Empirical Software Engineering (2024) 29:21
https://doi.org/10.1007/s10664-023-10358-z

A theory of factors affecting continuous experimentation (FACE)

Rasmus Ros[1] · **Elizabeth Bjarnason**[1] · **Per Runeson**[1]

Accepted: 5 June 2023 / Published online: 13 December 2023
© The Author(s) 2023

Abstract

Context Continuous experimentation (CE) is used by many companies with internet-facing products to improve their business models and software solutions based on user data. Some companies deliberately adopt a systematic experiment-driven approach to software development while some companies use CE in a more ad-hoc fashion.

Objective The goal of this study is to identify factors for success in CE that explain the variations in the utility and efficacy of CE between different companies.

Method We conducted a multi-case study of 12 companies involved with CE and performed 27 interviews with practitioners at these companies. Based on that empirical data, we then built a theory of factors at play in CE.

Results We introduce a theory of Factors Affecting Continuous Experimentation (FACE). The theory includes three factors, namely 1) processes and infrastructure for CE, 2) the user problem complexity of the product offering, and 3) incentive structures for CE. The theory explains how these factors affect the effectiveness of CE and its ability to achieve problem-solution and product-market fit.

Conclusions Our theory may inspire practitioners to assess an organisation's potential for adopting CE and to identify factors that pose challenges in gaining value from CE practices. Our results also provide a basis for defining practitioner guidelines and a starting point for further research on how contextual factors affect CE and how these may be mitigated.

Keywords Continuous experimentation · Data-driven development · A/B testing · Theory building · Multi-case study · Empirical research

Communicated by: Philipp Leitner

✉ Elizabeth Bjarnason
elizabeth.bjarnason@cs.lth.se

Rasmus Ros
rasmus.ros@cs.lth.se

Per Runeson
per.runeson@cs.lth.se

[1] Lund University, Lund, Sweden

1 Introduction

Continuous experimentation (CE) is an experiment-driven software engineering approach where assumptions about markets, customers, business models, product features and requirements are continuously tested with users. The aim is to reduce the risk of wasting resources on requirements of little or no value to the users. For example, changes to a business model or software solution can be compared to an old version or between new versions in an experiment, e.g. through an A/B test or demonstrations with real users, and only changes with a positive effect on usage are retained. Initially, this approach was primarily applied by large internet-facing software engineering companies such as Microsoft (Kohavi et al. 2009, 2013), Google (Tang et al. 2010), and Facebook (Feitelson et al. 2013) that apply an agile and continuous approach to software engineering. Lately, also more traditional software engineering organizations, such as companies in the business-to-business domain (Rissanen and Münch 2015), are recognising the value of moving towards an experiment-driven development model that provides the ability to evaluate product assumptions through continuous user feedback (Fagerholm et al. 2017).

Performing CE is not an easy task and one that requires companies to adapt their processes, structures, technical infrastructures, and culture (Fagerholm et al. 2014; Yaman et al. 2017). Several researchers report on the benefits of adopting CE and a high-level model on how CE is structured (the RIGHT model) has been proposed by Fagerholm et al. (2017). Despite this, not all companies are able to adopt CE, even though they might have sufficient user data available. We pose that this is due to the complex nature of software engineering and that many different factors and processes are at play. Some of these factors have been identified in previous research, such as having a service-oriented offering (Schermann et al. 2018) or one with cyber-physical systems (Bosch and Eklund 2012; Giaimo 2016), and whether the target users are consumers (B2C) or other businesses (B2B) (Rissanen and Münch 2015; Yaman et al. 2016). Also, only some companies are able to conduct experiments that affect business value (Schermann et al. 2018). However, there is a lack of overarching theory that describes which factors affect the efficacy of CE for different contexts. Thus, *the aim of our research* is to investigate if there are any common patterns that can explain the varying utility, challenges, and benefits (Auer et al. 2021) experienced by companies when applying CE.

We have performed a multi-case study of 12 companies to explore CE practices and contexts (Clarke and O'Connor 2012; Petersen and Wohlin 2009), and constructed an empirically-based theory through iterative and systematic analysis of this data. Our empirical data consists of 27 semi-structured interviews with practitioners working in various roles relevant to CE, such as software developers, quality assurance, data scientists, and product owners. We have previously published two initial findings based on parts of the interview material. First, a paper based on five interviews at one company in the e-commerce domain where we presented four conceptual scenarios in which experiments are used (Ros and Bjarnason 2018). Our analysis gave an indication that the context in which CE is performed matters. Second, a paper based on 14 interviews from five companies, where the concept of product-led growth in relation to continuous experimentation practices emerged in the analysis (Ros 2020). In this paper, with 13 additional interviews from seven companies added to our empirical foundation, we expand on the initial findings by deepening the thematic analysis and generating a theory of factors affecting CE, based on the full set of interviews.

The main contribution in this paper is a theory of Factors Affecting Continuous Experimentation (FACE). FACE considers socio-technical factors in organizational contexts surrounding CE and observed relationships between factors. According to FACE, there are three main

factors that influence how well an organization can expect to conduct CE. These factors are: (1) *CE processes and infrastructure*, in particular, the data infrastructure that companies have for conducting telemetry and analysis of results; (2) the complexity of the *problem* that the software solves for its users, limits the applicability of CE by making changes hard to deliver and observe the effect of on users; and (3) *incentive structures* in the business model that, e.g., affect the motivation for conducting product improvement or the ease of defining business relevant metrics for CE.

The rest of this paper is structured as follows. The background and related work on CE and theory building are presented in Section 2, and our research method is outlined in Section 3. Our FACE theory is presented in Section 4 and the empirical underpinnings and explanations for the theory are provided in Section 5. The findings are discussed in Section 6 including limitations and factors of CE. Finally, we conclude the study in Section 7.

2 Background and Related Work

Companies in many different sectors have adopted Continuous Experimentation (CE) (Auer and Felderer 2018; Ros and Runeson 2018), where features are evaluated through user feedback. Prototypes of a feature or product can be quickly validated with users before a costly implementation is finalized and released to all users. After implementation, the change in software can be subjected to a controlled experiment (such as an A/B test) where a comparison can be made with and without the new change. Only changes that have a positive impact on user feedback are accepted. The results of an experiment might beget further questions, especially for negative results, to figure out what went wrong. Thus, experiments are usually executed in a sequence, which is why the practice is coined *continuous experimentation*. Experiments in *simulated* (performed during prototyping) and *live* (performed after software delivery) environments are both of interest in this study. The literature topics covered in this section include software business models and how they relate to CE, an overview and related work of CE, and theory building in software engineering.

2.1 Software Business Models and Product Management

CE provides a way to measure the value that software development brings to users and businesses. As such, studying CE entails understanding how that value is created, delivered, and captured. In this study, we use business models and business strategy as a lens to structure our analysis of this value. A *business strategy* in the management field is a long-term vision for a company (Johnson et al. 2008). A business model is a concrete plan to execute that vision. The term *business model* is narrowly used in industry to refer to how a company collects revenue (Vanhala and Smolander 2013). In this paper, the broader definition by Osterwalder et al. (2010) is used: "*A business model describes the rationale of how an organization creates, delivers, and captures value*".

Similarly to business models, software product management also focuses on value creation through a combination of business and technical perspectives (Ebert and Brinkkemper 2014), though with a more distinct focus on the software product. Product management spans the entire software development cycle, from ideation and requirements, to delivery and maintenance, and interfaces both business processes and stakeholders such as customers and users. A product manager is a prime driver for requirement engineering for a software product or

service, with the aim of maximising the business value over time, while aligning with the company's business strategy and technical roadmap.

In the business model innovation field, experiments have been studied as a method to find a combination of a working business model and product (Brunswicker et al. 2013; Sorescu 2017; Wrigley and Straker 2016). This contrasts somewhat with the way experiments are used in CE research, where the focus is more on product improvement (Fagerholm et al. 2017) and where changing the business model might be considered out of scope for daily software engineering work. In addition, successfully implementing changes in an established business model is notoriously hard and risky (Chesbrough 2007; Chesbrough and Rosenbloom 2002). As such, business models are mainly used to describe the context of companies in this study, not as a subject of experiments.

There have been several attempts to describe commonalities in software engineering business models with frameworks (Rajala et al. 2003; Schief and Buxmann 2012). For example, Rajala et al. (2003) include aspects of product strategy, revenue logic, distribution model, and service and implementation model. The business model canvas (Osterwalder 2004) is probably the most popular framework to describe business models succinctly. We use an adaption of this framework as described in the next subsection.

2.1.1 Lean Startup

Lean startup is a methodology, originating in industry from Ries (2011), that applies lean manufacturing principles (Krafcik 1988) to entrepreneurship in general. Lean startup has also been studied in a software engineering context (Bjarnason 2021; Bosch et al. 2013; Fagerholm et al. 2017). The idea is to conduct product development in short cycles to obtain feedback as early as possible on whether a proposed business model is feasible and viable. Ries calls it the build–measure–learn cycle, where each cycle consists of a business hypothesis and an experiment to validate the hypothesis. As such, CE and lean startup have a clear connection.

Lean startup advocates exploring the solution domain through simulated experiments and testing early versions of the solution on a limited set of customers, starting with simple prototypes, e.g. sketches and mockups, to learn about customer needs in a cost effective manner and to validate solution ideas before building them (Bjarnason 2021; Gutbrod et al. 2017; Vargas et al. 2020). The goal is to identify a minimum viable product (MVP), which is the smallest set of features that solve the users' problems.

Maurya proposes an adaptation of the business model canvas to suit lean startup needs, called the *lean canvas* (Maurya 2012). The lean canvas is divided into a product and a market part. The product part contains the problem–solution pair that the product addresses, what key metrics are measured, and what the cost structure is for acquiring customers, developing code, operations, etc. The market part contains what the unfair advantage is, such that the product cannot be easily copied, what the channels to customers are, what the target customers are, and the revenue streams. The two parts are tied together with a value proposition message.

Maurya (2012) also describes the three phases of a startup that (1) start with finding a working problem–solution pair, i.e. problem–solution fit, (2) then identifying and validating that there is a good fit between the product and the market, and finally (3) once product–market fit is validated, the business can focus on growth. Experiments are often used differently at these stages where the initial focus is prototyping with simulated users and or product and later on live experiments can tune the product for product–market fit and growth.

2.1.2 Product-Led Growth and Growth Engineering

A specific archetype of business models has recently been popularized in industry, under the name of *product-led growth* (Bartlett 2020). A business model that has a product-led growth relies on the product itself to acquire new end-users rather than on the direct sales & marketing activities (e.g. advertisement or cold calling) of the sales-led business models. The purpose is to have an offering that can scale to high levels of demand. Furthermore, a new role has also been introduced to business with product-led growth, described by Kemell et al. (2019); Troisi et al. (2020), called growth marketers, growth engineers, or growth hackers. These roles are hybrids between marketers and software engineers that work with experiments in a data-driven fashion to propel a company's customer acquisition growth. Two of the companies in our study have employees with such titles.

A more precise definition of product-led growth—as the term is used in industry—has not been found. Instead the following typical characteristics are derived from Bartlett (2020):

- the software development organization elicits requirements in order to meet market needs;
- there is no customer specific development in order to ensure software development is directed towards improving the product for all users;
- the channels to acquire customers are scalable to many customers and are often organic (i.e. word of mouth instead of direct sales);
- the primary source of revenue is through product sales or subscriptions.

The last of the above characteristic, regarding the source of revenue, is related to the licensing model that software is sold under. According to Bartlett (2020), product-led growth is associated with *freemium*. A freemium product (Niculescu and Wu 2014) is available both for free and as a paid premium version. The premium version might, e.g., have more features or offer improved customer support. As such, freemium encourages growth by allowing more customers to use the product and spread the word.

2.2 Continuous Experimentation

CE has been studied from many different perspectives (Ros and Bjarnason 2018). In software engineering venues, the topics have been varied, e.g., designs of specialized tools for optimizing experiments (Ros and Hammar 2020; Schermann and Leitner 2018) and descriptions of the CE process in use at various companies (Bosch 2012; Fagerholm et al. 2017). Although software engineering is the focus of this study, there has also been considerable practitioner-focused research on CE in data science and user experience research venues. In the data science field, the seminal paper by Kohavi et al. (2009) provides a practical guide to starting with live experiments on software in a production environment, using randomized controlled experiments (e.g. A/B tests). In user experience research (Sauro and Lewis 2016; Schumacher 2009), live experiments has a less prominent role in favor of simulated experiments, such as user observations, card sorting, or interviews.

2.2.1 Process and Infrastructure Models

The infrastructure and process of how several different companies conduct CE have been described with reference models and experience reports (Bosch 2012; Feitelson et al. 2013; Kohavi et al. 2009). The models are conceptual generalizations based on observations of CE

in industry, but they do not explain the underlying factors behind the generalizations, which is the purpose of FACE. The RIGHT model by Fagerholm et al. (2017) contains a description of the process and infrastructure needs for conducting CE based on a multi-case study. The earlier HYPEX model by Holmström Olsson and Bosch (2014) has similar goal but is less comprehensive. These reports and models served as a starting point for the questions in the interview guide in this study, in particular the RIGHT model.

