The CRUSOE Framework: A Holistic Approach to Analysing Prerequisites for Continuous Software Engineering

Teemu Karvonen[1,✉], Tanja Suomalainen[2], Marko Juntunen[3], Tanja Sauvola[1], Pasi Kuvaja[1] and Markku Oivo[1]

[1] University of Oulu, Information Technology and Electrical Engineering
[3] University of Oulu, Oulu Business School
{teemu.3.karvonen, marko.juntunen, tanja.sauvola, pasi.kuvaja, markku.oivo}@oulu.fi
[2] VTT Technical Research Centre of Finland, Ltd
tanja.suomalainen@vtt.fi

Abstract. Continuous software engineering (CSE) is used for customer experiments and repetitive integrated processes within and between business planning and software development. First, this paper defines a new framework, called CRUSOE, for analysing CSE prerequisites. The framework allows for a more precise analysis of the interrelations and estimation of the changes that are prerequisites for moving from traditional product development to CSE. CRUSOE addresses prerequisites associated with and interdependencies among (1) the strategy, (2) architecture and (3) organisation. Second, this paper describes a case study conducted as part of a smartphone platform project to investigate the CSE prerequisites for product-focused software development. The results are synthesised together with recent related studies using the CRUSOE framework. The findings confirm challenges in moving towards CSE in embedded system development. Moreover, context-specific prerequisites should be considered, while it is still unclear as to how CSE can be systematically applied to the non-website development context.

Keywords: Continuous software engineering, strategy, architecture, organising, BizDev, software ecosystem

1 Introduction

Embedded and product-intensive software development project teams are becoming increasingly interested in applying practices and tools for continuous software engineering (CSE) [1]; e.g., the Lean Startup method [2], DevOps [3], continuous delivery (CD) [4] and continuous experimentation [5]. Although many of these practices are widely acknowledged in the field of website development [6,7], there are only a few frameworks that describe how CSE can be applied in product-focused embedded sys-

tem development (e.g. smartphones, cars etc.). Moreover, there is still very little empirical evidence of the actual usage of these practices in this context. The existing studies have mostly indicated severe challenges in adopting these practices in business-to-business (B2B) and embedded system development [8,9,10,11] contexts. In addition, CD and continuous experimentation still seem to mostly be used for small-scale website development projects [6], [8], [12]. Fagerholm et al. [5] have recently investigated continuous experimentation in university software laboratory projects with two case companies and have introduced a model for explaining how the continuous experimentation can be organised. However, more empirical studies are needed to increase our understanding of how these practices could be implemented in different software development contexts. Consequently, in this paper, our goal is to clarify the key prerequisites for applying CSE in product-focused software development.

The sustainable success of a company can be linked to its capabilities in terms of bringing new innovations to market. In today's competitive and turbulent business environment, time to market has also become very important. Consequently, business stakeholders have identified rapid fielding and continuous experimentation as important elements of their long-term strategies. Development stakeholders are tasked with finding a balance between development speed and stability, as the development process speed can often be temporarily increased by collecting *technical debt* (e.g., skipping some steps in the process), followed by a slowdown in development due to having to pay off the debt later. Consequently, companies need practices for maintaining a consistently high velocity. Bellomo et al.'s [13] suggests that companies must develop *combined practices* such as "release planning with architecture considerations" and a "prototype/demo with [a] quality attribute focus" to balance process speed and stability. Efficient integrative activities between software development and other functions (e.g., business and operations) are needed in all stages of the product lifecycle.

Fitzgerald and Stol [3] have emphasised continuous integration (CI) between software development and its operational deployment (i.e., DevOps) as well as continuously assessing and improving the link between the business strategy and software development (i.e., BizDev). However, they do not explicitly define how such a business strategy should be carried out or how it is enacted in a continuous manner. In this paper, we want to clarify strategy planning activities and their interrelationship with CSE. The focus of our study is on investigating CSE prerequisites in product-focused (e.g., embedded systems) software development projects. Our research contributions are as follows. First, we review the literature on CSE, strategy planning and models for analysing holistic aspects of software-intensive product development. Second, we construct and specify the CRUSOE framework (Continuous inteRdependencies in prodUct-focused SOftware Engineering) for analysing CSE in software-intensive projects. Third, we conduct a case study from a smartphone product platform project to validate the framework. The research question for the case study is: *What are the prerequisites for using the CSE approach in software-intensive product development?* Finally, we synthesise the case-study findings with recent related studies by applying the CRUSOE framework.

2 Background

2.1 Holistic Models for Analysing the Development of Software-intensive Products

Various aspects of business and software ecosystems have been identified as important research topics in the context of CSE [1], [3]. In addition, as explicitly stated by Fitzgerald and Stol [3], continuity is required in all stages of the product lifecycle. Subsequently, they stress that it is necessary to constantly evaluate and improve software development interfaces with adjacent business-oriented activities. Previous CSE studies [3], [14] have suggested that delays in product development are often caused by a lack of holistic thinking and/or models for analysing software product development in a holistic manner. For example, inefficiencies and delays related to "handoffs" [15], as addressed by lean thinking, have been identified as a typical form of waste in software development, and thus the planning and engineering aspects of software product development should not be decoupled in separate silos for efficiency reasons. Still, information gaps and waste in between the business planning and development cycle could become evident when organisations are pushed towards faster (e.g., daily) or continuous software release cycles.

