Practitioners' Eye on Continuous Software Engineering: An Interview Study

Jan Ole Johanssen
Technical University of Munich
Department of Informatics
Munich, Germany
jan.johanssen@in.tum.de

Anja Kleebaum
Heidelberg University
Institute of Computer Science
Heidelberg, Germany
anja.kleebaum@informatik.uni-heidelberg.de

Barbara Paech
Heidelberg University
Institute of Computer Science
Heidelberg, Germany
paech@informatik.uni-heidelberg.de

Bernd Bruegge
Technical University of Munich
Department of Informatics
Munich, Germany
bruegge@in.tum.de

ABSTRACT

Continuous software engineering (CSE) emerged as a process that is increasingly applied by practitioners. However, different perceptions of CSE among practitioners might impede its adoption in industry. We aim to support practitioners by giving a comprehensive overview of current CSE practices. Our observations provide guidance for practice on how to establish, assess, and advance CSE in their company. We conducted an interview study with 24 practitioners from 17 companies during 20 interviews. Following a semi-structured approach, we asked for their definition of CSE, most relevant elements for CSE, their experiences, and plans for further additions to their CSE process. From the practitioners' statements, we identified five perspectives on CSE and found tool- and methodology-driven definitions most prevalent. Automated tests, involved users, and a shared ruleset are perceived as most relevant for CSE. Practitioners' positive experiences with CSE are more frequent than negative ones; however, more than half of the responses were neutral. Practitioners' future plans focus on enhancement, expansion, and on-demand adaption of current practices. We conclude that CSE remains partially difficult to capture for practitioners. Therefore, we structure CSE in a model, the *Eye of CSE*.

CCS CONCEPTS

• **General and reference → Empirical studies**; • **Software and its engineering → Agile software development**;

KEYWORDS

Continuous Software Engineering, Interview Study, Continuous Integration, Continuous Delivery, Experience Report

ICSSP '18, May 26–27, 2018, Gothenburg, Sweden

ACM Reference Format:
Jan Ole Johanssen, Anja Kleebaum, Barbara Paech, and Bernd Bruegge. 2018. Practitioners' Eye on Continuous Software Engineering: An Interview Study. In *ICSSP '18: International Conference on the Software and Systems Process 2018 (ICSSP '18), May 26–27, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3202710.3203150

1 INTRODUCTION

Continuous software engineering (CSE) bundles activities, such as continuous integration and delivery, to enable continuous learning and improvement by frequently iterating on software increments [2, 4]. This classifies CSE as a software engineering process [5].

Krusche and Bruegge address the need for a formal description of the continuous aspects of CSE to enable its adoption in real world settings [9]. Similarly, researchers highlight challenges in the introduction and enhancement of CSE in companies [14, 21]. Practitioners need a starting point in order to approach CSE, since they might lack insight into interrelationships, potential risks, and challenges [15, 16]. Comparing their CSE process with what other companies have implemented allows practitioners to assess their own progress. Likewise, practitioners can benefit from guiding principles to establish CSE in their company [4].

In order to provide such guidance, we conducted 20 interviews with 24 practitioners from 17 companies between April and September 2017. We studied various aspects that are related to CSE, e. g., how companies understand CSE or how they utilize usage and decision knowledge. Our overall goal is the integration of usage and decision knowledge in CSE to support software evolution [6, 7]. For this paper, however, we focus on results regarding the general perception of CSE by practitioners in the industry.

The contribution of our work is threefold. First, we report on study data that provides insights into the characteristics of companies, practitioners, as well as projects in the context of CSE. Second, we describe a set of 19 observations that are derived from practitioners' responses. The observations detail practitioners' perspectives on CSE, elements of CSE perceived most relevant, CSE experiences, and strategies for implementing future plans for CSE. Third, we introduce and discuss a model to describe the components of CSE—the *Eye of CSE*. The model contains CSE elements and categories, highlights relations among them, and aims to support practice.

This paper is structured as follows. Section 2 describes how we derived CSE elements and categories. We introduce our research questions in Section 3 along with the applied research method. Section 4 provides descriptive statistics regarding companies, practitioners, and projects. Major observations of the interview study are presented in Section 5. We discuss our observations in Section 6 by describing the *Eye of CSE* and elaborating on threats to the validity of our study. In Section 7, we provide an overview of similar studies in the context of CSE. Section 8 concludes the paper.

2 CSE ELEMENTS

We examined the *Stairway to Heaven* by Bosch *et al.* [2] and the work by Fitzgerald and Stol [4] to acquire a preliminary list of characteristics that are typical for CSE. We refer to them as *CSE elements*, which we classify into six *CSE categories* as listed in Table 1.

Table 1: CSE elements and categories derived from [2, 4].