The process model in RIGHT is inspired by the build–measure–learn cycle of lean start-up Ries (2011). Fagerholm et al. use the concepts of a minimum viable feature (MVF) to bridge the theoretical gap between prototyping simulated experiments and live experiments, and so the model considers both types of experiments. There are five main phases of the CE process in RIGHT. (1) In the *ideation* phase hypotheses are elicited and prioritized and a change to the software is proposed. (2) *Implementation* of the minimum change that tests the hypothesis follows. (3) Then, a suitable *experiment design* and a criterion for success is selected (a metric in the case of a live experiment). (4) *Execution* involves deploying the product into production and monitoring the experiment. Finally, (5) an *analysis* and decision is made whether the results are satisfactory; if not the process restarts.

The technical infrastructure needs in RIGHT include tools for managing experiments and analytics, instrumentation in the product, and a continuous delivery pipeline. These tools are often referred to as an *experimentation platform* when considered as a whole, e.g. by Gupta et al. (2018) and Kohavi et al. (2009). The organizational infrastructure includes roles[1] involved with CE, of which there are many, since the CE phases cover the whole software engineering process. The necessary roles are according to Fagerholm et al. (2017): Business Analysts and Product Owners elicit hypotheses and maintain a CE road map; Data Scientists design, execute, and analyze experiments; Software Developers and Quality Assurance develop and verify the software; and Operations Engineers and Release Engineers deploy and deliver the software.

2.2.2 Factors Affecting Continuous Experimentation

We are aware of one other attempt at analysis of how effective CE is for various companies, albeit from a startup perspective. Melegati et al. (2022) studied factors affecting CE in terms of enablers and inhibitors of CE at early-stage startups. Many of the identified inhibitors point to a lack of resources to conduct experiments, which is more pressing for startups due to a general lack of resources. Also, the research only considers pre-deployment prototype experiments, which is much less technically demanding.

There are also many studies that describe or report experiences about conducting CE for various circumstances. We have identified five such clusters of papers with domain specific challenges in a systematic literature review (Auer et al. 2021). Two of the clusters relate to factors that might influence the gains with CE: business-to-business (B2B) and cyber-physical systems. The challenges in the remaining three clusters are either overlapping (mobile and cyber-physical systems) or potentially solved by statistical solutions (e-commerce and social media).

The challenges with CE in the *business-to-business* (B2B) cluster are many (Rissanen and Münch 2015; Ros and Bjarnason 2018; Yaman et al. 2016). B2B companies are usually involved with their customers' software engineering or IT departments, which causes issues

[1] The titles of roles associated with the tasks in RIGHT vary greatly among companies, e.g., the Business Analyst role can also be performed by an individual with a title of growth engineer (Troisi et al. 2020) or user researcher (Schumacher 2009).

with control of software deployment or with access to end-user data. Both of these are necessary and require constant collaboration efforts to resolve. Furthermore, the incentives to improve the software product in terms of end-user experience might not be there, e.g., if the company in question generates revenue per project instead of through the value delivered to users. These issues might not be faced to the same degree by companies with a business-to-consumer (B2C) business model.

For *cyber-physical systems* (Bosch and Eklund 2012; Giaimo 2016; Mattos et al. 2018; Olsson and Bosch 2019), the focus of the research is on suggesting and describing the required infrastructure to enable CE, such as continuous delivery of software and how telemetry can be implemented. Without this, CE is not as useful. The body of work on these domains is still in the early stages and no such companies are included in this study.

2.2.3 Previous Work

This work is the culmination of a research project with two prior publications (Ros and Bjarnason 2018 and Ros 2020). The findings from these papers are synthesized and expanded upon in FACE.

The first paper (Ros and Bjarnason 2018) was a single case study on a B2B company that develops behavior algorithms and used CE in four different scenarios: (1) to verify that changes in their algorithms are beneficial; (2) to help their customers use their algorithms correctly; (3) to validate that their algorithms outperform their competitors'; and finally, (4) as a black box optimization method to tune the algorithms automatically (Brodén et al. 2019). The degree of tool support was different between the scenarios, ranging from fully automated to no support. The scenarios give an early indication on how CE differs depending on the purpose.

The second paper (Ros 2020) used five of the 12 cases that this study is based on in a comparative case study. The study was about the role that a company's business model plays in relation to CE. The differences attributed to business models were sufficiently explained by whether the business model had product-led growth or not. Four drivers of a product-led growth focus were identified as affecting CE: (1) development and sales & marketing worked more closely together in the product-led cases, with mutual benefits; (2) the prioritization process used data to a higher degree to inform development; (3) what features were included were based on market needs rather than by customer requests, thereby decreasing excessive feature bloat; and finally, (4) the availability of metrics relevant to business was higher in the product-led cases.

2.3 Theory Building

Theories provide a means of structuring and conveying knowledge in a condensed form. Theories can be of different types, some provide explanations of phenomena or predict the outcome, e.g., of applying a certain practice. Theories that describe a particular aspect of software engineering are gaining traction (see, e.g., Rodríguez et al. (2022) or Munir et al. (2018)), presumably due to their usefulness in supporting research design and on improving software engineering practice. According to Stol and Fitzgerald (2015), it is preferable to base subsequent practitioner guidelines on theories such that they are underpinned with theoretical knowledge on why they hold.

We used the guidelines by Sjøberg et al. (2008) to build our theory based on a generalization of multiple cases observed in the real world. There are other theory building approaches (Wieringa and Daneva 2015), such as basing them on existing theory from other fields or grounded theory (Glaser and Strauss 1967; Stol et al. 2016), which is sometimes used when the researchers want no preconceived notions about how the data should be interpreted, etc.

We have framed our theory by applying Sjøberg et al.'s first four steps for theory building, namely defining *constructs*, defining *propositions*, providing *explanations* and by determining the *scope* of our theory. The fifth step of *testing* the theory through empirical research remains as future work. According to Sjøberg et al. (2008), a theory consists of *constructs* about which the theory makes statements in the form of *propositions* that describe relationships between constructs. In addition, a theory also contains *explanations* about why the propositions hold and a *scope* within which the theory is valid. We use this structure to frame our theory in terms of constructs, propositions, and scope, see Section 4, and by providing explanations for our theory grounded in the empirical data, see Section 5.

3 Method

The current study aims at investigating which organizational factors influence an organization's ability to gain from CE. We therefore launched a *multi-case study* of companies applying CE, followed by a *theory building* process, aimed at providing a generalized description of factors affecting a company's ability to obtain gains from CE. The resulting theory was inducted through iterative analysis of the interview data. An overview of our research method is provided in Fig. 1. The empirical data in our study is from case companies and we use the guidelines by Runeson and Höst (2009) for case study research, covering the interview process and thematic coding. The theory was then created through the codes and themes, based on the process for building theories by Sjøberg et al. (2008) which includes defining *constructs*, *propositions*, *explanations* and *scope of validity* for the theory. The theory is based on data from 12 case companies, denoted A–L, see Section 3.3.

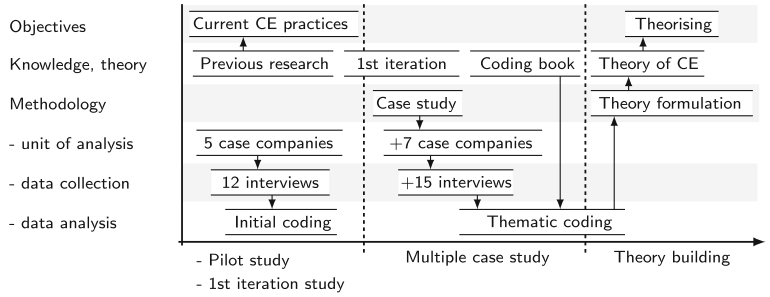

Fig. 1 An overview of the research method of the research project culminating in FACE. Two previous studies (to the left) Ros and Bjarnason (2018) and Ros (2020) were used as a starting point for the theory building

3.1 Multi-Case Study with Interviews

We performed *semi-structured interviews* in order to gain insights into what and how contextual factors affect an organization's ability to perform CE, and thereby provide a rich empirical basis for our theory building process. We designed an *interview guide* based on previous related research by Fagerholm et al. (2017), Kohavi et al. (2009), and Olsson and Bosch (2014), *sampled and recruited companies and practitioners* to include in our study, performed the *interviews* of these using an interview guide, and *analyzed* the interview data using a code book. The first author led the interview study, including the design of the interview guide and the code book, recruiting of the interviewees, performing and analysing all of the interviews. The second and third authors reviewed and provided feedback on the research design and the research artefacts, provided feedback on the interview guide, participated in two interviews each, and performed independent coding of one interview to improve the coding process.

An *interview guide* was used to support the interviews and to ensure that all relevant aspects were covered in each interview, namely: interviewee information, case company context, CE process and infrastructure, experimentation details, and the interviewees views on CE. The guide was designed with probes such that it could be adapted to the interviewee's background and role. The guide was based on our knowledge of the area and of previous research on CE, in order to cover as many relevant aspects as possible, and thereby further enhance the richness of the resulting empirical data. The descriptive models by Fagerholm et al. (2017) and Holmström Olsson and Bosch (2014) were used as the theoretical basis to derive the interview questions. Additional descriptions of CE from experience reports (Kohavi et al. 2009; Bosch 2012) were used to ensure that the questions covered all aspects of the CE process and infrastructure, and the context around CE in terms of the organization and business.

The interview guide was designed iteratively in the research project (see Section 2.2.3) by the authors with small updates after each initial interview. The first version of the interview guide is available in the pilot study by Ros and Bjarnason (2018). Note that, the analysis for the first iteration study (Ros 2020) was conducted after all interviews were conducted, but included only five of the 12 cases. The last 15 interviews (G2–L2 in Table 2) used the final version of the interview guide, which is provided in Appendix B.

We *sampled and selected* case companies through a mix of convenience sampling and snowballing guided by the aim of our study. We selected companies that had any experience in conducting experiments. In three of the cases we selected a branch of the company as the case (cases D, I, and G). Candidate companies were identified through searching on LinkedIn and Google for job listings for data scientists where A/B testing was mentioned, through personal contacts, and through asking the interviewees (i.e. snowballing) for other companies involved with CE; these methods contributed roughly equal to the final tally of interviewees. The cases were not restricted by geographical location; though, nine of the companies are from Sweden and the rest from North America, Australia, and Europe. Our aim was to interview practitioners with insights and experience of applying CE, and primarily in the roles with heavy CE involvement: software developers, product owners, and data scientist. We applied snowball sampling also within the companies by asking for additional interviewees within the companies during the initial interviews. The managers closest to CE were contacted via e-mail with information about the study and a request to perform interviews. In total 35 companies were contacted of which 12 were included in our study. Of the remaining, 11 did not respond and 13 were not applicable due to having no experimentation experience.

The *semi-structured interviews* were held during an initial 3 month period for the pilot study, which was then continued a year and a half later for another year. Interviewees were selected from the case organisation until a complete picture of how and why they used CE was obtained. Albeit in two cases (B and E) the process was completed early due to the subsequent interview prospects not wanting to be interviewed. Each interview was approximately 60 to 120 minutes long and was held as an open conversation aligned with the interview guide (see Appendix B). The interviews were audio recorded after permission for this was granted by the interviewees. The majority of the interviews were performed at the companies' premises. For six companies, this was not feasible due to their location, in which case the interviews were held on-line via Zoom or Skype. After the interviews, the audio recordings were transcribed word-by-word into 231 pages. Some of the interviews were done in Swedish and quotes from those interviews were translated to English.

Thematic coding was performed on the interview transcripts using the pre-defined codes of our code book. The code book was defined based on previous knowledge and insights into CE, from primarily three core publications (Bosch 2012; Fagerholm et al. 2017; Kohavi et al. 2009). It was iteratively refined throughout the coding process by discussing these codes within the group of authors. The codes were added per paragraph of transcribed text, and each paragraph could have multiple codes. The codes were clustered into 11 themes according to thematic analysis. The codes cover areas such as the company context, management of data, the business model, product, and CE process for the company, see Appendix C. After having coded the full set of transcripts, cross-case analysis was performed to identify factors and patterns common to several cases. The theory was gradually inducted as this cross-case analysis matured, see below.

3.2 Theory Building

We have built the theory by analysing our empirical data and gradually defining constructs, propositions, the scope and explanations for the theory as proposed by Sjøberg et al. (2008). The final step of theory building, namely testing our theory, remains as future work. The first author defined the initial version of the theory through analysis of the empirical data. The theory was then refined by the first and the second author through multiple iterations of review and discussions. As the theory matured, it was reviewed by all three authors and further improved.

The *constructs and propositions* of FACE represent the factors that affect CE and the relationships between these factors, relevant to CE. The constructs were identified through analysis of the thematically coded interview data. The themes from the thematic analysis formed the first iteration of defining the constructs of the theory. An additional coding step was carried out to find relationships between constructs within the transcripts. When such occurrences were found they were coded with codes consisting of theme pairs. Initial versions of propositions were defined based on these theme–pair codes. The constructs and the propositions were then gradually refined and adjusted through discussion within the team of authors, to provide a clear and concise description of the factors that affect a company's ability to draw benefits from CE. Describing the relationships between the constructs through visualisation (see Fig. 2) and written definitions (see Sections 4.2 and 4.3) facilitated this inter-author discussion. The supporting evidence in the material was used as a basis for the discussion. In total, 10 iterations of the theory were discussed in this way.

The *scope* and *explanations* of FACE were identified based on our empirical data. The *scope* of our theory was defined through considering the common characteristics of our

case companies, and thus the type of organizations that our theory may be applicable to. Explanations for our theory are provided as part of the definition and description of each construct and proposition.