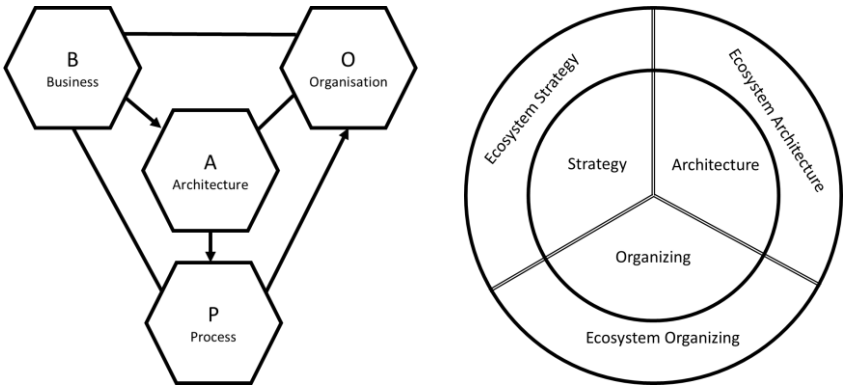

Fig. 1. The BAPO [16] and ESAO [17] models.

There are only a few documented approaches for analysing software product development in a holistic manner. As stated by Bosch et al. [17], few, if any, models exist that can analyse both the internal and ecosystem dimensions of research and development (R&D). The Business, Architecture, Process and Organisation (BAPO) [16] model (the left-hand side of Fig. 1) has been used for evaluating software product families and for analysing four main concerns addressed in product development: 1) how to make a profit from products, 2) the technical means to build the software, 3) responsibilities and relationships within software development and 4) the mapping of roles and responsibilities to organisational structures. As an update and extension to the BAPO model, the Ecosystem, Strategy, Architecture and Organizing (ESAO) [17] model (the right-hand side of Fig. 1) addresses both the *internal* and *ecosystem* dimensions for analysing company *strategy*, *architecture* and *organising*. Recently, both the

BAPO and ESAO model dimensions have been applied to describe the *evolutionary steps* in the transition from traditional development towards an innovation experiment system (IES) [18]. Fig. 1 illustrates the key dimensions of the BAPO and ESAO models. Later in this paper, we elaborate on the dimensions of the ESAO model as we construct the CRUSOE framework to analyse the CSE prerequisites. The CRUSOE framework illustrates possible interrelationships among ESAO dimensions that we consider to be important for CSE. These interrelationships (links) have also been previously addressed by the BizDev and DevOps concepts [3].

2.2 CSE in a Nutshell

When we use the term "CSE", we are referring to an emerging software development paradigm recently characterised by Bosch [14] (e.g., *Stairway to Heaven*, IES, and continuous deployment) and Fitzgerald et al. [3] (e.g., *Continuous*)*. IES is characterised by three coexisting aspects [19]: 1) "continuously evolving the software by frequently deploying new versions", 2) "customers and customer usage data play[ing] a central role throughout the development process", and 3) "development … focus[ing] on innovation and testing as many ideas as possible with customers". To be able to frequently deploy new versions, it is necessary to be capable of CD, which, as defined by Humble and Farley [4], is an array of software development, CI and continuous deployment patterns for enabling fast, reliable and automated deployments to production. Consequently, the CSE paradigm can be associated with the Lean Startup method [2] (i.e., rapid validated learning) and enterprise agility [20] *sensing* (i.e., the capability of continuously determining what is the most valuable feature for the customer) and *response* (i.e., the capability of continuously deploying new versions to production). Underlying agile principles emphasise collaborative work methods between business people and developers: "Business people and developers must work together daily throughout the project" [21]. Interdependencies between the business and development aspects are often addressed via the need for establishing frictionless information flow and decision making in product development [3], [15] (e.g., fast information flow, transparency and continuous planning practices). Continuous information flow and smaller batch sizes allow for better synchronisation of business planning with iterative release planning methods and tools [22] (e.g., iterative feature prioritisation and road mapping) and for breaking down requirements into small chunks that can be implemented, tested and deployed in approximately 1- to 2-week sprints, as emphasised in Scrum methodology [23]. Finally, shorter iterations allow for faster customer feedback cycles, thereby reducing the risk of developing the wrong product. In website development, continuous automated deployment can even enable rapid controlled experiments [7] (e.g., A/B tests) with end users.

2.3 Business Management Views on Strategy and Strategic Planning

When juxtaposing the ideology of CSE practices [1], [3] with the traditional, *rationalistic* view of strategy and organisations [24], they appear ill-matched. Whereas CSE stresses the importance of real-time actions and continuous change, the rationalistic

view of the strategy process [24] focuses on the creation of a structured future plan that is temporally and practically separated from its implementation [26,27]. This separation thus relies on the assumption of a comparatively static and predictable business environment that allows the rational managers [26] to first create a plan based on systematic scanning and positioning [25] and then implement it while having sufficient control over the consequences of their actions [27]. The usefulness of such theories for practice has been questioned, as they do not sufficiently reflect today's volatile organisational reality [28,29]. Therefore, there is a need for creating an understanding of strategy-making that better addresses the turbulent organisational reality.

Strategic planning is all about answering the questions of *where you are*, *where you want to be* and *how you get there*, as well as defining *how these aspects are connected* [30]. The strategy process varies across companies, but at the company level, it should be a continuous and issue-driven process [31]. In addition, the ways in which the strategy can be implemented fall into specific routines and work patterns that vary from firm to firm and between different types of firms [32]. Similarly, Brömmelstroet [33] defines how strategic planning phases can vary widely in terms of how they are organised (i.e., bottom up or top down), but all strategic planning processes can be seen as multilevel company processes in which planning actors work together towards a shared outcome. Even though the value of formal strategic planning has been strongly questioned [34], it is still an activity that is widely carried out in companies [35]. Formal strategy has power in affecting organisational actions and practices, as it defines roughly what is done and what is not done [36]. Eisenhardt and Brown [37] state that strategy should be seen as temporary, complicated and unpredictable and that strategy-making is a continuous process that is more oriented towards real-time operations than long-term stable goals [37].