CSE Categories	CSE Elements
User	Involved users and other stakeholders; learning from usage data and feedback; proactive customers
Software Management	Agile practices; short development sprints; continuous integration of work; continuous delivery; continuous deployment of releases
Development	Continuous planning activities; continuous requirements engineering; focus on features; modularized architecture and design; fast realization of changes
Code	Version control; branching strategies; fast commit of code; code reviews; code coverage
Quality	Automated tests; regular builds; pull requests; audits; run-time adaption
Knowledge	Sharing knowledge; continuous learning; capturing decisions and rationale

The frequent involvement of users is a major concept in CSE. Thus, we introduced *user* as a CSE category that refers to both customers who commissioned a project and end-users. *Software management* includes practices concerning the overall software process. The *development* category is composed of more specific development activities, such as requirements engineering and design excluding implementation and quality assurance. The *code* category includes implementation-related practices, such as *version control* and *branching strategies*. We bundled activities such as *audits* and *pull requests* in the *quality* category. The *knowledge* category collects practices supporting the overall knowledge management.

We acknowledge that the allocation of elements to categories is not always straightforward. For instance, we consider arguably technical practices, such as continuous delivery, under software management, in order to highlight their impact on the overall CSE process. Similarly, we include code reviews in the code category to emphasize their operational character, while pull requests, i. e., merging code, are viewed as quality-related tasks. We refined the list of CSE elements and categories based on the interview observations and discuss the issue of ambiguities in Section 6.

3 STUDY DESIGN

This section describes our research questions and research method.

3.1 Research Questions

The overall goal of this interview study is to understand how companies apply CSE during software evolution. From this goal, we derived the following four research questions.

RQ1: How do practitioners define CSE? With this core question, we intend to learn about practitioners' perception of CSE. Further, we want to know whether practitioners define a threshold that needs to be passed before a company can claim to practice CSE.

RQ2: Which CSE elements are perceived as most relevant by practitioners? To understand the perception of CSE in more detail, we asked the practitioners about the three CSE elements most relevant to them. In addition, we collected applied tools.

RQ3: What are practitioners' experiences with CSE? With this research question, we want to reveal positive, neutral, and negative experiences with the CSE elements. This is of particular interest for companies that plan to adopt them as well.

RQ4: What are practitioners' future plans for CSE? We asked for planned additions in the short and long term in order to understand trends of future CSE elements adoption.

3.2 Research Method

We performed a semi-structured interview study [13, 17] and organized our study into the three phases *design and planning*, *data collection*, and *data analysis*. Each phase is described subsequently; the first two authors were equally involved in each of the phases.

3.2.1 Design and Planning. We conducted a semi-structured interview study, since we focus on knowledge that resides in the minds of practitioners rather than in documents [17]. Furthermore, interviews allowed us to clarify problems right away and collect more information from the practitioners. We prepared a questionnaire containing descriptive data questions and interview questions derived from the research questions. The questionnaire contained six open questions that included sub-questions to further stimulate verbose answers by the practitioners. Figure 1 states example questions. We actively encouraged the interviewees to give detailed answers. We planned 90 minutes for the interview, which included research questions that are not further addressed in this paper.

We assembled a list of companies who to our knowledge apply the majority of our preliminary collection of CSE elements (Table 1). We formulated a template request mail for scheduling an interview. We attached a slide deck sketching the CSE elements and categories. We asked interview partners to agree to the interview only, if they have worked in at least one project that applied the majority of the CSE elements. We did not restrict interview partners by role descriptions. However, we provided examples of roles that we preferably address, e. g., developer or project manager.

Since two authors conducted the interviews simultaneously, we set up an interview guideline to ensure comparability. Besides the interview questions, the guideline comprised remarks to increase the questions' understandability. We run two dry runs with colleagues who have industry experience to practice the interview procedure.

3.2.2 Data Collection. We sent the interview requests to 22 companies, of which 18 replied. The first two authors conducted 19 interviews between April and June 2017 and one additional in September 2017. Half of the interviews were conducted in person; the others via phone. Descriptive data about the study participants are provided in Section 4. The interviews took 70 minutes on average and were audio-recorded with the permission of the interviewees. We transcribed the audio recordings and sent the transcripts to the interviewees to correct misunderstandings. We guaranteed anonymity of practitioners by only publishing aggregated results.

3.2.3 Data Analysis. Two authors of this paper analyzed the transcripts [18]. As shown in Figure 1, we used a *qualitative data analysis* software to apply two stages: one to allocate answers to research questions and one to code CSE elements. The allocation to research questions was made at sentence-, the coding at word-level.

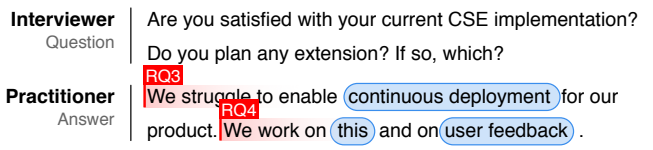

Figure 1: An example interview extract. Practitioners' answers may refer to multiple RQs (red), while these in turn may contain multiple occurrences of CSE elements (blue).