The *empirical underpinning* of FACE provides a motivation for the concepts of our theory grounded in the empirical data, thereby illustrating the empirical foundation for the constructs and propositions. In Section 5, supporting evidence in the material for each proposition in the theory is laid out along with expanded explanations. The empirical underpinning in terms of the constructs and propositions provides an *initial validation* of FACE, and illustrates its utility and explanatory power. As such, the empirical data is used both as a source and validation, by having the data used at the detailed level to construct the theory and then at the holistic level to describe our set of companies.

3.3 Case Companies

The twelve case companies in the study differ on many attributes such as size, product domain, business model and CE practices. Our case companies range from small to huge multi-national companies. Some of the companies work extensively with CE while some do next to no experiments at all.

Table 1 contains an overview of the business models and states of CE practices for the 12 case companies. Each case company is ranked according to its *expertise* with CE and the *extent* to which the company conducts experiments (both frequency of performing experiments and on what parts of a product or service that are experimented on). The business model is given as a brief summary for each company, primarily describing the company's offering and what type of customers that are targeted; either business-to-business (B2B), business-to-consumers (B2C), or both (B2X). Under the business model column, *direct sales* refers to having a business model where licenses to the product or service are sold by contacting customers directly. See Appendix A for further details on specific companies.

The 27 interviewees at the case companies are described in Table 2, which show their code corresponding to the case company and role name. Two additional roles have been observed in the interviews in addition to the ones presented in RIGHT (Fagerholm et al. 2017) (see Section 2.2), namely, the *User Researcher* and *UX Designer*, These roles have similar responsibilities within CE to that of data scientists and software engineers, respectively. That is, a UX designer or software engineer comes up with a change in user experience or software, and a user researcher or data scientist analyzes the results. The user researchers in the interviews primarily used prototyping qualitative experiments. The actual titles differ significantly from their assigned role and the titles in use include, e.g., head of growth, software engineer, head of customer success, growth engineer, head of research, etc.

4 Theory Formulation of FACE

Our FACE theory describes factors that affect continuous experimentation (CE) and how these factors contribute to an organization's ability to gain value through CE. The gains are achieved through effectively conducting experiments that enable improving the problem–solution fit (**P2**) and/or the product–market fit (**P3**). In essence, the theory states that the effectiveness of experiments can be increased through efficient processes and tools for CE (**P1**), addressing a user problem that is sufficiently simple to be measurable (**P4**), pivoting the business model to simplifying said problem complexity (**P5**), and/or selecting a business model that provides

Table 1 Overview of our 12 case companies, containing business model, size and age (rounded to nearest 5 year), and state of CE

Case		Business Model				CE	
		Summary	Type	Size	Age	Expertise	Extent
A[a,b]	E-commerce algorithms	Product with direct sales	B2B	Small	20	High	Medium
B	Local search	Service selling placement to B2B customers and serving ads to B2C	B2X	Small	15	Medium	Low
C	E-commerce consultants	Consulting on a per project basis	B2B	Medium	20	Low	Low
D[b]	Video streaming	Freemium service with premium features	B2C	Medium	10	High	Full
E	Web shop	Web shop for subscription to physical goods	B2C	Huge	10	Medium	Medium
F[b]	Customer relations	Product and service with direct sales	B2B	Medium	30	Low	Low
G[b]	Engineering tools	Freemium service with premium for larger teams	B2B	Huge	20	High	Full
H	Web shop	Retailer with small web shop	B2B	Large	20	Low	Low
I	Web shop	Web shop and retail	B2C	Huge	20	High	Medium
J	Product information	Service selling customer information and leads to B2B and free for B2C	B2B	Medium	10	Medium	Low
K[b]	Business intelligence	Product with direct sales	B2X	Large	30	Medium	Low
L	Employee management	Service with direct sales	B2B	Medium	20	Low	Medium

Small companies have less than 50 employees, medium have less than 250, large companies have less than 1000 employees, and huge more than 1000 employees. CE expertise and extent is estimated based on our interviews using the ordinal scale: low, medium, and high

[a] Participated in pilot study Ros and Bjarnason (2018)

[b] Participated in 1st iteration study Ros (2020)

Table 2 Overview of the 27 interviewees at the case companies

Code	Role(s)	Code	Role(s)	Code	Role(s)
A1	Software Developer and Data Scientist	D2	User Researcher and Software Developer	I2	Product Owner
A2	Product Owner and Data Scientist	E1	Data Scientist and Software Developer	I3	Data Scientist
A3	Software Developer and Release Engineer	F1	Product Owner and Business Analyst	I4	Business Analyst
A4	Product Owner and Business Analyst	F2	Product Owner and Software Developer	J1	Product Owner
A5	Operations Engineer	G1	Quality Assurance	J2	Software Developer and Data Scientist
B1	Software Developer and Data Scientist	G2	Product Owner	K1	Product Owner and UX Designer
C1	Software Developer	G3	Data Scientist	K2	User Researcher
C2	Software Developer and Quality Assurance	H1	Software Developer	L1	UX Designer
D1	Data Scientist and Business Analyst	I1	Software Developer	L2	User Researcher

The codes of the interviewees correspond to the case organisation that they belong to, i.e., A1–5 to Case A, B1 to case B, and so on. The role is assigned by the authors to best describe their primary role(s) in CE—not their title

incentives to conduct experimentation (**P6**). An overview of our theory is provided in Fig. 2. In this section, the theory is defined by describing its constructs, propositions, scope, and its validity is discussed. Expanded and empirically-based explanations of the theory are provided in the succeeding Section 5 by exemplifying the constructs and propositions as observed in our 12 case companies.

4.1 Definitions

The following terms are used to describe the constructs and propositions of FACE theory:

Experiment is an activity that introduces a *change* in a *business offering* with the *goal* of learning or improving the offering based on feedback through *user data*. The term includes *live experiments* where the experiment is executed with real users (e.g. A/B tests or natural experiments) and *simulated experiments* where the change or users are simulated. There must always be a way to decide if the *goal* of the experiment is met or not, otherwise it is not an experiment by our definition.

Live experiments are executed in a post-deployment production environment with real users. *Randomized controlled experiments* are the typical example and can be used to give precise answers to whether experiments provide a quantitative improvement to some metric. Live experiments have by definition higher infrastructure needs than *simulated experiments*.

Simulated experiments are executed pre-deployment, outside of a production environment (e.g. with prototypes) or without real users (e.g. by using feedback from employees). The fidelity of a prototype can be anything from a hand-drawn sketch to almost completed functionality. It is expensive to get sufficient numbers of users to evaluate a prototype quantitatively in a simulated environment, so qualitative data is often used instead.

Continuous Experimentation (CE) is the process of continuously using experiments. This process encompasses the whole software engineering process and involves multiple experiments conducted in iterative cycles. Experiments are *continuous* for two reasons. First, an initial *experiment* is not always decisive, and many experiments in a row might be needed to refine a *change*. Second, the experiment result might uncover new knowledge about the business offering, market, or users that begets further related inquiries or even new features.

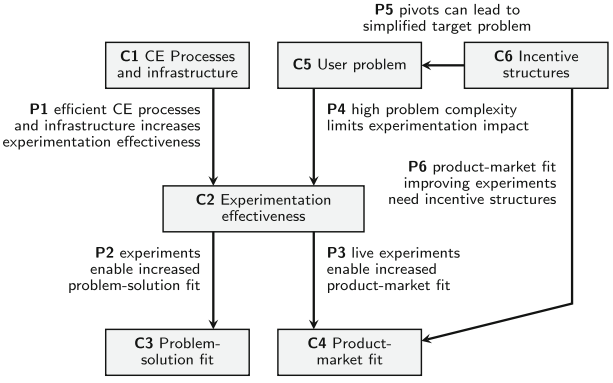

Fig. 2 The constructs and propositions of the Factors Affecting Continuous Experimentation (FACE) theory. The boxes represent constructs and the arrows are propositions

Experimenter is a person initiating and or being in charge of an experiment. It is not a formal role in the RIGHT model (see Section 2.2), but would usually correspond to the data scientist or business analyst.

Change refers to a modification to a *business offering*. It can be a new functionality (i.e. a *feature*), or a modification to an existing one, or a change to the quality of the software. The change can be in any part of the software and they can be subtle for users to perceive. The change does not need to be fully implemented, nor delivered to all users, as is the case in *simulated experiments* with prototypes. The scope of the changes we have observed in the interview material varies, from a small tweak of the font-size on a button, to rebuilding a large part of the product[2]. For some companies a change corresponds to a commit in a version control system (Kevic et al. 2017). A change corresponds to a *treatment* in traditional statistics or medicine literature.

Goal is what an experimenter wants to achieve with an experiment. In a live experiment with quantitative data, the goal can be described with a measurable improvement to a *user experience* or *sales metric*. The term *goal* is preferred in this work, since the more specific term *hypothesis* implies a deeper thought behind an experiment, which is not always the case. For example, when Google tested 41 shades of blue for their links (Holson 2009), the goal was to find the colour that got the most clicks, but the generalized learning obtained is limited since the hypothesis was not based on theory on how and why colours affect behaviour.

Business offering is a catch-all phrase referring to either a product, service, web shop, etc. This term is used when the distinction between these are not important.

User data refers to data obtained from actual users of the business offering. User data could also be obtained through other means, such as questionnaires, interviews, or by eye-tracking. There are two categories of metrics derived from user data: (1) *user experience metrics* and (2) *sales metrics*.

User experience metrics are derived from *user data* that measure the users' experience, such as degree of users that engage with the software during the experiment, time spent on parts of the software, rate of users that complete a certain task etc.

Sales metrics are derived from user data originating from the sales process of the offering. The sales metrics include revenue from sales or subscriptions, churn of subscribed users, conversion rate of users to paying users, etc.

Proxy metric is a metric that substitutes for another more relevant metric that cannot be used directly in an experiment for some reason. For example, consider an e-commerce web shop with insufficient traffic to obtain statistical significance on conversion rate or other sales figures. They could use clicks (i.e. a user experience metric) as a proxy metric for purchases (i.e. a sales metric) since far more users click on products than buy them. However, the signal-to-noise ratio would be much lower and an increase in clicks could even lead to a decrease in purchases if the user experience becomes more complicated as a result of the change (requiring more clicks). As such, the assumptions made when choosing a proxy metric should be continuously verified.

Business model is the way a company has structured their activities, revenue streams, costs, etc., to obtain and give value to users and or business, see Section 2.1.

Pivot is a major change in the business model done in order to better realize the business strategy (Chaparro and de Vasconcelos Gomes 2021; Kirtley and O'Mahony 2020). This may involve *changes* to the *business offering* (which is part of the business model), but a

[2] Many authors caution against large changes in an experiment to minimize risks of errors, e.g. Fagerholm et al. (2017); Kohavi et al. (2009).

pivot could also entail changes to, e.g. revenue model or sales channels. The distinction we draw between *changes* and *pivots* is that pivots have a larger scope and are often not based on evidence to the same degree as a change involved in an experiment. A pivot can be done for various reasons, such as to find a different user base or to overhaul the business offering in a major way.

4.2 Constructs

FACE identifies six constructs as described below. These constructs are referenced by the propositions of the theory.

C1 *CE processes and infrastructure* are software engineering practices, enabling *continuous experimentation*. This is primarily technical infrastructure, such as an experimentation platform, but also organizational infrastructure that provides necessary competences and roles. Since continuous experimentation involves implementing changes in the software for a business offering, this construct also encompasses standard practices for efficient software engineering, such as software testing and quality assurance, while it is not the focus of this study.
C2 *Experimentation effectiveness* is the degree of effective continuous experimentation conducted in an organization. It is a function of the throughput of experiments, the impact the experiments have on users or business, and the ability to accurately measure this experiment impact. *Throughput* is derived by the *speed* of which experiments are executed from end-to-end and the *capacity* of how many experiments that an organisation can handle simultaneously.
C3 *Problem–solution fit* is the degree to which an offering provides a solution to users' problems. Problem and solution are part of the lean canvas business model (Maurya 2012; Ries 2011). There can be multiple problem–solution pairs in the same offering and a solution can address multiple problems. The degree to which the problem–solution fit can be measured depends on the specific offering but is usually measured with *proxy metrics* based on *user experience metrics*.
C4 *Product–market fit* is "*[a] measure of how well a product satisfies the market."* (Olsen (2015), p. 7). This construct describes the ability of a business offer to generate economic value (usually revenue from sales) and thus that there is a market for the offering. Product–market fit is advocated in lean startup as the ultimate target for business success for all companies (Maurya 2012), not just startups. Sales figures in the form of user retention, growth of the user base, etc., can serve as *proxy metrics* for product–market fit.
C5 *User problem* is what problem the users solve with the *business offering* and how the problem is solved.
C6 *Incentive structures* is the way an organization has structured rewards (or punishments) related to the business model and the value proposition therein. The construct primarily concerns the way in which users are enticed to pay for, and employees are enticed to improve, the *business offering*. This includes, e.g., the licensing and revenue model under which the software is sold and how the performance of software developer teams are measured.

4.3 Propositions

The propositions describe how the constructs influence an organization's ability to perform effective experiments (**P1** and **P4**) that in turn indirectly (**P5** and **P6**) or directly (**P2** and **P3**) affects the problem–solution or product–market fit of their business model through continuous experimentation.

P1: **C1→C2** *Efficient CE processes and infrastructure increase experimentation effectiveness.* Efficient CE processes brings: correct prioritization of experiments where the changes with the highest potential impact are selected, that the experimenter and the organization sufficiently trust the experiments so they can act on it, that the most relevant metrics are selected, etc. Efficient CE infrastructure include a CE platform that enables parallel experimentation, an advanced data infrastructure that can collect and analyze data in all parts of the *business offering*, etc.