From the software development perspective, and especially from the agile and lean software organisational perspective, Mavengere [38] clarifies that supply chain participants should have their own strategic plans which should be related to the whole supply chain's plan, but he does not go into detail about how these plans are created (i.e., the planning process in detail). On the other hand, Koenigsaecker [39] presents a lean strategic organisational process in which strategic planning is typically done once per year and is a learning experience in and of itself. In addition, monthly strategy deployment meetings are held to review progress and create opportunities for sharing lessons learned. The monthly strategy deployment reviews also help get the enterprise thinking about how to make the work process fundamentally better every month.

According to [40], among software development companies, the time frame for long-term strategic planning is commonly 3 years plus the current year. One of the case companies in this study used continuous strategic planning in which strategic plans were reviewed quarterly and monthly. The two other case companies reviewed and updated their strategic plans annually. Their strategic planning practices were closer to traditional project-based strategies than to continuous strategy practices – because the planning was performed annually, management approved plans which were then implemented for the rest of the year. Continuous strategy is extremely vital in a business environment that is constantly changing [41]. Suomalainen et al. [41] clarify that even though a strategy exists all the time, it should be iteratively and continuously updated

based on market and customer demands. For example, past financial crises have forced companies to realise that continuous planning is required throughout their organisations; not only at the software development level, but also at the strategic, business and financial levels.

3 The CRUSOE Framework

In this section, we introduce a new framework called CRUSOE that we later use to analyse the CSE prerequisites.

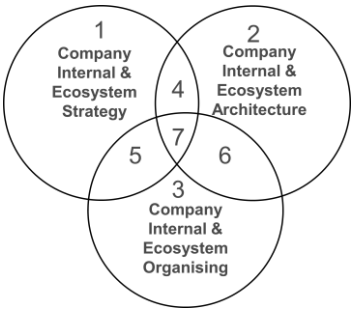

Fig. 2. Simplified illustration of the CRUSOE framework.

Fig. 2 provides a simplified illustration of the CRUSOE framework, which utilises the dimensions of the ESAO model [17]: 1) Strategy: ecosystem strategy (ES) and company internal strategy (CIS); 2) Architecture: ecosystem architecture (EA) and company internal architecture (CIA); and 3) Organising: ecosystem organising (EO) and company internal organising (CIO). The CRUSOE framework is an enhancement of the ESAO model that, according to the principles of CSE [1], [3], highlights the interdependencies between the three dimensions. These interdependencies are illustrated in Fig. 2, in which Areas 4, 5, 6 and 7 overlap with two adjacent dimensions. These areas illustrate interdependencies (e.g., integrative activities and combined practices) between dimensions. Area 7 illustrates the most overarching, holistic practices for company governance. Consequently, Areas 4, 5 and 6 illustrate more explicit integrative activities between dimensions, such as the BizDev concept addressed by Fitzgerald et al. [3]. We argue that the BizDev activities are associated with Areas 4, 5 and 7. According to Linden et al. [16], due to the interrelationships among the dimensions, any change in one dimension may have consequences in another dimension. These interrelationships have also been illustrated in the BAPO model with arrows and lines (Fig. 1). In the rationalistic view of the strategy process, and as stated by Bosch et al. [17], strategy should "idealistically" drive architecture and architecture should drive organising. However, in practice, "one has to allow for bi-directional dependencies" [17]. Moreover, the strategy must conform to *empiricism* and business realities; e.g., seemingly irrational customer behaviour and the existing constraints and capabilities of information technology (IT) and R&D. To summarise the notions referred to earlier about the need for a flexible and dynamic strategy process, we argue that the strategy process

should not be seen as the process governing "only business" or "all company processes", but rather as a process that can, and should, be continuously influenced by other company processes. Subsequently, there are also bi-directional interactions and dependencies illustrated as overlapping areas between each dimension and highlighted with numbers 4 to 7 in Fig. 2. Although the CRUSOE framework is used herein to analyse the CSE prerequisites, we anticipate that the framework could also be used for analysing other kinds of software-intensive product development processes.

In Table 1, we further elaborate on the key aspects of each area of the CRUSOE framework. Areas 1 to 3 are adopted from the ESAO model definitions. Our contribution to this area of study relates to the questions associated with establishing integrative practices among the various ESAO dimensions (Areas 4–7). These questions highlight relations between the ESAO dimensions and choices that the company has available to it. For example, there could be a vast number of choices for how to build a software-intensive product. However, only a few of the choices perhaps allow for the proper means with which to generate revenue in the future (e.g. providing a proper platform for a service business based on continuous deployment). Meanwhile, different ecosystems could provide different technical and procedural capabilities for CSE. In this paper, the purpose of the framework is to aid in analysing CSE in software-intensive product development.

Table 1. CRUSOE framework areas explained.