For the first stage, i. e., allocating answers to research questions, both authors analyzed a single interview to measure the *intercoder reliability*; representativeness and completeness were criteria for choosing the interview. One author found 85 answers related to the four research questions, the other 77. Given the total number of 162 instances, the authors matched in 134 and mismatched in 28, leaving a result of 82.72 % in equally allocated answers. The 28 mismatched instances were jointly discussed and resolved by mutual consent. The discussion necessary for this purpose strengthened the shared understanding of the authors. We observed that almost all mismatches were caused by a missing allocation, not by the allocation to different research questions. In case of doubt, we agreed on allocating multiple research questions to an answer to prevent information loss. The allocated interview answers form units for the coding stage. After the allocation, we updated our initially elected list of CSE elements in Table 1, based on the insights we derived from the answers. The updated collection is presented in Section 6.1, along with the rationale for the applied changes.

The second stage covered the coding of answers to research questions using codes for every CSE element from the updated collection. This was done manually, since searching for keywords is insufficient, given that practitioners use varying formulations to describe the same aspect. For instance, in Figure 1, *this* refers to the CSE element *continuous deployment*, leaving the decision to add a code up to the author. Occasionally, the authors had to decide which code to use based on the context. We practiced the coding and agreed to prefer to code an instance if in doubt. Each author coded their own interviews. We analyzed the results quantitatively (Figure 3 and 4) and collected qualitative answers to identify emerging themes, as described in Section 5. An initial version of this paper was sent to the interviewees to validate the interview results.

4 DESCRIPTIVE STUDY DATA

We report descriptive data about the companies, practitioners, and projects that were analyzed. Figure 2 visualizes a summary of the following subsections. Overall, we interviewed 24 practitioners from 17 companies during 20 interviews. One company was interviewed twice, another one three times. We aimed for a diverse composition of interviews through companies varying in size, practitioners of different roles, and projects from various domains.

4.1 Companies

Four companies (24 %) are considered as small and medium-sized enterprises[1] (SME), which means a maximum staff headcount of 250. We call the remaining 13 companies corporations, while they could be further categorized in companies of up to 2.000 (8), around 50.000 (2), and 100.000 or more employees (3). We report the overall number of employees, since we assume that the CSE process is not limited to a specific role from the development team.

Seven (41 %) of the interviewed companies offer consultancy services, mostly to other business, while ten (59 %) companies develop software products for the consumer and business markets.

4.2 Practitioners

Based on their role description, we grouped the 24 practitioners into five categories: *CSE specialists* (21 %), a role with a reference to CSE, e. g., a continuous deployment manager or a DevOps engineer, *developers* (25 %), *project managers* (25 %), a role with project-focused responsibilities, and *technical leaders* (25 %), a role with technical-focused responsibilities; one practitioner reported as an executive director. On average, the practitioners spent two years in the respective role. All practitioners hold a Bachelor or Master degree with three-quarters in a field in or close to computer science. With one lacking response, on average, 23 practitioners have an experience in IT projects of 10 years and participated in 19 IT projects.

We asked the practitioners whether they see themselves in one of the following three roles: *using*, *defining*, or *planning* CSE. Practitioners in a *using role* frequently apply and benefit from CSE, e. g., developers who regularly commit code. Practitioners in a *defining role* set rules on how CSE elements are applied, e. g., whether a code integration is triggered by an event, such as a commit, or on an hourly basis. It is the responsibility of a *planning role* to think ahead and take future addition to a CSE environment into consideration. Many practitioners saw themselves in multiple roles: 14 using, 15 defining, and 14 planning. Seven practitioners reported to adhere to all three roles—*I am everything* or *straight through*. Six other practitioners grouped themselves into two roles at the same time: In one role, they collect knowledge regarding a CSE element and in the other role they share it with other practitioners. Six practitioners adhered to a single role only. Two developers saw themselves solely in a using role; a project manager and a CSE specialist classified themselves in a defining and planning role, respectively. Two other practitioners did not see themselves in one of the three roles provided. They and a third practitioner proposed an additional role: *promoting*. This role pushes CSE efforts forward, in particular in situations in which it does not appear reasonable from a time or cost perspective—but would pay off in the long run.

[1] http://ec.europa.eu/growth/smes/business-friendly-environment/sme-definition

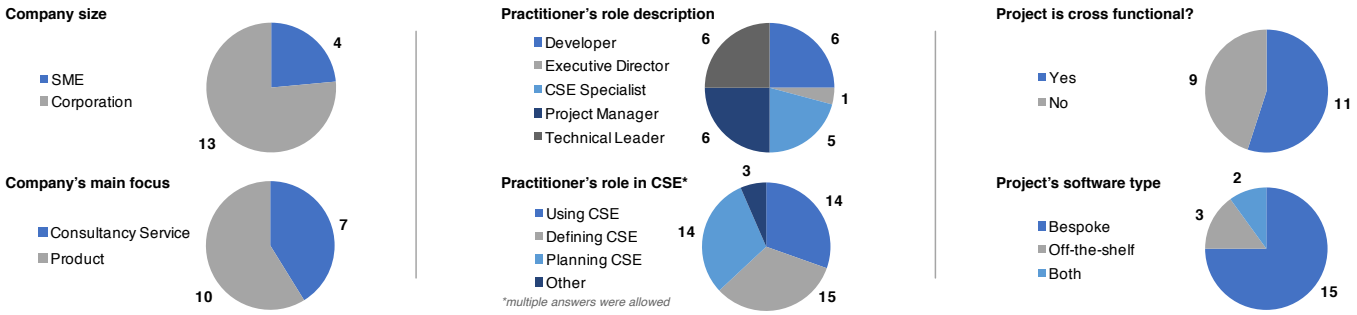

Figure 2: Descriptive data of the interview study: 24 practitioners from 17 companies were interviewed on 20 projects. In each interview, the practitioners related their answers to one particular project; four interviews were attended by two practitioners.