P2: **C2→C3** *CE experiments enable increased problem–solution fit.* Experiments that target problem–solution fit aim to improve the users' ability to solve problems with the product. Either *live experiments* or *simulated experiments* can be used for this purpose.

P3: **C2→C4** *Live experiments enable increased product–market fit.* Product–market fit is more difficult to target than problem–solution fit due to having to affect the customers' overall purchasing intent. Thus, the users' problems need to be sufficiently solved *and* they have to be solved in a way that the customer want to pay for it. Experiments performed live on real users can measure product–market fit.

P4: **C5→C2** *High problem complexity limits experimentation impact.* Software that solves complex problems is inherently complex (Brooks 1986). This complexity manifests in difficulties to deliver changes, especially if it also affects users' current workflow. Also, complex problems are likely difficult to measure with a singular metric and to quantify into sessions of usage. Furthermore, *live experiments* have strict requirements and are thus affected by high problem complexity to a higher degree.

P5: **C6→C5** *Pivots can lead to simplified target problem.* CE usually involves making changes to the *solution* part of the problem–solution pair, i.e., the software. However, changes that address the problem is a bigger endeavour, because it will likely require involvement of the whole company (i.e., software engineering, sales & marketing, and management departments) and users. CE is not suited for such large changes because it is too costly to reverse in case of failure. Instead, a *pivot*, i.e., a change of direction in the business model that affects the value proposition and thus the product can be required to realize changes to user problem complexity.

P6: **C6→C4** *Product-market fit improving experiments need incentive structures.* Affecting product–market fit with CE is challenging, according to **P3**. Market considerations are outside traditional software engineering responsibilities and sales metrics might not be available for CE depending on the licensing and revenue model. As such, targeting product-market fit requires incentive structures to be in place with a link between business value, user value, and software engineering activities.

4.4 Scope

The scope within which FACE is applicable is *user-intensive software companies with a user-facing offering*, e.g., media content services (Case D), e-commerce web shops (Cases E, H, and I), and application software products (Case K). Not all companies in the study have software development as their primary activity, but software is a central part of their

business model. The theory is derived from empirical observations on those companies and it is unknown whether the theory is applicable or useful outside this context of software intensive companies, e.g., non-profit organizations developing open-source software, B2B companies without access to users, or companies that are involved with experimentation from a marketing perspective but that do very little software development (e.g., news websites or web shops without IT departments).

4.5 Theory Validity

We evaluate FACE using the criteria proposed by Sjøberg et al. (2008), namely testability, empirical support, explanatory power, parsimony, generality, and utility.

Testability. FACE makes high level claims about company processes, organization, etc. As such, the ability for research to set up randomized controlled experiments (Wohlin et al. 2012) to test the theory is very low due to the large scope—and consequently cost—that would be required of such an experiment. However, the theory makes claims about real world phenomena and those claims can be validated by using it in case studies (Runeson et al. 2012) to analyse and explain CE practice at other companies.

Empirical support. The theory is based on an extensive multi-case study with 12 cases of various contexts and 27 interviewees with various backgrounds and roles. The cross-section of empirical underpinning per case and proposition is shown in Table 3 below. Each proposition is supported by evidence from multiple cases, which is transparently reported in Section 4.

Explanatory power. While there is no ability to make quantitative predictions from FACE, the theory can be used to differentiate between different instances of CE in real companies and thus explain why some companies derive more value from CE than others. Descriptions of the context of all case companies, that the theory is based on, are also available (see Section 3.3 and Appendix A). These descriptions can be used by practitioners to compare with their own organization's context, and thus judge the applicability of the theory for that context, using theoretical generalisation (Runeson and Höst 2009). The theory also uses terms and concepts from established pre-validated knowledge, such as lean canvas from lean-startup (Maurya 2012; Ries 2011).

Parsimony. The number of constructs and propositions have been continuously reduced and combined during the theory building process, as described in Section 3.2. The remaining constituents of FACE are needed to explain the data from the cases.

Generality. The scope of the theory is user-intensive companies which limits the generality of the theory to such companies. The case companies on which the theory is based are quite different in terms of size, age, and domains (though e-commerce is over-represented) so the breadth of the scope is wide.

Utility. FACE and the underpinning case descriptions can be used by software organizations to understand their ability to perform CE and the factors that influence this. The interviewees in the study were generally very interested in learning more about CE, which hints at the overall industry relevance, in addition to the large number of industry authors active in the research about CE (Auer and Felderer 2018; Ros and Bjarnason 2018). To further the utility for practitioners, we plan in line with the advice by Stol and Fitzgerald (2015) to derive

guidelines from the theory, that show what state-of-the-art CE is and how companies can elevate their CE based on their context.

5 Theory Explanations and Empirical Underpinning

In this section, the supporting evidence from each of the 12 cases, for each of the six proposition in FACE is presented in order, along with an expanded explanation of the meaning and impact of each proposition. We refer to theory constructs as **C1**–**C6** and propositions as **P1**–**P6**, and *italicize* the key terms of our theory when mentioned in the text. Note that, only the salient evidence is discussed in the text. See Table 3 for a mapping of the supporting evidence per case and proposition. The strength of the evidence is judged qualitatively based on how much and how direct the propositions are discussed at the interviews. As such, the table does not reflect, e.g., how good or bad the cases are at conducting experiments (**P1**).

5.1 CE Processes and Infrastructure at the Case Companies (P1: C1→C2)

Companies in the study conduct and depend on CE to different degrees (see Table 1). The case companies also have matching degrees of processes and infrastructure support to match their level of CE (with some exceptions discussed here). All interviewees at companies with frequent CE mentioned efficient *process and infrastructure* (**C1**) as crucial for *efficient experimentation* (**C2**). The analysis on **P1** is divided into four parts: (1) CE processes and infrastructure efficiency at the case companies, (2) impact of efficient processes, (3) impact of efficient infrastructure, and (4) the impact of low throughput.

5.1.1 CE Processes and Infrastructure Efficiency at the Case Companies

The four companies with High CE expertise in Table 1, cases A, D, G, and I, also have *efficient processes and infrastructure* (**C1**) to support their CE. *Case A* has low demand on experiment

Table 3 Cross-section of empirical underpinning per case and proposition in FACE. Cells are marked View for proposition–case pairs with strong evidence and View for pairs with only some evidence

Case		Proposition					
		P1	P2	P3	P4	P5	P6
A	E-commerce algorithms	✕	✕		✕	╲	✕
B	Local search	╲	✕	╲			╲
C	E-commerce consultants		╲		✕	╲	╲
D	Video Streaming	✕	✕	✕	✕	✕	✕
E	Web shop	✕	╲	╲			╲
F	Customer relations		╲		✕	╲	✕
G	Engineering tools	✕	✕	✕			✕
H	Web shop	╲	╲		╲		
I	Web shop	✕	✕	✕			╲
J	Product information		✕	╲			╲
K	Business intelligence	╲	✕		✕	╲	✕
L	Employee management	╲	╲		✕		╲

throughput and instead focus their efforts on advanced statistical techniques and speed of CE. *Case D* and *G* do experiments on all their parts of their software products and have both advanced techniques and a streamlined processes for CE. *Case I* has similar advanced CE at some teams in the company (though the experimentation expertise varies greatly at the company between teams). As expressed by Interviewee I4 who's team conducts large amounts of CE: *"If you have a team that has [CE] in its soul, then you want a streamlined process for how to conduct experiments."* Developers in *Case I* should be able to put an idea under test within hours of its inception and have experiment results a few days later. This team struggled initially with their commercial-off-the-shelf CE platform used throughout the company because the data volume overwhelmed it, until they built their own.

Two case companies in the study, *Case E* and *I* in particular in some of their teams, struggled with implementing the *processes and infrastructure* (**C1**) with the efficiency that they desired. *Case E* has an ad-hoc software engineering process which causes issues for their CE. According to Interviewee E1, there is a constant change of direction from management in prioritization and what metrics to optimize for, incorrect CE execution such as stopping experiments too soon or not using statistical tests, and management not taking account of CE results. This often led to *ineffective experimentation* (**C2**). Some of these issues could also be attributed to organizational culture, but the issues manifested as a lack of adherance to *processes* (**C1**). At *Case I* two of the teams (responsible for search engine and recommendation engine, respectively) achieved *efficient CE* (**C2**) by having their own purpose built CE platform for their part of the product, as described above. The other software engineering teams at *Case I*, about 50, used a central support team with a data science focus that conducted the CE independently. This was described as *"totally unreasonable"* by I3 due to the low CE throughput, caused by both technical limitations in the commercial cloud-based CE platform (**C1**) and the overhead of having to involve another team.

5.1.2 Impact of Efficient Processes

CE requires following a rigid processes to ensure trustworthy results. This is important for experiments using both qualitative and quantitative methods. In experiments with qualitative data there is much manual and subjective work that must be conducted consistently across users and experimenters. For quantitative methods, the tools must be used correctly to get accurate numbers. In addition, there are many methods and techniques that can be used to get more efficient experimentation:

- Having *data-driven prioritization* ensures that the most important experiments are done first (done at *Case A, D, G, I, and J*). This could be done by analysing historic user data to see where users have issues, by surveying users, or by splitting *changes* into the smallest possible implementation (such that it can be aborted early in case of failure).
- Applying *data mining* after performing experiments can be used to get more information out of the experiment results, such as dividing the users into segments and analyzing whether the results differ in the segments (done at *Case A and G*).
- Making *power calculations* to figure out how many data points are needed in an experiment is a recommended step to do before the experiment is started (Kohavi et al. 2009). However, all interviewees except the ones at *Case G* admitted to never doing it or using the same calculation for all experiments.
- Using both *live and simulated experiments* is recommended. Many companies only have infrastructure for one type of experiment, but *Case D and I* have both and these companies are able to select the experiments that best suit the situation at hand.

– Optimizing the *statistical test* to a more precise version that suits the given situation increases chances of obtaining statistically significant results (done at *Case G and I*). However, Interviewee G3 also warned about spending too much time on this since it requires specialized knowledge and is technically challenging.
– Using *experiment designs* with multiple variables achieves more nuanced results, such as multi-variate tests or multi-armed bandits (done at *Case A and G*). This also requires specialized knowledge and is technically challenging.

The amount and quality of available *user data* was a frequent topic of discussion for all cases. User data is not an explicit part of FACE, but there are processes for adapting CE to low availability of users data. Both the amount and quality of user data has a direct impact on the *CE throughput* (**C2**). A certain pre-determined number of data points needs to be collected to get some degree of certainty in the results. Low quality on user data lowers the information that can be learned from each data point, thereby also lowering *CE throughput* (**C2**). User data is a limitation at companies with vast amounts of user data too, since some experiments only target certain users (e.g., that has a certain feature enabled or belongs to a certain user segment).

5.1.3 Impact of Efficient Infrastructure

Getting started with CE was not described as technically hard by any of the interviewees. As phrased by Interviewee G3, *"The original CE system was like 15 lines of Scala, it was just really really simple. It's interesting how you can start up really easy."* Also, the requirements for simulated experiments with qualitative data is even lower because the qualitative methods do not rely on technical infrastructure. However, as live experimentation is scaled up the demands on *infrastructure* (**C1**) increase to keep up with having *efficient experimentation* (**C2**).

When the case companies discovered the value of CE they steadily increased their frequency of CE to cover more of their new developments and also on old features that has never been subjected to experiments. The *infrastructure* (**C1**) is a bottleneck to enable increased CE but all case companies except *Case H* were able to keep up with increased infrastructure demands due to CE being prioritized in the organizations. *Case H* lack the resources needed to increase their infrastructure.

The overall *CE infrastructure* (**C1**) was similar at all cases, albeit at different levels of maturity. The infrastructure includes three parts. (1) *Data infrastructure* to store and provide query support for user data. This includes telemetry of user data and product data in all parts of the software, collecting various information about users for segmentation, and a system for storing and accessing this data (i.e., a data warehouse). In addition, all data needs to be stored securely and in compliance with legislation (e.g., GDPR). (2) A *CE platform* with support for starting and stopping experiments, configuring which metrics to target and what additional metrics to monitor, support for segmentation and arranging metrics in hierarchies if there are too many, alerting in case things go wrong, ways of running experiments in parallel (non-overlapping in case their changes are conflicting), etc. Finally, (3) *competences* to develop and support infrastructure and to conduct experiments (see Section 2.2.1). Also, the developers need to be educated in CE if they are to take part in it, which was a challenge for the cases where CE was scaled up, i.e. *Case D, G, and I*.

Of the three infrastructure parts, *data infrastructure* is the most demanding to develop, according to interviewees at *Case D, G, and I*, because data infrastructure must be implemented in the entire software offering, rather than as a standalone development. Investment

in data infrastructure is one of the reasons that *Case A* is able to conduct CE with *High expertise* despite their small company and software department size. The case company's product relies on user data for algorithms, such as recommender systems that need the same infrastructure, and the case company was able to use this infrastructure for conducting advanced experiments (multi-variate tests and multi-armed bandit variants, as described by Ros and Runeson (2018)) even though CE is not widespread at the company.

5.1.4 Impact of Low Throughput on Experimentation Efficiency

There are additional consequences of low throughput to the impact and *effectiveness of experiments* (**C2**), caused by either low capacity or low speed as follows, besides that companies fail to perform the desired amounts of CE.