CRUSOE Framework Areas 1–7	**Analysis Scope: Company Internal (I)**	**Analysis Scope: Ecosystem (E)**
1* - Strategy	What are the options for how the company generates revenue now and in the future? [17]	What are the options that the company has available in its current role in the ecosystem? [17]
2* - Architecture	What are the options for technology choices, technical means and technical structures to build software-intensive products? [17]	What are the options for how to design interfaces between the company's internal architecture and related ecosystem partners, such as suppliers providing solutions and firms that build software on top of a product or platform? [17]
3* - Organising	What are the options for ways of organising work, ways of working, roles, responsibilities, processes and tools within software development? [17]	What are the options for how a company works with customers, suppliers, and ecosystem partners in terms of processes, tools used, ways of working and ways of organising the collaboration? [17]
4 - Strategy & Architecture interdependencies	What are the options to connect the internal strategy and architecture? E.g. what are the practices for continuously validating technology choices, technical means and technical structures that generate revenue now and in the future?	What are the options to connect the ecosystem strategy and architecture? E.g. what are the practices for continuously comparing different ecosystems' technical capabilities and interfaces that generate revenue now and in the future?

5 - Strategy & Organising interdependencies	What are the options to connect the internal strategy and organising? E.g. practices for continuously adopting efficient ways of organising work, ways of working, roles, responsibilities, processes and tools.	What are the options to connect the ecosystem strategy and ecosystem organising? E.g. practices for continuously validating investments in ecosystem processes, tools, ways of working and ways of organising the collaboration in the ecosystem.
6 - Architecture & Organising interdependencies	What are the options to connect the architecture and organising? E.g. practices for continuously refactoring technical structures that provide efficient organising ways of working, roles, responsibilities, processes and tools.	What are the options to connect the ecosystem architecture and organising? E.g. practices for providing appropriate technical structures for continuous deployments and collaboration with customers and ecosystem partners.
7 - Strategy & Architecture & Organising interdependencies	What are the overarching company governance options for connecting the company strategy with technical architectures and with ways of organising? E.g. practices for enabling a company culture of continuous improvement, experimentation and innovation.	What are the overarching company governance options for connecting the company strategy with ecosystem interfaces and ways of collaborating with customers and ecosystem partners? E.g. practices for enabling a culture of continuous improvement, experimentation and innovation with customers and ecosystem partners.

*Areas 1 to 3 are the same as in the ESAO model [17].

4 Case-study Design

To investigate CSE in a real software development context and to validate the CRUSOE framework, we applied the case-study method [42]. Our goal in conducting interviews was to gather data for a comprehensive understanding of the project's goal and of development and deployment practices. In addition, we asked about information flow and interactions among company stakeholders, customers and suppliers. In eight semi-structured face-to-face interviews, we asked participants to describe company strategy planning practices and product development processes. We also explicitly asked interviewees to identify the benefits and barriers associated with using the CSE approach to product development.

The unit of analysis in our case study was the project; i.e., developing a smartphone platform. Due to confidentiality reasons, we cannot provide a very detailed description of the features of the product. The platform included both software and hardware components. Consequently, this project is large, employing over 100 people directly inside the case company and also several partners involved in hardware and software development. We interviewed company personnel who were directly involved in the product development or company-level strategy planning. We also collected data from the organisation's public webpages for a better understanding of the project's purpose, company vision and strategic significance at the organisational level. We used convenience sampling for selecting interviewees and projects (i.e., those involving the company that

we could easily access). In addition, related workshop materials such as video clips, photos and field notes were collected and stored to support the analysis. All of the data was collected in 1- to 1.5-hour semi-structured interviews with eight company employees. The interviewed employees' job titles and responsibilities are summarised in Table 2. All interviews were recorded for later transcription and a qualitative data analysis was conducted using the NVivo tool [43]. The analysis was performed by considering the CRUSOE framework dimensions and the questions presented in Table 1.

Table 2. Interviewees' job titles and responsibilities.

Job title	Job responsibilities	Interview duration
Senior product manager	Responsible for delivering product programs to customers	117 min
Software platform product owner	Platform software component–related supervising	103 min
Quality manager	Product quality management including conformance to product safety standards and environmental regulations	105 min
Senior specialist	Design and implementation of continuous deployment processes and tools	104 min
President of the business segment	Chief Executive Officer for the business segment	67 min
Business developer	Product business development	78 min
Scrum master	Responsible for coordinating software development team work	86 min
Product manager	Responsible for coordination of the product program	92 min

5 Findings

In this section, we analyse the prerequisites for applying CSE through case-study data and by applying the CRUSOE framework areas introduced in section 3.

5.1 ESAO Overview: Strategy (1), Architecture (2) and Organising (3)

The case company offers a wide range of products, platforms and R&D services that typically involve radio technologies and wireless data transfer. Consequently, the company is involved in multiple ecosystems. The company is also an active contributor to several open source software projects. Recently, the company has expanded its product portfolio towards Internet of Things (IoT) solutions and data analytics services. Its main customers are from the B2B domain, including both private- and public-sector organisations. Hence, when analysing the strategy and connected software development practices, it is necessary to explicitly specify which product category and customer segment is under analysis. The company has an established position in manufacturing systems for public safety and the military. However, more recently, the company has adopted a strategy for developing product platforms that allow tailored products to be created that

could be sold to consumer markets (B2C). The smartphone platform project that we analysed in this case study is an example of this type of product.

We consider this company to be a very interesting research context for CSE because the company has many products in their portfolio and there are many different kinds of customers involved. This clearly addresses the challenges in defining product- and company-level processes.