4.3 Projects

For each interview, we asked the practitioners to select one project that they are currently working on or which contains several CSE elements. On average, 20.25 employees work in a project, for SMEs 10.0 and for corporations 22.81. Eleven practitioners (64.71 %, including all SMEs) consider their project as cross functional, e. g., involving several other stakeholders from within the company, such as marketing professionals to represent the users. Notice that practitioners from all four SMEs stated a cross functional project structure. Three-quarters (15) of the projects develop *bespoke* software, e. g., custom software. Three projects (all by SMEs) develop commercial off-the-shelf software; two projects target both types. We asked the practitioners, if their projects and the way they have to be approached need to comply with any legal, security, medical, or environmental factors; every second practitioner confirms this. However, some practitioners referred to the rules for the product, rather than the project.

5 STUDY RESULTS

We illustrate the results of our quantitative analysis of CSE elements and categories mentioned in the interviews in Figures 3 and 4. They relate to the model in Section 6.1. The following subsections address the research questions based on both figures: At the beginning of each subsection, we answer a research question and then provide a more detailed analysis by stating several observations.

5.1 Practitioners' Definition of CSE

Although multiple CSE elements have been applied for decades, the term CSE only emerged in recent years. We asked the practitioners how they define CSE. We plot the individual results in Figure 3.

RQ1 — How do practitioners' define CSE? We found that the practitioners' definitions of CSE are mainly driven by CSE elements from the software management category, i. e., continuous integration of work (9), agile practices (8), and continuous deployment of releases (8). Further, CSE elements from the development and user category were mentioned repeatedly. Based on the responses, we identified five different perspectives on CSE that influence the definition of CSE; namely a *tool* (5.1.1), *methodology* (5.1.2), *developer* (5.1.3), *life cycle* (5.1.4), and *product management* (5.1.5) perspective.

Out of the 24 practitioners interviewed, slightly more than half (54 %) were using the term *CSE* as part of their active vocabulary. About two thirds (66 %) of all interviewees gave a definition of their understanding of CSE. Notably, 75 % of the interviewees in SMEs both gave a definition and actively used the term *CSE*. In general, these numbers support an uneven adoption of the term *CSE* among practitioners. For some practitioners, CSE is still ambiguous. They describe it as *fuzzy*, *abstract*, and *lacking a distinction*.

5.1.1 Tool Perspective. Practitioners, i. e., six developers and CSE specialists, make use of tool descriptions when defining CSE. Their descriptions are built on statements such as *in that regard, company A provides tool B*, or *after introducing tool C, we were able to accomplish element D*, or *we are currently looking into tool E of company F*. Four practitioners explicitly highlight that it is a well-chosen *tool chain* that enables CSE. In their opinion, the successful accomplishment of the steps availability, integration, and usage of tools allow a company to claim to be implementing CSE.

Observation 1 *Practitioners, in particular developers and CSE specialists, rely on a tool-driven approach for defining CSE. Commercially available tools influence their understanding of CSE.*

5.1.2 Methodology Perspective. In more than half of the interviews, practitioners cite a methodological perspective to define CSE. They emphasize a focus on short iterations and feedback.

Not mentioning specific tools, many practitioners highlight the importance of *how* the tools are applied. For instance, a sophisticated branching strategy should be preferred, instead of the exhaustive use of capabilities a version control system might offer. A state of *on-going iteration* should be reached, in which each commit leads to a finalized product. Different elements, such as continuous integration or agile practices, are applied to achieve a high level of automatization. Notably, some practitioners reflect on the combination of multiple CSE elements to achieve synergy effects. This implies that their perspective takes the impacts of CSE on other areas, such as the management of requirements, into account.