A *low capacity* to run experiments increases the risk of releasing features with user-related issues, since only a subset of the desired experiments can be conducted. It is not always obvious what changes will have an impact on users or not. This was something all case companies experienced except the most advanced ones (*Case G and I*). As phrased by Interviewee A1: *"A/B tests are always unpredictable, that has been proven. Again and again by us. We do release some non A/B tested functionality and I can't be totally sure it is all good, but it is what it is. [...] We always focus on what customers need as much as possible and do our A/B tests on that."* Low capacity is primarily caused by lack of user data or developer resources.

Low speed of CE can be frustrating, as Interviewee E1 put it when queried about challenges of CE: "*long lead times make the whole being data-driven thing almost impossible*". We identified the following two additional consequences of low speed of CE. (1) Low experiment speed can result in other changes made to the product or market causing the invalid CE results. Due to, e.g., changes in company priorities (mentioned by E1 and J1), the experimenter forgetting details about the change (mentioned by A3 and D1), or conflicting software changes that make the change incompatible (mentioned by A3, A4, and K2). In addition, (2) it will be hard to use the insights to inspire future changes in the follow-up projects if they are started before the previous projects' experiment is completed. Thus, CE is not really *continuous* when experiments are slow (mentioned by B1).

The issues with lack of speed were observed at some of our case companies though caused by different factors. At *Case A* the lack of speed was due to challenges with continuous delivery and at *Case B* it was due to having experiments with very large scope. At *Case D and J* the lack of speed was caused by various inefficiencies in their process due to their recent start with CE. Finally, at *Case C and H* low user data volumes required long lead times for experiments, and at *Case K and L* the use of primarily qualitative methods with much manual work caused lack of speed.

5.2 CE impact at the Case Companies (P2: C2→C3 and P3: C2→C4)

All cases have experiences with using CE to increase *problem–solution fit* (**P2**). However, not all companies are able to affect *product–market fit* (**C4**). Only *Case D, G, and I* experiment with product–market fit regularly. *Case B, E, and J* target product–market fit only to some extent, *Case B and J* have multiple user groups of which they are only able to target one with experiments, and *Case E* have issues with *CE processes* (**C1**). The remaining cases do not target product–market fit due to *high problem complexity* (**P4**) and lack of *incentive structures* (**P6**).

Sales metrics (revenue, conversion rates, churn rate, etc.) were mentioned often in the interviews as an appealing metric to use for CE. As explained by Interviewee D1, such metrics are both directly relevant to business, and users presumably only pay for software that they think fulfills a need for them. As such, when an experiment can use sales metrics, CE can be used to optimize the product for both business needs and user needs simultaneously, thereby increasing *experiment effectiveness* (**C2**) and impact on *product–market fit* (**C4**).

According to **P3**, *live experiments can increase product-market fit*, while both simulated prototyping experiments and live *experiments can increase problem-solution fit* (**P2**). None of the case companies were able to target *product-market fit* (**C4**) with anything other than controlled experiments with sales metrics. Presumably due to that it is harder to measure *product-market fit* in a way that cannot be done with only a prototype since it requires users to actually pay for something, hence sales metrics.

5.3 Problem Complexity at the Case Companies (P4: C5→C2)

When the product or service aims to solve a *complex problem* (**C5**), it is challenging to *experiment efficiently* on it (**C2**). The problem complexity of the offerings at the case companies span a wide range, from low complexity (*Case D and G*), medium (*Case A and K*), to high (*Case C and L*). By problem complexity, we mean that there are various challenges in the way the offering delivers value to users. We identified three issues with how problem complexity affects CE: (1) problem complexity makes changes hard, (2) complex user experience makes measurements hard, and (3) configurability addresses problem complexity but splits CE effort.

Some degree of *complexity in the user problem* (**C5**) can be overcome by using simulated experiments with qualitative methods. User observations are regularly used at *Case K and L* and can be used even when the user experience cannot be quantified into meaningful sessions nor specified as an adequate metric. The CE process with user observations at *Case K and L* is slightly different than for a live experiment. There is usually no control group, instead the experimenter selects a small group of users and observe their interaction with the change in the product and compares with earlier results (as in a natural or quasi-experiments). As such, qualitative methods can evaluate a change and is considered to be a valid experiment. However, the efficiency of qualitative methods in terms of reliability per work hour is low due to the cost of interviewing, recording, coding, etc., compared to the cost and scalability of a live experiment once the infrastructure is in place. The ability to be precise in exactly which change has what effect is also lower compared to live experiments. Qualitative methods do have advantages with richer data that can be used to explain why changes fail or succeed. But as a method for conducting an experiment with realistic circumstances they are limited.

5.3.1 Impact of Problem Complexity on Making Changes

Some software offerings can be hard to implement changes on, due to having to be integrated with other software, causing communication barriers with their developers, or that there are requirements or expectations from customers that the software should not change. In such cases, the CE process becomes hard due to the complexity of implementing the change in the experiment. This affects the ability to conduct controlled experiments (**C2**).

When experimenting with software with integration needs, the experimenter has to interface with software that is external to the organization. The experimenter cannot make changes incompatible to it, or they have to communicate with the software engineering organization

responsible for the integrated system, which would cause delays in CE. At *Case C* their business is centered around integrating different systems to build web shops, Interviewee C2 mentions this regarding their customers' ability to A/B test on their site: *"What customers can change is actually only content [text and images]. [...] Supposedly, at best, the customers can A/B what it looks like [user interface]." Case H* is one of their customers that did do their own A/B testing with involvement from Interviewee C1. At *Case A*, where their product is used in other companies' software, the interviewees complain that they cannot *change* how their product is being used (sometimes incorrectly) and that deploying the changes takes considerable time for some customers.

Finally, *user or customer expectations or requirements* can hinder CE because some users might not want changes. At most of the case companies, the users are able to overcome changes easily, so this aspect of *problem complexity* (**C5**) was only discussed by interviewees at *Case D, K, and L.* At *Case D* they mention how their B2B product has customer requirements that mandate the behavior of certain features. It is not possible for them to make changes to these features even if it would be beneficial to their other customers. Of the other B2B companies in the study, only *Case D and F* accept customer requirements to their product in this way. At *Case K and L*, the interviewees mentioned that their users do not want frequent *changes* due to their software being used in a corporate setting where users do not want changes to disrupt their work flow.

5.3.2 Impact of Product Complexity on Defining Measurements for Controlled Experiments

The user experience of software with a complex and open-ended work flow cannot be neatly quantified into chunks or summarized with a single metric. Sales metrics can sidestep the issue of defining a user experience metric, but not all companies can use it for experiments directly and it might not be a suitable target for all experiments. *Case F, K, and L* cannot use sales metrics at all due to lack of incentives given by their business model, thus **P6** is not in play, and they also have *complex and open-ended user experiences* (**C5**). All of the other case companies mention using metrics that are related to the user experience as a target metric to improve *problem–solution fit* (**C3**). User experience metrics are also monitored on experiments that target *product–market fit* (**C4**) to ensure that business value does not come at the expense of users at all cases able to target product–market fit.

Case F, K, and L have similar underlying reasons for having a *complex and open-ended user experience* (**C5**) as explained in the interviews. The software is used throughout the day, such that it cannot be neatly quantified. The goal that users have when they use the software is hard for the software engineers to extract and measure, and the number of features is so large that there might not be enough user data on some features to measure it.

While it is always possible to find *something* to measure in the user experience of all software, such as number of clicks or other user interactions, it is far from certain that optimizing those metrics will transfer to concrete gains. According to Interviewee L2, good user experience metrics should measure whether the user is able to accomplish their goals with using the software or not. Clicks was specifically mentioned as often being a "*terrible metric*" by Interviewee A4 and G3 due to being able to be too easily influenced by experiments without having a real impact on more relevant measures. Intuitively, the number of clicks should be kept low to have an efficient user experience. But if the program is something users enjoy spending time on (and revenue is earned through that), then the number of clicks will be beneficial to increase instead.

5.3.3 Impact of Configurability on CE

When software is built to be configured into multiple unique variants (as in software product line development (Pohl et al. 2005)) the problem of having enough user data for each feature is exasperated. In such cases, the amount of users for each such unique configuration is lower than for the total set of users, which leads to experiments taking longer to complete due to having less amounts of user data per time period. Having software be capable of configuration is viewed here as a consequence of dealing with high problem complexity. *Case A* has a product with some degree of configuration. They do experiments at different customers and analyze them independently. However, they have a limited number of customers so it is not described by the interviewees as very challenging for them to handle. *Case F* has a product with a very high degree of configuration and they are not able to conduct CE partially because of that. They have a lot of features in their product and the features exist in multiple variants and experiments would have to be repeated for the different variants, which is not efficient or might not be feasible.

5.4 Business Model Pivots at the Case Companies (P5: C6→C5)

According to **P4**, there are limits to what can be done with CE when *the problem complexity* (**C5**) is high. Companies can simplify what problem the software solves by pivoting their business model to solve another problem, one for which CE can be applied to achieve a higher *problem–solution fit* (**C3**). CE is not suitable for all development tasks (Bosch et al. 2018; Melegati et al. 2019) and is unlikely to help reduce problem complexity significantly; we see two reasons for that. First, while CE can improve *problem–solution fit* (**C3**), actually changing what *problem* the software solves is a large *change* that goes beyond the scope of CE. Changes in an experiment need to be sufficiently small such that the change can be reversed in case of bad results. Second, high *problem complexity* (**C5**) limits the ability to gain from experimentation (**P4**), which is a catch-22 scenario; CE requires a goal to target, but no goal can be specified due to the high complexity.

Case D pivoted recently by changing their target users from business customers to private individuals. After the pivot, their prioritization was data-driven to meet market needs and they had less customer requirements on their user experience, thus they were able to simplify the user experience. They also increased their CE significantly as a result of the pivot. Note that, the pivot that *Case D* conducted was *not* done specifically to enable CE. Rather, the goal was to reach a larger market which needed a simplified product, and increased *experimentation effectiveness* (**C2**) was a side-effect of simplifying the problem complexity.

No other company in the study has gone through a similar substantial pivot. Though, at *Case K*, they have experimented with licenses, such as, changing from bulk sales to a more dynamic licensing with single orders. The Product Owner (*Interviewee K1*) said this improved incentives for performing CE somewhat. *Interviewees A5, C2*, and *F1* mentioned pivoting (though not necessarily with that term) in regards to what changes they would have to implement to improve CE in their company. *Cases A* and *F* would need freemium licensing while *Case F* would need drastic changes from a consultancy based code deliveries to a product or service.

5.5 CE Incentive Structures at Case Companies (P6: C6→C4)

Some companies in the study have business models with *incentive structures* (**C6**) that enhance *experimentation effectiveness* (**C2**) and enable targeting improved *product–market fit* (**C4**). These business models provide incentive structures that motivates and facilitates CE, e.g., by having metrics available that software engineers can use to directly affect the product sales and user growth. Part of the differences in incentive structures can be explained by the *sales-led and product-led growth dichotomy*, see Section 2.1.2. Companies with a business model with product-led growth rely on the product to obtain customers, while the companies with a sales-led growth rely on a sales department to obtain customers. Incentive structures impact CE in three ways as explained in the subsections below: (1) user-business alignment, (2) sales & marketing interplay, and (3) increased user-data.

5.5.1 Impact of User-Business Alignment

According to **P3**, affecting *product–market fit* (**C4**) relies on using *live experiments* (**C2**) that target sales metrics. While the availability of sales metrics is necessary to experiment on *product–market fit* (**C4**), *incentive structures* (**C6**) are also necessary. *Case A and C* are examples that have sales metrics from their B2B customers, but cannot target *product–market fit* (**C4**) with those figures, since an increase in their customers sales figures does not come with direct immediate business value to them. As such, CE to improve the sales figures is only done for *product–market fit purposes* (**C4**) at *Case A* and not at all at *Case C*.

In contrast, *Case D and G* use various sales metrics extensively in experiments, presumably due to their user-business alignment. At both cases, the sales metrics come from software sales and they also both have a subscription-based license model where the user pays continuously to use their service. Interviewee G3 phrased it as such: *"Our incentives are mostly aligned with our customers' because it wouldn't mean anything if somebody purchases [our flagship product] and then decide that they hate it and a week later they cancel the subscription."* Interviewee D2 also expressed the importance of user-business alignment: *"So the way you build things for freemium is that it has to solve a user problem, that's part of the product breed, that's part of why you do the thing, all your metrics align to that."* Freemium is a license variant where the product has a free and a paid premium version, see Section 2.1.2.

Case D and G both also use metrics from the sales funnel to find and prioritize issues with their product. As explained by Interviewee D2: *"There's industry standard metrics, like the AARRR model Acquisition, Activation, Retention, Revenue, and Referral. We use them to analyze our platform and our product and see where we are missing things."* The AARRR Pirate Metrics Framework was proposed by McClure (2007) and covers multiple business model aspects.

5.5.2 Impact of Development and Sales & Marketing Interplay

Companies with product-led growth models will have the same *incentives* (**C6**) on the software engineering department and the sales & marketing department: to increase the sales figures. This makes it easier for them to cooperate. At *Case D* they had a few individuals with the hybrid role of growth engineer that worked as an intermediary between the two departments. At *Case G* they included growth engineers/marketers in their specialized team that conducted experiments. They were primarily tasked with finding and analyzing potential changes in the product to experiment on.