In this way, the company has systematically developed capabilities to adapt to different customer contexts and software ecosystems and also to rotate employees among projects to develop the employees' skills and technical knowledge. Consequently, the interviewees often referred to other projects that they were aware of or had worked on previously. The interviewees emphasised that although there was a company-wide defined product development process that was, in principle, guiding all company projects, individual projects often improvised; i.e., very different methods were used or they worked in collaboration with specific customers and other ecosystem partners. Hence, when asked about the benefits of and barriers to using the CSE approach in product development, the interviewees considered opportunities for using the CSE approach as highly context-sensitive. Different products and customer segments were considered as having very different CSE prerequisites. These customer segments could be characterised by two extremes: "conservative public-sector customers" and "fast-moving private-sector customers". Different product segments could be characterised by "large and complex multivendor legacy systems" and "compact consumer products".

5.2 CRUSOE Area 4: Connecting Strategy and Architecture for CSE

This section analyses the options for connecting the internal and ecosystem strategy with the architecture for CSE; i.e., interfaces and technical structures to create revenue. The Android operating system (OS) [44] was selected as the software baseline for the smartphone project. When considering the various smartphone OS ecosystems, the Android OS currently has by far the largest market share, dominating markets with over 80% of the total market share [45]. Hence, the selected platform also allowed for opportunities to generate future revenue as additional product applications could be continuously provided via the Google Play store (http://play.google.com/store/apps).

The project budget and product strategy management-level planning occurred in monthly cycles. The development teams organised their work into 2-week sprints that also guided and synchronised several planning-related activities such as the planning associated with validating technical interfaces, features and release-content prioritisation. The project actively used a CI system, thus new versions of the product could be produced several times a day. These new software versions were mainly used for internal testing purposes. Nevertheless, the project was considered capable of continuously delivering new versions to demonstrate the latest interfaces and product features for customers and ecosystem partners. We consider that interdependency with business planning (i.e., *continuous synchronisation of development sprints with continuous strategic planning and budgeting cycles*) was clearly a key prerequisite for using the CSE approach because it allowed for continuous feedback cycles and transparency in terms of how the project was progressing.

5.3 CRUSOE Area 5: Connecting Strategy and Organising for CSE

This section analyses the options for connecting the company's internal and ecosystem strategy with internal and ecosystem organising for CSE. The interviewees emphasised how the transition towards CSE was a strategic decision that governed practices relating to how products were designed and how the company was organised. Moreover, adopting CSE was considered as a competitive advantage in that it provides better transparency, efficiency and flexibility when working with different customer projects.

When considering the smartphone project-level CSE strategy, we could see that the CSE approach was mainly limited to software development practices such as CI and test automation. The interviewees also pointed out that while the company was a small player in the Android OS ecosystem, it had to adjust its internal plans according to supplier schedules and technology roadmaps. Meanwhile, larger smartphone vendors had more power to affect their supplier's plans. Consequently, the interviewees considered *supply chain support* as a key CSE prerequisite and also as a key hindrance to not being able to fully implement the CSE approach (e.g., CD in the project).

To summarise our findings regarding strategic dimension interdependencies with organising, we consider that currently, the company's internal strategy is to adopt a CSE capability. This is also the main driver for using CSE in the company projects. The company had already made significant investments in terms of promoting CI and CD solutions (e.g., automation) for use in software-intensive product projects. As concrete evidence of the strategic decision, the company had established a *team of experts to implement the technical CI and CD framework (toolbox)* so that it could be adopted in all company projects. This expert team was also in charge of *coaching projects on how to adopt CI development practice*. We consider this activity as a key interdependency between the strategy and organising aspects that was clearly contributing to the company's transition towards CSE.

When considering other aspects of the ecosystem, some of the existing customers showed very little interest in using the CSE approach in product development projects. So far, only a few private-sector B2B customers had insisted on using the CSE approach in the development of consumer products. One interviewee pointed out that some customers have very little knowledge of and experience in agile product development methods. Meanwhile, public-sector customers in particular often have established and formal staged processes for acquiring software-intensive products; e.g., communication equipment for the military and for public safety must go through rigorous testing and certification processes before it can be put into actual use. Consequently, an *educated and motivated customer* was considered as an important CSE prerequisite.

5.4 CRUSOE Area 6: Connecting Architecture and Organising for CSE

This section analyses the options for connecting the company's internal and ecosystem architecture with organising for CSE. The interviewees considered the selected Android OS platform architecture (e.g., the hardware and software technology platforms and associated tools) to provide an adequate technical capability in terms of delivering software to end users rapidly and *over-the-air* (OTA) [46]. We consider this as

a key prerequisite associated with the interdependency between the organising and architecture dimensions.

Several interviewees emphasised how the technical capability of providing updates continuously must be aligned with *quality assurance practices and cycles for testing*. The interviewees stated that the prerequisite of frequently delivering new updates to the end user would have to precede the rigorous internal testing period. One interviewee stated that some bugs can be identified only after using the product for a long period of time, which could be a challenge for CD. System updates that require rebooting or interrupting end-user product usage were also considered problematic since they could easily annoy end users. Moreover, updates in business and critical safety systems cannot interrupt or compromise the availability of the service. Consequently, it is a prerequisite to *minimise breaks in service availability and deliver updates so that the end user is not interrupted*.

We consider the interviewees' previous experiences in using rapid prototyping and demoing to be somewhat controversial. Although rapid prototyping was acknowledged as very important and good practice for identifying key functionalities and requirements for the product in the early phases of development, there were also drawbacks such as undisciplined processes for bug fixes and feature prioritisation. Two interviewees identified the problem known as the HiPPO; i.e., the "Highest Paid Person's Opinion" [7]. One interviewee emphasised how the processes for building prototypes and actual products were very different. Although prototypes can often be used for demonstrating new functionalities, they do not typically meet the proper internal quality criteria that are required for real products. Consequently, some managers are often too optimistic about how much work is still to be done in order to finalise the product. Therefore, the company needs to develop balanced processes that integrate speed and stability in order to build actual products in an experimental manner. An important CSE prerequisite is thus that the company *increases its understanding of the experimentation process* and that it *reviews current best practices, milestones and checklists for product development*. Methods for *managing technical debt* are particularly important prerequisites for CSE.