Observation 2 *Practitioners define CSE from a methodological perspective that aims for short iterations during software evolution. This perspective relies on well-defined steps in tool usage, workflows, and procedures. Every step is designed to enable a seamless workflow from a single commit until its finalization in form of a build.*

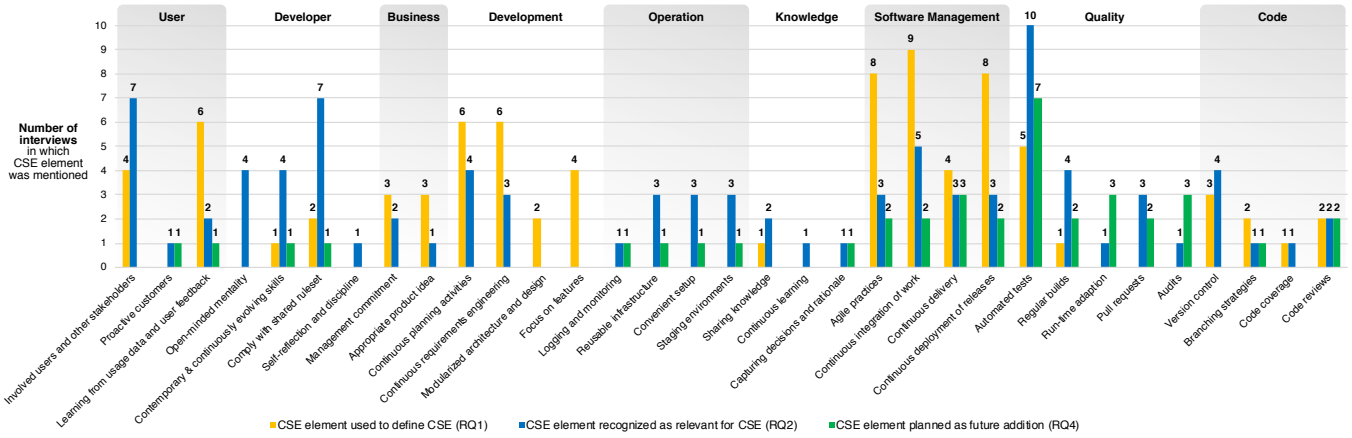

Figure 3: How often is a CSE element mentioned regarding a research question? Yellow bars indicate the number of interviews in which the respective CSE element was used for defining CSE (RQ1). Blue bars indicate practitioners' tendency towards relevant elements for CSE (RQ2). Green bars indicate CSE elements intended as future additions. Answers were summarized as one interview, in case multiple practitioners participated at the same time. The CSE elements are headed by their categories.

Several practitioners mention the importance of instant feature visibility to users. This enables constant retrieval of user feedback on the latest releases and more iterations with input from outside are performed. One practitioner advocates that every CSE process should be designed in accordance with the goal of matching customers' requirements through the implementation of short feedback loops. Another practitioner advises the collection of feedback *as often as necessary, rather than as often as possible.*

Observation 3 *One characteristic of CSE is the ability to make changes instantly visible to users. As a result, feedback from users can be elicited and used to match the software to requirements.*

Observations 2 and 3 are related and depend on each other to reach their full potential. However, we observed that not all practitioners implement both, leaving opportunities for improvement.

5.1.3 Developer Perspective. Several developers, project managers, and CSE specialists suggest a developer-driven perspective on CSE. Similar to the methodology perspective, practitioners did not base their descriptions on any specific set of tools.

First, they emphasize that CSE enables developers to fully focus on their main task, i. e., developing software, rather than deal with other processes, such as infrastructure management. This includes increasing the speed of the development process by removing idle times. Second, CSE allows practitioners to better estimate and classify their daily tasks. It ensures that newly introduced changes do not break code—an aspect which is not only in the interest of the overall product, but also a factor in the mind of developers. Providing a safe environment to develop and test software is a major characteristic of CSE. Third, according to practitioners, CSE supports the detachment of recurring, yet specific tasks from an individual; in particular, by introducing defined processes, knowledge vaporization can be prevented.

Observation 4 *CSE allows developers to fully concentrate on their respective tasks; it creates a safe environment for development. It enables the possibility for specific tasks to be removed from individuals.*

Remarkably, one practitioner highlights the increased responsibility of developers when giving their definition of CSE. This relates to the fact that developers independently create and deploy releases which—in order to unlock the full potential of CSE—should take place without any clearance or dedicated release plan.

5.1.4 Life Cycle Perspective. Various practitioners agreed that CSE opens up a new perspective on the software life cycle: development, deployment, and operational phases blend into each other. Practitioners report shorter intervals between the development and the production phases. They further state that CSE is characterized by the fact that a system's functionality is extended continuously and shaped by *continuous application life cycle management.*

Observation 5 *Practitioners characterize CSE by the blending of different phases of software engineering, such as development, deployment, and operation. According to their perception, this makes long-living systems easier to maintain.*

5.1.5 Product Management Perspective. Project managers, technical leaders, and executive directors formulate a definition of CSE from a product perspective. In their opinion, CSE is represented by constant funding, provided to continuously improve a product. This includes project managers' ability to continuously acquire new requirements as well as to create and re-prioritize product tasks.

One technical leader admits that not every product is guaranteed to follow this pattern. According to this practitioner, the application of CSE cannot simply be defined for a project: it is the product that determines whether CSE can be applied or not. Most of the projects are designed to follow other software evolution practices, and not CSE in particular. A product's compatibility with CSE processes needs to be ensured before applying CSE. Further, the environment in which the user receives the product plays an important role, as pointed out by a practitioner from a large corporation. It is required to keep pace with the CSE practices as defined in observation 2. For instance, if the deployment of a product requires certain manual steps, the product itself cannot be developed using CSE processes.

