph.D. Thesis)
A-2018-5	J. Berg: Solving Optimization Problems via Maximum Satisfiability: Encodings and Re-Encodings. 86+102 pp. (Ph.D. Thesis)
A-2018-6	J. Pyykkö: Online Personalization in Exploratory Search. 101+63 pp. (Ph.D. Thesis)
A-2018-7	L. Pivovarova: Classification and Clustering in Media Monitoring: from Knowledge Engineering to Deep Learning. 78+56 pp. (Ph.D. Thesis)
A-2019-1	K. Salo: Modular Audio Platform for Youth Engagement in a Museum Context. 97+78 pp. (Ph.D. Thesis)
A-2019-2	A. Koski: On the Provisioning of Mission Critical Information Systems based on Public Tenders. 96+79 pp. (Ph.D. Thesis)
A-2019-3	A. Kantosalo: Human-Computer Co-Creativity - Designing, Evaluating and Modelling Computational Collaborators for Poetry Writing. 74+86 pp. (Ph.D. Thesis)
A-2019-4	O. Karkulahti: Understanding Social Media through Large Volume Measurements. 116 pp. (Ph.D. Thesis)