5.5.3 Impact of Increased User-Data

The case companies that have a strategy of finding many customers and growing rapidly have *incentive structures* (**C6**) that lead to gaining access to more data as the company's user base increases. User-data volume is critical for CE throughput and to affect *problem–solution fit* (**C3**) and *product–market fit* (**C4**). The interviewees at cases with sales-led growth (*Case A, C, F, K, and L*) all mentioned a focus on large important business customers that can pay more instead of many customers, which gives a higher return on the manual work that the sales force put in. Another difference lie in that the product-led growth cases (*Case D and G*) both have a freemium license model that further increases user-data, since the users that use the software for free contribute with user-data.

6 Discussion

We have presented a theory of Factors Affecting Continuous Experimentation (FACE) on what contextual factors in a company's business model and software organization affects continuous experimentation (CE) and how. The theory is based on empirical observations in 12 case companies through semi-structured interviews and subsequent theory building. In summary, the theory states that processes and infrastructure increase experiment throughput (**P1**) and experiments can increase problem–solution fit (**P2**) and live experiments can increase product–market fit (**P3**). However, CE is limited by how complex of a problem the software solves for users (**P4**), the problem complexity can be reduced by pivots in the business model (**P5**). Finally, CE on product–market fit requires business models with the right incentive structures that connect business and user value (**P6**).

We address two discussion points: (1) what the limitations of CE as a method for software engineering are and (2) which factors affect a company's ability to conduct CE.

6.1 What are the Limitations of CE?

There are *limitations to what can be achieved with CE*. According to best practice in CE (see Section 2.2), each experiment should be split into the smallest constituent to avoid having to do unnecessary implementation work in case the change is bad. As such, CE entails incremental work. In the road-map for continuous software engineering, Fitzgerald and Stol (2017) argue that sometimes abrupt changes are needed when creativity and innovation is involved, because incremental work constraints the creativity to similar solutions to what currently exists.

Furthermore, in CE, experiments are ultimately used as a method for optimizing software towards a given goal by taking small steps in the right direction. However, there is a risk of getting stuck in local optima where no individual small change can improve the current software design, but there might be a better solution to be found if a larger change is introduced. Larger changes are pivots, the impact of which in FACE are described by the proposition **P5**. When a pivot is introduced to a product that has been polished with CE, it is quite likely that the new version of the product performs worse because it is less polished. There could be minor problems that over time could be improved with CE to find a better solution. Whether or not it is worth the effort of maintaining multiple versions of the software product during this period is ultimately a business decision that cannot be settled within the confinements of CE.

The value provided by CE diminishes over time. When CE is first introduced at a company there are low-hanging fruits to experiment on. At the case companies, *Case A, D, E, G, I, and J*, there were long held assumptions about the product and customers that were finally able to be tested and this led to some surprises. However, over time CE becomes established at the company and more parts of the product are stabilized; thereby relatively reducing the gains of CE. Also, the size of experiments becomes smaller since the experimenters become more adept at reducing the scope of the experiments, which has also been reported previously by Kohavi et al. (2013). Hence the utility of each individual experiment tends to decrease unless a pivot happens. This could result in a reporting bias in favor of CE when studying companies that recently adopted CE.

Finally, FACE cannot make claims about applicability of CE to companies outside the defined scope. However, we argue that an experiment-driven approach to software development is unsuitable for much of the software outside the scope, that is, software that is not user-intensive (e.g., company intranet websites (Paulsson et al. 2022) or not user-facing (e.g., low level software libraries). Other requirements elicitation techniques or software performance optimization methods might be better suited for these examples, such as using profiling tools to pinpoint performance bottlenecks with low lead-time.

In summary, CE can increase problem–solution fit and product–market fit. However, CE should not be the only strategy for improving software at companies due to the limitations of incremental work. Pivots also play an important role in enabling the product to be optimized with CE.

6.2 What Factors are at Play in CE?

In this study, we show that CE is used for different purposes: problem–solution fit or product–market fit. Similar observations has been done before. Schermann et al. (2018) and Ros and Bjarnason (2018) describe that experiments are either regression-based to verify or validate features, or business-based to optimize. Our results highlight that most companies are only able to experiment with improving problem–solution fit and few are able to affect product–market fit. However, according to the entrepreneur Andreessen (2007): *"The only thing that matters is getting to product–market fit"*. Our main research goal in constructing the theory is to find the factors that explain differences in why companies can apply CE to different effect. That is, why some companies are able to experiment with product–market fit and others cannot.

From previous research, we know that the availability of user data is the main limitation (Kohavi et al. 2009) and that offering software-as-a-service (SaaS) (Kohavi et al. 2009) facilitates easier CE. Also, there is substantial work conducted on the challenges of applying CE in a B2B setting by Rissanen and Münch (2015); Ros and Bjarnason (2018); Yaman et al. (2016) and in cyber-physical systems by Bosch and Eklund (2012); Giaimo (2016); Mattos et al. (2018). FACE synthesizes these findings; in fact most of these studies are all on aspects of complexity in the problem the product solves for users; corresponding to **P4**. FACE highlights additional factors as follows.

Three of the propositions (**P1**, **P4**, and **P6**) affect companies' ability to use CE to increase problem–solution fit or product–market fit and those form the factors. The first (**P1**), on CE processes and infrastructure, covers many aspects and not all are regarded by us as very impactful on whether a company can apply CE or not. While the throughput would be lower without sufficient processes and infrastructure, it is unlikely to be a complete blocker to CE. In addition, several of the companies (see Section 5.1) have described how they gradually

improved their CE support in parallel with ramping up CE. Data infrastructure is an exception that requires significant investment. The final three *derived factors* correspond to constructs **C1**, **C5**, and **C6**, and are:

1. *Data infrastructure* is needed to be able to use the metrics that are desired and to cover telemetry of all parts of the software. Companies that rely on user data for other parts of their products, such as recommendation systems, will get a head start on CE by reusing data infrastructure.
2. *User problem complexity* is the complexity of the problem that the software solves for users. High problem complexity makes quantifying the user experience hard and making changes in the software hard, which severely impacts the ability to experiment. Modifying the user problem is challenging but possible with pivots in the business model.
3. *Incentive structures* are required to provide a measurable link between business value, user value, and software engineering activities, such that experiments can affect product-market fit. Companies that do not have access to sales metrics in experiments will derive less benefits from experiments since user experience metrics are hard to define in a way that they can be used for optimization.

The final factor, incentive structures, seems to be particularly impactful. For example, *Cases D and G* have freemium product licenses and thereby rely on the product to convince users to convert to paying customers. CE plays a key role to help develop their offering to be able to acquire more users. Incentive structures is not explicitly pointed out as a factor influencing CE in previous work (see 2.2.2), though the inverse is mentioned, such that incentives are lacking for B2B companies for conducting experiments (Rissanen and Münch 2015; Yaman et al. 2017).

7 Conclusions

We conducted a multi-case study with 12 companies and built a theory called Factors Affecting Continuous Experimentation (FACE), based on the empirical material for understanding what factors are at play for companies conducting continuous CE. Six propositions are included in FACE. (1) *Efficient infrastructure and process improves experimentation effectiveness*. Starting with experiments is easy, but companies can put endless effort into: scaling up experiments, obtaining more insights from experiments, and improving the speed of experiments. (2) *Experiments can affect either the problem–solution fit* or (3) *product–market fit*. Targeting product–market fit is far more challenging, but preferable, since it can improve business and user value simultaneously. Many companies are restricted to only improve the user experience and thereby problem–solution fit. (4) *The complexity of the problem the software solves for users strongly limits experiment applicability*. High complexity can limit the ability to make desired changes in the software or to quantify the user sessions. Following that, (5) *pivots in the business model is necessary to simplify the problem complexity*, experiments on their own are unlikely to succeed. Finally, (6) *improving product–market fit needs incentive structures* in the form of metrics from the sales process.

FACE can be used to evaluate company contexts to gauge CE applicability at their companies. There is still future work to further validate the theory by using it in this way to evaluate companies. Additionally, we plan to derive guidelines from the theory to further guide practitioners in their CE.

Appendix A: Case Companies

Case A: E-Commerce Algorithms

This case company offers an e-commerce platform that is sold to companies (B2B). The platform consists of various algorithms for ranking products and an administration interface. While the algorithms target end-consumers, the administration interface targets managers at the e-commerce companies. The algorithms provide value for the end-consumers by increasing the relevance of the product that they see on the web shops. The case company is fairly small with about 50 employees and was established 20 years ago. The company's business model is to sell usage licenses to other companies and the company has salespersons working with direct sales as their only source of revenue. Company A has experienced several periods of growth after which the company has had to scale down due to failure of a big sale. The company conducts a medium amount of CE but only on the software that targets their end-user consumers, not on the administrative interface. The company only performs quantitative experiments since the software that is experimented on does not have a graphical interface. Company A also assist their business customers with their CE.

Case B: Local Search Service

Case B offers a search engine service for search within a local region, a.k.a., yellow pages. The search engine is free to use and the major source of revenue is companies buying promotion in the search result rankings and visual presentation. The case company has a large sales team that work with direct sales by calling potential business customers. The company is about 20 years old and has stayed stable for some years at about 50 employees. The company was recently acquired by a larger business group and additional products were integrated with the local search service, such as a ticket booking service. CE is not frequently applied at the case company and is only used to verify large changes to the search engine ranking algorithm with quantitative data.

Case C: E-Commerce Consultants

Case company C is a consultancy firm that develops and maintains web shops for other companies with relatively large demands on traffic volume and/or product catalogue. The work includes a lot of integrating various software systems, e.g., product information management (PIM), content management systems (CMS), payment gateway systems, search or recommendation engines, etc. These software systems each come with their own administrative tools and the case company helps their customers to use these tools. Company C has built its own software to optimize and support the process of building the web shops, but the business model is to sell consulting hours in a per project basis. The company has existed for more than 20 years, currently has about 100 employees and is experiencing steady growth in the number of employees. The company does not conduct any CE on its own software but has assisted its customers in conducting quantitative experiments on their web shops.

Case D: Video Sharing

Case company D develops and sells a video sharing platform where users can record and edit videos for marketing purposes. The company has been operating for 10 years, and has

about 200 employees of which about 30 people are in the software development department, distributed over four teams. Recently, the company pivoted its business model by adding a new product that is targeted at individuals; aimed at consumers and smaller companies that wish to market themselves. This new product is the defining boundary of case D. Prior to the pivot, the customers have mainly been other businesses (B2B) that Company D has reached through a fairly big sales department with direct sales. The new product has an entirely separate development department and no sales persons are involved. The pivot enabled the use of CE at the company. The new B2C product is offered under a freemium license, where the free version is offered with limited video uploading capacity, and the paid version offers additional features. The B2B part of the company performs no CE. In contrast, the B2C team conducts extensive CE and has various specialized roles for supporting CE and related activities, viz., data scientists and data engineers for quantitative experiments and user researchers for qualitative experiments.

Case E: Web Shop

This case company offers a web shop with a subscription service to physical goods. The web shop drives sales through their online presence only and does not have any retail stores. The company was founded less than 10 years ago and has experienced rapid growth in revenue and number of employees. At the time of the interview, the company has around 3 000 employees, most of whom work with delivering the physical goods. The company's IT department accounted for about 250 employees. In addition to the web shop, the IT department also operates in-house software systems for handling logistics, marketing, etc. The interviewee at the company was involved with optimizing multiple of these products at the company. There is a drive from the management to be data-driven and CE is encouraged on a strategic level. However, this is hindered in practice by chaotic management, caused by the rapid expansion.

Case F: Customer Relations Product

Case F sells a customer relations product (and service) to other businesses. The product is highly customizable and each new customer gives rise to a new integration project. The integration includes adapting to another company's data model and internal software systems, e.g., for human resources. The company is 30 years old and has 200 employees, 35 of which are in software development, divided into three teams. Sales is a large part of the company, and all sales are done through direct sales. Half of the company is committed to pre-sales and operations that support customers before and after sales due to the complex integration. No free version or trial of the product is available. Company F has conducted a few prototyping qualitative experiments on recent new features, but not on all developments. The company is aware of the concept of quantitative experiments but has not conducted any and have no concrete plans for applying CE in the near future. However, the company has some of the technological pre-requisites to conduct post-deployment CE in place already, and make use of feature flags to verify new developments at specific customers, but not in a systematic way.

Case G: Software Engineering Tools

Case G is a huge international company with multiple products that focus on supporting the software engineering process. The main products are a project tracker, issue tracker, and a

team collaboration platform. The company has existed for roughly two decades and today have more than 4 000 employees. The majority of these employees are within IT. Each of the company's products has its own development organization. The company advocates agile software development. Notably the company has no large sales department and do not use direct sales at all, despite their focus on business to business (B2B). The company conducts huge amounts of experiments on all aspects of their products—both on new and old products, and makes use of both quantitative and qualitative experiments. A specialised team supports product teams in applying a CE approach. Since there are multiple teams involved with CE on multiple products, the company has an extensive CE platform. The company also has a formal process for conducting CE developed by the CE team.

Case H: Web Shop

This case company offers a web shop for business customers only. The company has over 2 000 employees, of which most are employed at retail stores. The company used case company C to develop their web shop and also has their own small IT department that manages and improves the web shop. The company has conducted ad hoc experiments on occasion and uses off-the-shelf CE tools (Google Analytics) for supporting their CE. The biggest limiting factor of the Company H's CE practices is the lack of human resources at the IT department.