One practitioner defines CSE as the integration of customer, business model, software, and hardware. If a company successfully combines these four aspects, it is implicitly implementing CSE practices such as continuous integration and continuous delivery.

Observation 6 *Practitioners' definition of CSE is influenced by the product under development. Product-related factors such as funding, functionality, business model, and its future target environment need to match the continuous development capability.*

5.2 Practitioners' Relevant Elements of CSE

We asked practitioners' for CSE elements that *drive* CSE. Thus, they were required to list the three—in their opinion—most relevant CSE elements. In case a practitioner mentioned a CSE element that was not part of the CSE elements listed in Table 1, we recorded it as a relevant element. We plot the individual results in Figure 3.

RQ2 — Which CSE elements are perceived as most relevant by practitioners? Practitioners perceive CSE elements from three categories as most relevant: *quality*, i. e., automated tests (10), *user*, i. e., involved users and other stakeholders (7), and *developer*, i. e., comply with a shared ruleset (7). Besides, practitioners mention additional CSE elements: in particular the developers consider elements from the *code* category, such as version control, as obligatory, pivotal, and indispensable to any further steps in CSE. This strengthens the first stair in the *Stairway to Heaven* model of Bosch *et al.* [2]. We summarize the above-mentioned categories as follows: *user commitment* (5.2.1), *team commitment* (5.2.2), and *automated loop* (5.2.3).

5.2.1 User Commitment. In particular practitioners from SMEs that develop off-the-shelf software highlight contact with users as an CSE element. One technical leader points out a significant difference in the user audience: While there is a large number of users that are *passively* using software, i. e., users that do not state an interest in new additions that do not affect their typical workflows, it is the less represented *active* users who help to make the CSE feedback loop efficient and functional. One project manager continues this thought by stating that interaction with the users is barely technically defined in CSE. They approach this issue by actively trying to involve users through the promotion of nightly and beta builds. Two technical leaders claim that the success of a CSE project depends to a great extent on the degree of user involvement—*if there is no involved user, you lose.* Some practitioners remark that *users do not know what they want until they see it.* Enabled by continuous delivery, users can frequently provide feedback and thereby steer the development process towards their needs.

Observation 7 *Practitioners perceive users' commitment to take an active part in the development process as a relevant aspect of CSE.*

5.2.2 Team Commitment. Many practitioners perceive the team and its commitment as a highly relevant part of CSE. According to one developer, it is important that team members are open-minded towards the development process and take an active role in its formation. They need to adhere to a shared ruleset to work successfully. This poses challenges, as noted in Section 5.3. Practitioners illustrate a need for the full support of managers and executives.

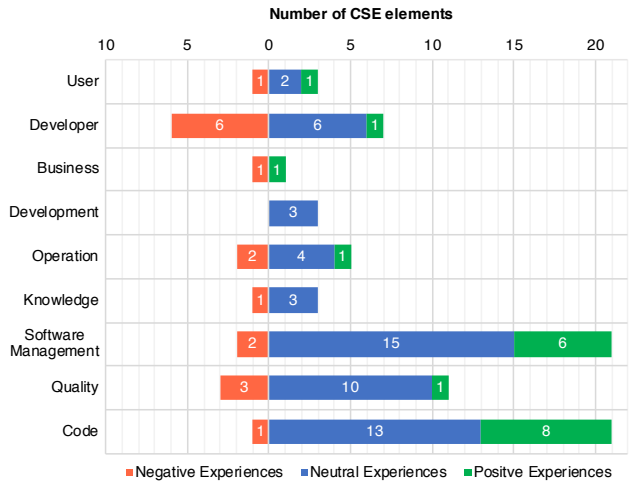

Figure 4: Practitioners' negative, neutral, and positive experiences with CSE elements (RQ3), grouped by category.

They should provide their attention to the project and trust the performance and skills of the team. Rather than tools, it is the methods and processes introduced by agile practices that bind team members together. A *continuous process improvement* activity should be carried out by the team on a regular basis.

Observation 8 *Practitioners perceive an open-minded team mentality that complies with a shared set of rules as the basis of successful CSE teams. Management commitment is indispensable, while agile practices serve as the main unifying factor.*

5.2.3 Automated Loop. Half of the practitioners declare the automatization of process loops to be the core of CSE. According to these individuals, discrete phases should be replaced by short, compact loops. They use the term *continuous pipeline* to describe a well-defined, highly automated process, which can be further adapted to the characteristics of the product under development. The practitioners share a vision of a non-linear process that can either be serialized or else run in parallel. *Automated tests* is the most relevant CSE element for ten practitioners. Others mention continuous integration and continuous deployment as major building blocks of an efficient automated loop comprising different fulfillment levels. Operational aspects, such as *staging environments*, complete their idea of an automated loop.