5.5 CRUSOE Area 7: Overarching Governance for CSE

Finally, this section analyses the options for overarching company governance for CSE. As stated earlier, we considered that the company management had clearly made a strategic decision to promote the CSE approach in all of its company projects. Two interviewees stated that any investment promoting CSE was an important "investment for the future". The interviewees considered it important to establish and improve systems for company-wide information transparency and for the real-time availability of customer feedback and product quality metrics. We consider the *company's senior management's commitment* to promoting the capability of using the CSE approach in product development as a key CSE prerequisite. We can identify several activities that indicate the company management's commitment to investing in CSE, such as an investment in people, tools and processes for enabling the rapid deployment of CI in all company projects; increasing test automation coverage; developing methods for end-

user data collection (product platform instrumentation for data collection); systematising customer feedback collection (e.g., customer surveys); and developing tools and processes for analysing user data and sharing information internally in the company via IT systems. The company also arranged regular sessions for employees to promote *internal experience sharing and bottom-up strategic planning*.

5.6 Findings' Summary

In summary, we identified the following key CSE prerequisites in a smartphone platform project: *1) customer education and motivation, 2) software ecosystem support, 3) supply chain stakeholder support, 4) leadership commitment, 5) process rigor for experimentation, 6) quality assurance process cycle duration, 7) technical debt management, 8) OTA updates with minimised breaks in service availability, and 9) internal experience sharing and bottom-up strategic planning*. The CRUSOE framework has significantly helped us to systematically categorise and more clearly articulate the prerequisites for using CSE in the case-study (smartphone platform) project. Based on our case-study findings, applying CSE to product-oriented development can involve a complex organisational change within and between software development and business activities. Whereas the adoption of technical infrastructure and development practices is an important starting point for CSE, one should also consider the company's culture, leadership and key stakeholder relations.

6 Discussion

This section continues our interpretation of the case-study results and synthesis of the prerequisites for CSE in software-intensive projects together with recent related empirical studies on the research topic. In Table 3, we list the main findings from the studies.

Table 3. Recent empirical studies on obstacles and challenges for CSE.

Leppänen et al. [8] Obstacles for CD: "resistance to change", "customer preferences", "domain constraints", "developer trust and confidence", "legacy code considerations", "[test automation] duration, size and structure", "different development and production environments", "manual and non-functional testing".
Lindgren et al. [9] Domain-independent challenges associated with continuous experimentation: "organizational culture", "availability and sharing of data", "data analysis", "identifying metrics", "release cycle speed", "defining product roadmap", "time [resources]", funding [resources], "technical obstacles".
Rissanen et al. [10] B2B specific challenges of CD: "technical challenges", "customer challenges", "procedural challenges".
Olsson et al. [47] Challenges identified in the adoption of CD: "diverse adoption of agile practices among teams", "complexity of team resource allocation", "dependence on resources outside of the team", "difficulties in analyzing and maintaining automated tests", "difficulty in removing or reducing old tests", "difficulties in establishing efficient rollback mechanisms", "no effective mechanism for analysis of customer data", "lack of understanding about feature use", "no pro-active use of customer data".

6.1 Synthesising the Prerequisites for CSE

As identified in previous studies referred to in Table 3, the challenges associated with CSE are often multidimensional. Incorporating pilot or lead customers in the development process and business-model change is clearly a commonly identified prerequisite for CSE that involves both the company's internal and ecosystem processes. In addition, the supply chain (e.g., the component and technology platform suppliers) must be incorporated in the development cycle to be able to continuously integrate all of the product components and test the product.

As stated by Facebook's release engineering manager, "Mobile deployments are more challenging than Web deployments because we don't own the ecosystem, so we can't do all the things that we would normally do" [12]. A company's role in the ecosystem can significantly affect how feasible it is to use the CSE approach. Earlier case studies have identified ecosystem-related challenges in CD, such as dependencies on the hardware platform component supply process and interrelated customer processes, such as periodic tendering, periodic budgeting, product piloting and acceptance testing practices. These ecosystem-related constraints could require overarching changes in a company's business model, architecture and organising.

As identified in previous case studies, the product architecture must provide capabilities for adequate componentisation for partial and staged release, including roll-back mechanisms. Additionally, internal and ecosystem stakeholder needs must be addressed in the architecture; e.g., enabling CI and partial deliveries of products without updating the whole product. Technical capabilities for CD and continuous testing in production-like (staging) test environments are prerequisites for pushing reliable, bug-free releases into the customer's production environment. While deployment to the production environment can technically be similar to deployment to the staging environment, it involves risks directed towards the customer's business. Consequently, deployment to production must ensure the rapid identification of any abnormalities in the system and, if needed, an immediate roll-back to the previous functional configuration.

The experimentation of new functionalities on Facebook is conducted via "canarying" [12]; i.e., collecting user data from alpha and beta test groups before engaging in mass deployments, where changes are pushed out to all production systems (servers). As identified by Rahman et al. [6], CD is used almost solely for deploying websites. Hence, we consider that Internet- and cloud-based virtualisation technologies provide the best technical capabilities for CSE. Cloud-based services are nowadays often integrated with embedded systems and consumer products (e.g., sports-tracking and health-monitoring applications). Consequently, this trend may also enable the increasing use of the CSE approach in the future.