Case I: Web Shop

The next case is a huge international conglomerate. The company has a long and rich history in retail and have operated a web shop for over 20 years. The web shop is becoming increasingly important to the company. The company has about 200 000 employees of which about 5 000 are employed in the IT organization. The company have many products for managing their logistics, etc., but in this study we focus on their web shop product only. Several cross-functional development teams are responsible for various parts of the web shop (such as the recommender engine). Not all parts of the web shop are experimented on, but the trend is moving towards more CE work. There are multiple teams involved with CE at the company, some of them have their specialized CE infrastructure but most teams use a centralized experimentation platform and contact a data science team for help with CE.

Case J: Product Information Platform

Case J offers a service for product information within the building industry that is free to use. The company was established less than 10 years ago. The source of revenue come from customers that want to know who accessed their product's information in order to obtain sales leads. The case company describes their business model as being a middle man that sells information, and has a sales department that works with direct sales. The company also relies on organic growth based on their free users. There are about 200 employees at the case company, half of which work within the IT organization. Company J does not make use any of qualitative methods and quantitative experiments are new to the company. Currently, the company does not experiment on all parts of their software. There is a small and dedicated team with two employees that conducts experiments, and the company expects that the other software engineering teams will soon start applying CE.

Case K: Business Intelligence

This company offers several advanced business intelligence products for different needs. The products are used to make graphs, tables, and other visualizations from various data sources. The products are advanced to use and the flagship product even has a proprietary domain-specific programming language for data manipulation. Since the company is more than 20 years old, the products are in different stages of the life cycle, albeit all of them are still offered. The company is a large enterprise with about 3 000 employees and 500 of them in the development department. The company has grown rapidly during the latest years with an increase in the number of employees. The sales department is very large and uses primarily direct sales to other businesses. The company has a team specialized in user experience research that gathers qualitative feedback on a regular basis. The company conducts qualitative experiments to evaluate both prototypes and completed functionality with users. However, not all development teams are on board with this yet so not all features are evaluated in this way. The user experience research team is aware of and interested in quantitative experiments but getting such systems in place has not been a company priority.

Case L: Employee Management Product

The final case company develops a product for employee management. The product is intended to be embedded in the customers' intranet and has both administrative and end-users within each customer organization. The company was founded 20 years ago and has about 250 employees of which about a third in the IT department. The company relies on direct sales with a sales force and most of the revenue comes from projects that integrate their product at customer sites. The company pays close attention to user experience and has a team involved with user research that conducts qualitative experiments with prototypes. Since the product is still considered to be early in development this CE is somewhat frequent.

Appendix B: Interview Guide

The following questions should be adapted to suit the interviewees background and role. The nested bullet lists indicate probes that are only asked when the answer to the main question requires clarification.

B.1 Introduction

- Inform about consent
  - Interview will be recorded
  - Free to withdraw at any time (including afterwards)
  - Data will be treated confidentially and anonymized
- Explain purpose of study
  - Investigating context and process of experimentation and data use

B.2 Case Context

1. What does your company do?
2. What is the overall business strategy?
3. What is the business model?
   (a) What user problem does your offering solve?
   (b) Solution: Product? Service? Consultancy? Other?
   (c) Who are the customers? B2B? B2C? B2X?
   (d) What are the key metrics and how is it measured (channels)?
   (e) Costs and revenue?
4. How stable is the business model?
   (a) How do you know when to pivot (change direction)?
5. How many employees are there in: Total? Dev? Sales & marketing? Ops?
6. Could you describe your own role(s)?
   (a) What is your background?
7. What does your team do within the company?
   (a) Is it cross-functional?
8. In what ways does your company use user data?
   (a) Is there a specialized data science or engineering team?
   (b) Could you shortly describe your overall infrastructure for data?
9. How does your SE team prioritize what to build?
10. How does user data help with prioritization?

B.3 Experimentation Process

1. How was experimentation introduced at the company?
   (a) In what department was it started?
   (b) In what departments is it done now?
2. Why do you do experiments?
   (a) Knowledge? Prioritization? Optimization? Regression? Validation?
3. Could you break down the steps that are taken by you and your team when conducting an experiment?
   (a) Duration? Roles? MVF? Analysis? Power?
   (b) Are you aware of any missing steps?
4. What do you experiment on?
5. To what extent do you experiment?
   (a) Duration? Frequency? Coverage?
6. Is there overlapping experimentation?
   (a) From different teams?

(b) Coordinated?

B.4 Experimentation Details

1. Could you describe your infrastructure for experimentation?
   (a) CI/CD pipeline? Reporting? Reliability? Scalability?
   (b) Do you use blue/green deployment or feature flags?
   (c) What would you like to improve in your infrastructure?
2. What type of experiment designs do you use?
   (a) A/A experiments? MVTs? Bandit testing?
   (b) Do you have decision algorithms taking causal decisions?
3. Do you use any qualitative methods?
   (a) Focus groups? User studies?
   (b) Do you have specialized teams or individuals for it?
   (c) How does it interplay with quantitative experimentation?
4. What types of metrics do you use?
   (a) What metrics would you want to use?
   (b) How does your metric translate to company or team success?
5. Are experiments analyzed or executed in different segments?
   (a) Explicit segments: Verticals? Products? B2B customers? Web pages?
   (b) Implicit through data mining?
   (c) How do you handle diverging results?
6. Do you ever do experiments involving external code bases?
   (a) How did it affect experimentation?
7. How do you share knowledge from experiments?
   (a) Mail? Meetings? Documentation?
8. Do you do any long term follow ups on experiments?
   (a) Repetitions? Long-running experiments?

B.5 Holistic Experimentation View

1. What are the main challenges with experimentation?
2. What are the main benefits with experimentation?
3. Do you face any ethical dilemmas involving your use of user data or experimentation?
4. How do you strive to improve your experimentation?

B.6 Final Remarks

1. Do you have any final comments, anything that should have been asked?
2. Could you recommend us any additional interviewee (or organization?)?

3. We will get back to you within 1 to 2 weeks about a summary of what was said here.

Appendix C: Code Book

All paragraphs of the interview text were coded. Multiple codes were used on the same paragraph. Some codes are marked with an X to indicate that they can vary, e.g., *Scenario X* is as *Scenario optimization* or *Scenario validation*, these were expanded upon during coding. The division of detail codes into sections are intended only for improved readability, they are not themes or categories. The first two sections indicate context around experimentation, and the last two are on experimentation process and evidence of actual use.

C.1 Software Development and Infrastructure

- Testing (*any testing aspects discussed*)
- Prioritization (*any prioritization aspects discussed*)
- System architecture X (*description of type of architecture e.g. embedded or micro services*)
- Data infrastructure (*description of e.g. data warehouses, query engines, data science teams, data engineering teams*)
- Data availability (*whether data is capable to be used for experiments*)
- Data governance (*cleaning, data provenance, data*)
- Experimentation platform (*description of an experimentation platform in use at company*)
- Infrastructure improvements
- Infrastructure challenges
- Organizational structure (*How departments are structured and how much of e.g. sales and development there is at a company with relevance to experimentation*)
- Company culture (*descriptions of culture that influences experimentation*)
- Knowledge sharing (*how knowledge is shared between departments*)
- Technical infrastructure
- Software stack technology
- CICD pipeline
- Infrastructure maturity

C.2 Business Model and Strategy

- Pivoting (*change in business model according to strategy*)
- Product customization (*general or tailored to different market segments*)
- Product complexity (*description of a complex product in e.g. size or user experience*)
- Problem-solution pair (*what problem the product solves for users*)
- Key metrics
- Market constraints (*ethics, legislation*)
- Cost structure
- Revenue stream (*pricing model*)
- Growth model (*how new customers are acquired, channels, customer segments, number users, etc.*)
- Channels (*path to customers*)
- Unique value proposition

– Target customers
– Unfair advantage

C.3 Experimentation Process

These codes describe strategies for various stages in the process and why experimentation is conducted.

– Ideation (*how ideas/hypotheses are elicited and prototyping is performed at the company*)
– Experiment design (*how design are decided, pros and cons of designs*)
– Metrics (*which are used, what specific metrics mean, how they are derived*)
– Analysis (*how analysis is conducted*)
– Scenario X (*different experiment archetypes e.g. optimization, validation, verification, learning*)
– Dark patterns (*a dark pattern is an unethical anti-pattern in UX for tricking users*)
– Experimenter X (*used to indicate what role initiates and owns an experiment*)
– Role X (*used to indicate experimentation involvement*)
– Experiment handover (*description of how experimentation is conducted by some specialist and then hand over to product team after completion*)
– Experiment inhibitor (*something hinders experimentation, use in combination with another context giving code*)
– Experiment enabler (*something enables experimentation, use in combination with another context giving code*)
– Experimentation frequency

C.4 Experimentation Usage

These codes are used to gauge how much experimentation is done at a particular company, the first three should be mutually exclusive at a company unless there is some conflicting reports by different interviewees.

– Experimentation awareness (*awareness of experimentation at a company but no intent to start*)
– Experimentation intent (*aware of experimentation at a company and want to start experimentation but have not done so yet*)
– Experimentation adoption (*in the process of adopting experimentation and/or increasing scale of exp*)
– Experiment duration
– Experiment goal

These codes are used when a technique is mentioned as being used at the case.

– Sprint experiment (*experiment is conducted as part of ordinary development and is prioritized with other software development*)
– Stand-alone experiment (*experiment that is conducted outside of the development organization*)
– Controlled experiment usage (*A/B test, quasi experiment*)
– Optimization usage (*multi-armed bandits, A/Bn tests, MVT, simulations*)
– Qualitative methods usage (*focus groups, observations*)
– Survey usage (*questionnaires*)

- Data mining usage
- Feature flag experiments (*considerations for running experiments through feature flags in the same deployment environment*)
- Blue-green deployment experiments (*considerations for running experiments through parallel deployments*)
- Repeating experiments (*same hypothesis repeated for some reason e.g. disbelief, bugs, etc.*)
- Experimentation cycles (*describes an actual cycle of experimentation where one experiment lead to a new hypotheses and so on*)
- Overlapping experiments (*considerations for running many experiments in parallel*)

Acknowledgements We want to thank all the participating anonymous companies and interviewees for their contribution to this project. Thanks to Klaas-Jan Stol, Helena Holmström Olsson, Lars Bengtsson, and Fabian Fagerholm for their feedback on an earlier manuscript version in the first author's PhD thesis. We also thank the anonymous reviewers for their constructive feedback to help improve the clarity of the presentation

Funding Open access funding provided by Lund University.

Data Availability The protocols for the case study including interview guide and code book are available in Appendix B and C, respectively. Data collected during the interviews are not publicly available due to reasons of confidentiality

Declarations

Conflicts of interest This work was funded by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP), which in turn is funded by Knut and Alice Wallenberg Foundation. The authors have no other financial or non-financial interests, or connection to the involved case companies, to disclose

Informed consent All interviewees participated voluntarily. At the beginning of each interview, the interviewees were informed about their right to withdraw at any time and how the study data was treated with respect to confidentiality and anonymity, as defined in the interview guide in Appendix B. The consents given by the interviewees were audio recorded

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Andreessen M (2007) Part 4: The only thing that matters. https://pmarchive.com/guide_to_startups_part4.html. Accessed 10 Dec 2021

Auer F, Felderer M (2018) Current state of research on continuous experimentation: A systematic mapping study. In: Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA, pp. 335–344. https://doi.org/10.1109/SEAA.2018.00062

Auer F, Ros R, Kaltenbrunner L, Runeson P, Felderer M (2021) Controlled experimentation in continuous experimentation: Knowledge and challenges. Inf Softw Technol 134:106551. https://doi.org/10.1016/j.infsof.2021.106551

Bartlett B (2020) What is product led growth? how to build a software company in the end user era. Accessed: 15-Jan-2021. https://openviewpartners.com/blog/what-is-product-led-growth

Bjarnason E (2021) Prototyping practices in software startups: Initial case study results. In: Proceedings of the 29th International Requirements Engineering Conference Workshops, REW, pp. 206–211. https://doi.org/10.1109/REW53955.2021.00038

Bosch J (2012) Building products as innovation experiment systems. In: Proceedings of the 3rd International Conference on Software Business, ICSOB, pp. 27–39. https://doi.org/10.1007/978-3-642-30746-1_3

Bosch J, Eklund U (2012) Eternal embedded software: Towards innovation experiment systems. In: Prodeedings of the 5th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change, ISoLA, pp. 19–31. https://doi.org/10.1007/978-3-642-34026-0_3

Bosch J, Olsson HH, Björk J, Ljungblad J (2013) The early stage software startup development model: a framework for operationalizing lean principles in software startups. In: Fitzgerald B, Conboy K, Power K, Valerdi R, Morgan L, Stol KJ (eds.) Lean Enterprise Software and Systems, pp. 1–15. Springer Publishing Company. https://doi.org/10.1007/978-3-642-44930-7_1

Bosch J, Olsson HH, Crnkovic I (2018) It takes three to tango: Requirement, outcome/data, and AI driven development. In: Proceedings of the 1st International Workshop on Software-intensive Business: Startups, Ecosystems and Platforms, SiBW, pp. 177–192

Brodén B, Hammar M, Nilsson BJ, Paraschakis D (2019) A bandit-based ensemble framework for exploration/exploitation of diverse recommendation components: An experimental study within e-commerce. ACM Transactions on Interactive Intelligent Systems (TiiS) 10(1):1–32. https://doi.org/10.1145/3237187

Brooks FP (1986) No silver bullet–essence and accident in software engineering. In: Proceedings of the IFIP Tenth World Computing Conference, p. 1069–1076