Observation 9 *Practitioners perceive a high maturity level of automatization as an essential aspect of CSE. This is enabled by well-defined steps that form a non-linear process model. Furthermore, practitioners state automated tests as the most relevant CSE element.*

5.3 Practitioners' Experience with CSE

We asked practitioners about positive, neutral, and negative experiences with CSE elements. Figure 4 shows the results grouped by their respective categories. Note that not every practitioner provided an experience report and, due to the grouping into categories, practitioners may be represented multiple times, if they responded to more than one CSE element of the same category.

RQ3 — What are practitioners' experiences with CSE? 19 positive, 56 neutral, 17 negative experiences with CSE elements were reported. Notably, more than 50 % of the positive experiences are stated by SMEs, while forming roughly a quarter of the interviewee sample. Categories with many positive experiences as in *code* and *software management* are an indicator for CSE elements that can serve as an entry point to CSE, since they may be easy to implement. Few positive mentions as is the case with *knowledge*, *business*, and *user* may be a sign of the low maturity of CSE elements. Neutral responses may indicate that practitioners are currently evaluating various CSE elements in the field. A large number of negative experiences as with the *developer* category indicates challenging CSE elements. We discuss distinct experience reports derived from five CSE categories: *developer* (5.3.1), *operation* (5.3.2), *software management* (5.3.3), *user* (5.3.4), and *quality* (5.3.5).

5.3.1 Developer. Most negative experiences were reported in the developer category, in particular for *complying with shared ruleset* and *contemporary and continuously evolving skills*. Negative experiences are amplified by problems with other CSE elements, such as *branching strategies*. One practitioner reports problems when dealing with too many branches, which they consider *poisonous to continuous integration*. However, they admit that it is sometimes inevitable to have several branches, though this situation can be approached with well-defined rules; for example, keeping the lifespan of a branch as short as possible, and committing code frequently. According to six practitioners, this demands attention from developers. Switching to a new way of developing software requires the willingness to evolve skills and extensive knowledge—which is why one practitioner views young graduates as having advantages over long-serving employees. One practitioner sketches solutions on how to overcome obstacles: providing incentives for successful work, enabling and supporting in-house training, as well as creating showcase projects. The practitioners agree that an open-minded mentality on the part of developers, as well as their ability to adapt and withstand the speed and frequency of CSE amount to both the basic requirement and also a major challenge for developers.

Observation 10 *Practitioners acknowledge a major challenge in developers' capability to comply with shared rulesets and in their open-minded mentality to continuously evolve their skills.*

Two practitioners note that it is the automatization of CSE that makes developers use methods which they would otherwise dispense with. However, they admit that the automatization makes it easier for developers to neglect other responsibilities. Consequently, they stress that CSE demands *self-reflection and discipline*. Practitioners with a leading role state that they trust their team members. They initiate discussions to find a consensus whenever necessary.

Observation 11 *From practitioners' experience, CSE does not solely build on developers' skills, but also on their ability to reflect on their work and on their sense of responsibility.*

5.3.2 Operation. Some large corporations with multiple departments struggle with legacy burdens, e. g., existing tool contracts that are not intended for CSE. Similarly, tools intended for CSE are used for other purposes, such as issue tracking systems for internal incident management, rather than for software development. Practitioners of other corporations emphasize the fact that tools are not a hurdle for them, since they can be easily bought—it is the integration that poses the challenges. Other corporations have to adhere to formal regulations that impede or prevent CSE from being applied in a given project, e. g., by relying on paperwork-driven processes. One practitioner states that there is a risk that agile projects will fall back into their previous static patterns.

Observation 12 *While practitioners are willing to apply CSE, it is their company's current set of tools that keeps them from making a complete transition and fully adapting CSE. Furthermore, requirements in regulated domains hinder the implementation of CSE.*

One practitioner complains that a major cost factor can be found in setting up the infrastructure for new projects in order to use CSE elements. Furthermore, given the internal hierarchical and management structure, some corporations do not have the capacity to respond rapidly to changes within the project.

Observation 13 *Practitioners state that the successful implementation of CSE requires the ability to set up a new project without major cost or time penalties.*

5.3.3 Software Management. In general, practitioners report positive experiences with the implementation of *agile practices*. Buildings-blocks such as sprints, review meetings, or SCRUM-Boards are well-received and provide high value. Some difficulties arise during task prioritization, since only limited resources—time and money—are available. Apart from that, CSE elements, such as continuous integration, are essential to practitioners. One practitioner mentions significant synergy effects when using tools of the same vendor for issue tracking, source code management, as well as continuous integration and delivery.

Observation 14 *Practitioners attest that CSE elements related to software management, such as agile practices or continuous integration of work, are widely and successfully adopted in their projects.*

5.3.4 User. CSE elements related to users are barely referenced whenever practitioners are asked about their experiences. The user's role is rated as *fuzzy* and there is a danger here that user feedback does not continuously flow back to developers. One practitioner ascribes this to the fact that CSE does not produce major releases that might be perceived as a notable change by the users. Ultimately, this creates a problem whereby user feedback is submitted late in the process in form of incidents or change requests. At that point, developers lack traceability links to changes that might have caused the feedback. One project manager is concerned that software quality suffers from the release frequency in the short run, and immature releases impede users' confidence in the product.