As identified in previous studies, it is a prerequisite that a project must have the capability for continuous integration and testing of the whole product. The existing system for CI can often be incrementally upgraded for automated delivery; i.e., a continuous release process also involving the release of the decision-making (e.g., acceptance) process for customer deliveries. As the release cycle may shorten quite dramatically, it is paramount that the user experience (UX) and system design functions are integrated into the development team to enable the efficient planning of features.

From a release planning point of view, in traditional software projects, the customer's role is to be involved in the planning and freezing of requirements at the start-up stages of the project. Consequently, the customer accepts project delivery based on customer validation at the end of the project. From the development process point of view, a customer's role and responsibilities could change radically when moving from periodic traditional methods towards CSE. Although a product owner can represent the customer, the actual customer must also take a more active stakeholder role throughout the project because deliveries can be experimental (tentative and prone to change) and they can occur more frequently. Subsequently, the delta between deliveries to be accepted by the customer is smaller. CD, however, depending on the industry domain, may radically impact how value is delivered to users.

Moving away from periodic delivery (large releases) often leads to inevitable business-model transformation. CD involves customer use and purchase processes. Internal business planning, requirement prioritisation and release delivery processes are also involved. However, organisational capabilities for CD must be considered. As stated by Facebook's release engineering manager, "CD works for small teams, within 20 to 30 changes per day". Hence, more complex systems and large projects (i.e., with more than 100 developers) working in a common code base may have to settle for a lower continuous deployment frequency.

6.2 Study Limitations, Future Studies and Threats to Research Validity

The main limitation in our study relates to the constructed CRUSOE framework because it has only been validated through one case study. More empirical studies are needed to validate the framework's true utility. Moreover, as our goal was to understand the holistic prerequisites for CSE, we acknowledge that is it also beneficial to investigate explicit CSE practices and also dependencies within individual dimensions. Consequently, for future studies, it might be useful to either scope the research topic and/or apply the framework in a different software project context. Applying the CRUSOE framework in a different context would also provide information on how to improve the framework.

The main threats to validity and the limitations in case studies are typically addressed by the data-collection methods, data interpretation, reliability and generalisability of the results. Due to the nature of the case-study method, the results are not generalisable to the whole industry. To mitigate the risks associated with construct validity (e.g., misunderstandings and misinterpretations between the interviewer and interviewees), we started each interview with a 5- to 10-min introduction on the research topic and key concepts of CSE (e.g., CD and continuous experimentation). The semi-structured interview method allowed us to ask clarifying questions throughout the interview. We also arranged for an interactive feedback session with the case company representatives to share interview summaries and to get feedback on our analysis; i.e., how we (the researchers) interpreted the interviewees' answers. The company stakeholders confirmed that our findings were correct.

7 Conclusion

In this paper, we have applied the case-study method to a smartphone platform project to investigate the prerequisites for CSE. First, we specified the CRUSOE framework in terms of it allowing for a holistic, systematic and structured investigation of the prerequisites for CSE. We consider that the CRUSOE framework enabled us to more precisely articulate and analyse the prerequisites for CSE. The framework can further aid in developing estimations regarding the changes that are needed when moving from traditional product development to CSE. Finally, the case-study findings were synthesised alongside related recent studies. The results indicate that using the CSE approach in product-focused software development could involve several areas within and between the strategy, architecture and organising dimensions. Moreover, novel integrative activities are needed for eliminating disconnects and for balancing speed and stability (e.g., feature-driven development and managing cumulating technical debt). Although these are initial ideas on how to organise continuous experimentation in software development, rigorous processes are needed between the customer and the supplier. Our case study indicated that opportunities for using the CSE approach in product development are often context-sensitive (e.g., customer and product dependent). Moreover, customer motivation and ecosystem support for CSE are important. Although the CSE approach is mostly used for website development, more systematic use of CSE could enhance the competitiveness of product-oriented companies. However, more prescriptive models and best practices are clearly needed to describe how CSE should be implemented in product-oriented software development.

Acknowledgments

This work was supported by TEKES as part of the Need for Speed Project (http://www.n4s.fi/) of DIMECC (Digital, Internet, Materials & Engineering Co-Creation).

References

1. Bosch, J.: Continuous Software Engineering: An Introduction. Contin. Softw. Eng. (2014).
2. Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. (2011).
3. Fitzgerald, B., Stol, K.J.: Continuous software engineering: A roadmap and agenda. J. Syst. Softw. (2015).
4. Humble, J., Farley, D.: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. (2010).
5. Fagerholm, F., Guinea, A.S., Mäenpää, H., Münch, J.: The RIGHT Model for Continuous Experimentation. J. Syst. Softw. (2016).
6. Rahman, A.A.U., Helms, E., Williams, L., Parnin, C.: Synthesizing Continuous Deployment Practices Used in Software Development. In: 2015 Agile Conference. pp. 1–