Brunswicker S, Wrigley C, Bucolo S (2013) Business model experimentation: What is the role of design-led prototyping in developing novel business models? In: Curley M, Formica P (eds.) The experimental nature of new venture creation: capitalizing on Open Innovation 2.0, pp. 139–151. Springer Publishing Company. https://doi.org/10.1007/978-3-319-00179-1_13

Chaparro XAF, de Vasconcelos Gomes LA (2021) Pivot decisions in startups: a systematic literature review. International Journal of Entrepreneurial Behavior & Research **27**(4). https://doi.org/10.1108/IJEBR-12-2019-0699

Chesbrough H (2007) Business model innovation: it's not just about technology anymore. Strat Leadersh 35(6):12–17. https://doi.org/10.1108/10878570710833714

Chesbrough H, Rosenbloom RS (2002) The role of the business model in capturing value from innovation: evidence from Xerox Corporation's technology spin-off companies. Ind Corp Chang 11(3):529–555. https://doi.org/10.1093/icc/11.3.529

Clarke P, O'Connor RV (2012) The situational factors that affect the software development process: Towards a comprehensive reference framework. Inf Softw Technol 54(5):433–447. https://doi.org/10.1016/j.infsof.2011.12.003

Ebert C, Brinkkemper S (2014) Software product management-an industry evaluation. J Syst Softw 95:10–18. https://doi.org/10.1016/j.jss.2013.12.042

Fagerholm F, Guinea AS, Mäenpää H, Münch J (2014) Building blocks for continuous experimentation. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, pp. 26–35. https://doi.org/10.1145/2593812.2593816

Fagerholm F, Guinea AS, Mäenpää H, Münch J (2017) The RIGHT model for continuous experimentation. J Syst Softw 123:292–305. https://doi.org/10.1016/j.jss.2016.03.034

Feitelson DG, Frachtenberg E, Beck KL (2013) Development and deployment at Facebook. IEEE Internet Computing 17(4):8–17. https://doi.org/10.1109/mic.2013.25

Fitzgerald B, Stol KJ (2017) Continuous software engineering: A roadmap and agenda. J Syst Softw 123:176–189. https://doi.org/10.1016/j.jss.2015.06.063

Giaimo F, Yin H, Berger C, Crnkovic I (2016) Continuous experimentation on cyber-physical systems: Challenges and opportunities. In: Proceedings of the Scientific Workshops of XP, pp. 1–2. https://doi.org/10.1145/2962695.2962709

Glaser B, Strauss A (1967) The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine-Transaction

Gupta S, Ulanova L, Bhardwaj S, Dmitriev P, Raff P, Fabijan A (2018) The anatomy of a large-scale experimentation platform. In: Proceedings of the 15th International Conference on Software Architecture, ICSA, pp. 1–109. https://doi.org/10.1109/icsa.2018.00009

Gutbrod M, Münch J, Tichy M (2017) How do software startups approach experimentation? empirical results from a qualitative interview study. In: Proceedings of the 18th International Conference on Product-

Focused Software Process Improvement, PROFES, pp. 297 –304. https://doi.org/10.1007/978-3-319-69926-4_21
Holson LM (2009) Putting a bolder face on Google. New York Times **1**. https://www.nytimes.com/2009/03/01/business/01marissa.html
Johnson G, Scholes K, Whittington R (2008) Exploring corporate strategy: Text and cases. Pearson Education
Kemell KK, Feshchenko P, Himmanen J, Hossain A, Jameel F, Puca RL, Vitikainen T, Kultanen J, Risku J, Impiö J, et al (2019) Software startup education: gamifying growth hacking. In: Proceedings of the 2nd International Workshop on Software-Intensive Business: Start-ups, Platforms, and Ecosystems, IWSiB, pp. 25–30. https://doi.org/10.1145/3340481.3342734
Kevic K, Murphy B, Williams L, Beckmann J (2017) Characterizing experimentation in continuous deployment: a case study on Bing. In: Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track, ICSE-SEIP, pp. 123–132. https://doi.org/10.1109/ICSE-SEIP.2017.19
Kirtley J, O'Mahony S (2020) What is a pivot? explaining when and how entrepreneurial firms decide to make strategic change and pivot. Strat Manag J Special Issue. https://doi.org/10.1002/smj.3131
Kohavi R, Crook T, Longbotham R, Frasca B, Henne R, Ferres JL, Melamed T (2009) Online experimentation at Microsoft. Proceedings of the 3rd International Workshop on Data Mining Case Studies **11**
Kohavi R, Deng A, Frasca B, Walker T, Xu Y, Pohlmann N (2013) Online controlled experiments at large scale. In: Proceedings of the 19th International Conference on Knowledge Discovery and Data Mining, KDD, pp. 1168–1176. https://doi.org/10.1145/2487575.2488217
Kohavi R, Longbotham R, Sommerfield D, Henne RM (2009) Controlled experiments on the web: Survey and practical guide. Data Min Knowl Disc 18(1):140–181. https://doi.org/10.1007/s10618-008-0114-1
Krafcik JF (1988) Triumph of the lean production system. Sloan Manag Rev 30(1):41–52
Mattos DI, Bosch J, Olsson HH (2018) Challenges and strategies for undertaking continuous experimentation to embedded systems: Industry and research perspectives. In: Proceedings of the 19th International Conference on Agile Processes in Software Engineering and Extreme Programming, XP, pp. 277–292. https://doi.org/10.1007/978-3-319-91602-6_20
Maurya A (2012) Running lean: iterate from plan A to a plan that works. O'Reilly Media
McClure D (2007) Startup metrics for pirates: AARRR! https://500hats.typepad.com/500blogs/2007/09/startup-metrics.html. Accessed 15 Jan 2022
Melegati J, Edison H, Wang X (2022) Xpro: A model to explain the limited adoption and implementation of experimentation in software startups. IEEE Trans Softw Eng 48(6):1929–1946. https://doi.org/10.1109/TSE.2020.3042610
Melegati J, Wang X, Abrahamsson P (2019) Hypotheses engineering: first essential steps of experiment-driven software development. In: Proceedings of the Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution, RCoSE/DDrEE, pp. 16–19. https://doi.org/10.1109/RCoSE/DDrEE.2019.00011
Munir H, Runeson P, Wnuk K (2018) A theory of openness for software engineering tools in software organizations. Inf Softw Technol 97:26–45. https://doi.org/10.1016/j.infsof.2017.12.008
Niculescu MF, Wu DJ (2014) Economics of free under perpetual licensing: Implications for the software industry. Inf Syst Res 25(1):173–199. https://doi.org/10.2139/ssrn.1853603
Olsen D (2015) The lean product playbook: How to innovate with minimum viable products and rapid customer feedback. John Wiley & Sons
Olsson HH, Bosch J (2014) The HYPEX model: from opinions to data-driven software development. In: Bosch J (ed.) Continuous Software Engineering, pp. 155–164. Springer Publishing Company. https://doi.org/10.1007/978-3-319-11283-1_13
Olsson HH, Bosch J (2019) Data driven development: Challenges in online, embedded and on-premise software. In: Proceedings of the 20th International Conference on Product-Focused Software Process Improvement, PROFES, pp. 515–527. https://doi.org/10.1007/978-3-030-35333-9_36
Osterwalder A (2004) The business model ontology a proposition in a design science approach. Ph.D. thesis, Faculty of Business and Economics of the University of Lausanne
Osterwalder A, Pigneur Y (2010) Business model generation: A handbook for visionaries, game changers, and challengers, vol. 1. John Wiley & Sons
Paulsson A, Runeson P, Ros R (2022) A/B testing in the small. In: Taibi D, Kuhrmann M, Abrahamsson P (eds.) Product-Focused Software Process Improvement, PROFES, LNCS 13709, pp. 449–463. Springer International Publishing. https://doi.org/10.1007/978-3-031-21388-5_31
Petersen K, Wohlin C (2009) Context in industrial software engineering research. In: Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM, pp. 401–404. https://doi.org/10.1109/ESEM.2009.5316010

Pohl K, Böckle G, van Der Linden F (2005) Software product line engineering: foundations, principles, and techniques. Springer Publishing Company

Rajala R, Rossi M, Tuunainen VK (2003) A framework for analyzing software business models. In: Proceedings of the 11th European Conference on Information Systems, ECIS, pp. 1614–1627

Ries E (2011) The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Crown Business

Rissanen O, Münch J (2015) Continuous experimentation in the B2B domain: a case study. In: Proceedings of the 2nd International Workshop on Rapid Continuous Software Engineering, RCoSE, pp. 12–18. https://doi.org/10.1109/RCoSE.2015.10

Rodríguez P, Urquhart C, Mendes E (2022) A theory of value for value-based feature selection in software engineering. IEEE Trans Softw Eng 48(2):466–484. https://doi.org/10.1109/TSE.2020.2989666

Ros R (2020) Continuous experimentation with product-led business models: A comparative case study. In: Proceedings of the 11th International Conference on Software Business, ICSOB, pp. 143–158. https://doi.org/10.1007/978-3-030-67292-8_11

Ros R, Bjarnason E (2018) Continuous experimentation scenarios: A case study in e-commerce. In: Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA, pp. 353–356. https://doi.org/10.1109/seaa.2018.00064

Ros R, Hammar M (2020) Data-driven software design with constraint oriented multi-variate bandit optimization (COMBO). Empir Softw Eng 25(5):3841–3872. https://doi.org/10.1007/s10664-020-09856-1

Ros R, 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, pp. 35–41. https://doi.org/10.1145/3194760.3194766

Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empir Softw Eng 14(2):131–164. https://doi.org/10.1007/s10664-008-9102-8

Runeson P, Höst M, Rainer A, Regnell B (2012) Case study research in software engineering: Guidelines and examples. John Wiley & Sons

Sauro J, Lewis JR (2016) Quantifying the user experience: Practical statistics for user research. Morgan Kaufmann Publishers. https://doi.org/10.1016/C2010-0-65192-3

Schermann G, Cito J, Leitner P, Zdun U, Gall HC (2018) We're doing it live: A multi-method empirical study on continuous experimentation. Inf Softw Technol 99:41–57. https://doi.org/10.1016/j.infsof.2018.02.010

Schermann G, Leitner P (2018) Search-based scheduling of experiments in continuous deployment. In: Proceedings of the 34th International Conference on Software Maintenance and Evolution, ICSME, pp. 485–495. https://doi.org/10.1109/icsme.2018.00059

Schief M, Buxmann P (2012) Business models in the software industry. In: Proceedings of the 45th Hawaii International Conference on System Sciences, HICSS, pp. 3328–3337. https://doi.org/10.1109/HICSS.2012.140

Schumacher R (2009) The handbook of global user research. Morgan Kaufmann Publishers

Sjøberg DIK, Dyb å T, Anda BCD, Hannay JE (2008) Building theories in software engineering. In: Shull F, Singer J, Sjøberg DIK (eds.) Guide to Advanced Empirical Software Engineering, pp. 312–336. Springer Publishing Company. https://doi.org/10.1007/978-1-84800-044-5_12

Sorescu A (2017) Data-driven business model innovation. J Prod Innov Manag 34(5):691–696. https://doi.org/10.1111/jpim.12398

Stol KJ, Fitzgerald B (2015) Theory-oriented software engineering. Sci Comput. Program 101:79–98. https://doi.org/10.1016/j.scico.2014.11.010

Stol KJ, Ralph P, Fitzgerald B (2016) Grounded theory in software engineering research: a critical review and guidelines. In: Proceedings of the 38th International Conference on Software Engineering, pp. 120–131. https://doi.org/10.1145/2884781.2884833

Tang D, Agarwal A, O'Brien D, Meyer M (2010) Overlapping experiment infrastructure: More, better, faster experimentation. In: Proceedings of the 16th International Conference on Knowledge Discovery and Data Mining, KDD, pp. 17–26. https://doi.org/10.1145/1835804.1835810

Troisi O, Maione G, Grimaldi M, Loia F (2020) Growth hacking: Insights on data-driven decision-making from three firms. Industrial Marketing Management: the international journal for industrial and high-tech firms. 90:538–557. https://doi.org/10.1016/j.indmarman.2019.08.005

Vanhala E, Smolander K (2013) What do we know about business models in software companies?: systematic mapping study. IADIS Intern J on WWW/Intern 11(3):89–102

Vargas BP, Signoretti I, Zorzetti M, Marczak S, Bastos R (2020) On the understanding of experimentation usage in light of lean startup in software development context. In: Proceedings of the 24th International

Conference on Evaluation and Assessment in Software Engineering, EASE, pp. 330–335. https://doi.org/10.1145/3383219.3383257

Wieringa R, Daneva M (2015) Six strategies for generalizing software engineering theories. Sci Comput Program 101:136–152. https://doi.org/10.1016/j.scico.2014.11.013

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer Publishing Company

Wrigley C, Straker K (2016) Designing innovative business models with a framework that promotes experimentation. Strategy & Leadership **44**(1). https://doi.org/10.1108/SL-06-2015-0048

Yaman SG, 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: Proceedings of the 17th International Conference on Product-Focused Software Process Improvement, PROFES, pp. 344–359. https://doi.org/10.1007/978-3-319-49094-6_22

Yaman SG, Munezero M, Mü nch J, Fagerholm F, Syd O, Aaltola M, Palmu C, Männistö T, (2017) Introducing continuous experimentation in large software-intensive product and service organisations. J Syst Softw 133:195–21. https://doi.org/10.1016/j.jss.2017.07.009

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.