10. IEEE (2015).
7. Kohavi, R., Henne, R.M., Sommerfield, D.: Practical guide to controlled experiments on the web. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07. p. 959. ACM Press, New York, New York, USA (2007).
8. Leppänen, M., Mäkinen, S., Pagels, M., Eloranta, V.-P., Itkonen, J., Mäntylä, M. V, Männistö, T.: The highways and country roads to continuous deployment. IEEE Softw. 32, 64–72 (2015).
9. Lindgren, E., Münch, J.: Software Development as an Experiment System: A Qualitative Survey on the State of the Practice. Springer International Publishing, Cham (2015).
10. Rissanen, O., Münch, J.: Transitioning Towards Continuous Delivery in the B2B Domain: A Case Study. Springer International Publishing, Cham (2015).
11. Lwakatare, L.E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H.H., Bosch, J., Oivo, M.: Towards DevOps in the Embedded Systems Domain: Why is It So Hard? In: 2016 49th Hawaii International Conference on System Sciences (HICSS). pp. 5437–5446. IEEE (2016).
12. Adams, B., Bellomo, S., Bird, C., Marshall-Keim, T., Khomh, F., Moir, K.: The Practice and Future of Release Engineering: A Roundtable with Three Release Engineers. IEEE Softw. 32, 42–49 (2015).
13. Bellomo, S., Nord, R.L., Ozkaya, I.: A study of enabling factors for rapid fielding combined practices to balance speed and stability. Proc. - Int. Conf. Softw. Eng. 982–991 (2013).
14. Bosch, J.: Continuous Software Engineering. Springer International Publishing, Cham (2014).
15. Poppendieck, M., Poppendieck, T.: Implementing Lean Software Development: From Concept to Cash. (2006).
16. Van Der Linden, F., Bosch, J., Kamsties, E., Känsälä, K., Obbink, H.: Software Product Family Evaluation. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 3154, 110–129 (2004).
17. Bosch, J., Bosch-Sijtsema, P.: ESAO: A Holistic Ecosystem-Driven Analysis Model. Springer International Publishing, Cham (2014).
18. Olsson, H.H., Bosch, J.: Climbing the stairway to heaven: Evolving from agile development to continuous deployment of software. In: Continuous software engineering. pp. 15–27 (2014).
19. Bosch, J.: Building products as innovation experiment systems. Springer Berlin Heidelberg, Berlin, Heidelberg (2012).
20. Overby, E., Bharadwaj, A., Sambamurthy, V.: Enterprise agility and the enabling role of information technology. Eur. J. Inf. Syst. 15, 120–131 (2006).
21. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Agile Manifesto, http://agilemanifesto.org/.
22. Ruhe, G.: Product Release Planning Methods, Tools and Applications. Auerback Publications, Taylor and Francis Group, LLC (2010).
23. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. (2001).
24. Hutzschenreuter, T.: Strategy-Process Research: What Have We Learned and What Is Still to Be Explored. J. Manage. 32, 673–720 (2006).

25. Tsoukas, H., Chia, R.: Philosophy and Organization Theory. Emerald Group Publishing Limited (2011).
26. Vaara, E., Kleymann, B., Seristo, H.: Strategies as Discursive Constructions: The Case of Airline Alliances. J. Manag. Stud. 41, 1–35 (2004).
27. MacKay, R.B., Chia, R.: Choice, Chance, and Unintended Consequences in Strategic Change: A Process Understanding of the Rise and Fall of NorthCo Automotive. Acad. Manag. J. 56, 208–230 (2012).
28. Sandberg, J., Tsoukas, H.: Grasping the logic of practice: Theorizing through practical rationality. Acad. Manag. Rev. 36, 338–360 (2011).
29. Chia, R.: A "Rhizomic" Model of Organizational Change and Transformation: Perspective from a Metaphysics of Change. Br. J. Manag. 10, 209–227 (1999).
30. Bryson, J.M.: Strategic Planning for Public and Nonprofit Organizations: A Guide to Strengthening and Sustaining Organizational Achievement. (2011).
31. Bogsnes, B.: Implementing Beyond Budgeting: Unlocking the Performance Potential. (2008).
32. Nordqvist, M., Melin, L.: The promise of the strategy as practice perspective for family business strategy research. J. Fam. Bus. Strateg. 1, 15–25 (2010).
33. Brömmelstroet, M. te: Performance of Planning Support Systems. Comput. Environ. Urban Syst. 41, 299–308 (2013).
34. Mintzberg, H.: The Rise and Fall of Strategic Planning. (2000).
35. Whittington, R., Cailluet, L.: The Crafts of Strategy. Long Range Plann. 41, 241–247 (2008).
36. Balogun, J., Huff, A.S., Johnson, P.: Three Responses to the Methodological Challenges of Studying Strategizing*. J. Manag. Stud. 40, 197–224 (2003).
37. Eisenhardt, K.M., Brown, S.L.: Competing on the Edge: Strategy as Structured Chaos. Long Range Plann. 31, 786–789 (1998).
38. Mavengere, N.B.: Information technology role in supply chain's strategic agility. Int. J. Agil. Syst. Manag. (2013).
39. Koenigsaecker, G.: Leading the Lean Enterprise Transformation. (2009).
40. Suomalainen, T.: Defining Continuous Planning Through a Multiple-Case Study. Springer International Publishing, Cham (2015).
41. Suomalainen, T., Kuusela, R., Tihinen, M.: Continuous planning: an important aspect of agile and lean development. Int. J. Agil. Syst. Manag. 8, 132 (2015).
42. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14, 131–164 (2008).
43. QRS International: NVivo, http://www.qsrinternational.com/, (2016).
44. Google: Android, https://www.android.com/.
45. Gartner: Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015, http://www.gartner.com/newsroom/id/3215217.
46. Hoffman, T.L.: Over-the-air programming of wireless terminal features, https://www.google.com/patents/US6622017, (2003).
47. Olsson, H.H., Bosch, J.: Towards agile and beyond: An empirical account on the challenges involved when advancing software development practices. Lect. Notes Bus. Inf. Process. 179 LNBIP, 327–335 (2014).