Observation 15 *Practitioners have not yet created processes that interact with users in a way similar to well-established practices such as continuous integration. This is mainly due to the fact that users' responses to ongoing changes are difficult to record, trace, and assess.*

5.3.5 Quality. Practitioners welcome CSE elements such as *pull requests* combined with *code reviews*. *Code coverage* and *audits* are practices that are gaining in importance. However, some practitioners raise concerns because quality metrics are not being tracked.

We observed that practitioners' responses regarding the *quality* category are driven by various testing and exploration reports. First, every project strives for high software quality and therefore tries to invest effort into improving. Second, as there is no final release, processes to improve software quality can always be developed further. Third, the influence of changes to software quality might become apparent only at a later time.

Observation 16 *Practitioners have had varying experiences with quality elements during CSE, but they still invest into improvements.*

5.4 Practitioners' Future Plans for CSE

We asked practitioners which CSE elements they are planning to add in the future. Thereby, we intend to discover future trends in CSE. We plot the individual results in Figure 3.

RQ4 — What are practitioners' future plans for CSE? Practitioners' plans are vague and mostly distributed across elements. 19 CSE elements either received only one, two, or three mentions by practitioners in the interviews. One CSE element stood out with seven mentions: *automated tests*. We found that the majority of practitioners described plans that span multiple CSE categories. We identified three main strategies in practitioners' answers: *enhancement* (5.4.1), *expansion* (5.4.2), and *on-demand adaption* (5.4.3).

5.4.1 Enhancement Strategy. Practitioners base their strategy for the future on a combination of the *methodology perspective* (observation 2) and *quality*, one of the most relevant categories mentioned (observation 9), yet one with mostly neutral experiences (observation 16). Seven practitioners mention automatization in the context of quality as one of their major plans for the short and long term. While *automated tests* are applied for some parts of the products, they should be made available for all. Two practitioners mention their plans of combining elements from the *operation* category, such as deployment in containers to enhance automatization. Three practitioners list activities to enhance their current state: code quality workshops, giving code metrics a meaning by calling for action rather than representing read-only information, and connecting CSE elements from different CSE categories.

One technical leader plans to bring the interaction with the user to the next level by detaching feedback collection from the individual—which is currently often the case—and creating a well-defined, high maturity level process similar to the one used for continuous integration. Other practitioners mention various ways of optimizing the implementation of agile practices or the application of branching strategies.

Observation 17 *Practitioners aim for a fully automated loop to increase the level of software quality by applying goal-driven enhancements to existing CSE elements.*

5.4.2 Expansion Strategy. The future plans of four practitioners can be summarized as an *expansion strategy*, i.e., applying recently established CSE elements to other areas of a project. The expansion of continuous delivery to more platforms is mentioned several times. This means adapting similar practices such as automatic deployment in mobile environments to their server-side counterparts. Similarly, expanding continuous integration to more platforms is mentioned several times; for example, one practitioner praises progress in *Java* environments, while they struggle with with *JavaScript*. Another practitioner mentions the expansion of documentation to more areas than it is the case at the present time.

Observation 18 *Practitioners aim to extend efficient CSE elements to other areas of the project or similar products.*

5.4.3 On-Demand Adaption Strategy. Three practitioners indicate a general interest in future CSE additions, however, they rely on an event-triggered or *on-demand* strategy to adapt, i.e., enhance or extend, their CSE elements. One practitioner describes an exploratory process in which CSE elements are added *step by step*. If they encounter a situation that would benefit from improvements, they initiate further investigations into possible solutions. Another practitioner makes the addition of further CSE elements dependent on the team's dynamic. A project manager highlights the effort in time that is required to implement certain CSE elements, making an overall process transition a time-consuming undertaking.

Observation 19 *Practitioners make enhancements and additions to CSE dependent on events that call for action. They postpone decisions for further additions to a later point in time.*

6 STUDY DISCUSSION

In this section, we summarize our insights in form of a model and discuss threats to validity regarding our research procedure. Above all, we find it necessary to state that some observations from this study appear obvious, ambiguous, or may even be seen as a tautology. However, since we are searching for empirical evidence on which to base assumptions that were previously only anecdotal, to some extent this leaves us with reporting *obvious conclusions* [22].

6.1 Eye of CSE

Based on practitioners' answers, we updated the CSE elements and categories in Table 1 by adding three new categories—*developer*, *business*, and *operation*—and removing two CSE elements, which are represented in the new categories. We refer to the final set of CSE elements and categories as the *Eye of CSE*, as shown in Figure 5.

The eye's focus lies on a comprehensive implementation of CSE, represented by the pupil. The process of reaching this goal is influenced by the categories that are part of the eye's iris. They define the diameter and size of the pupil and thereby the perception of CSE. We learned from the interviews that CSE categories are intertwined and have fuzzy boundaries. This is partly different to the sequential nature of the *Stairway to Heaven* by Bosch *et al.* [2]: even if some CSE elements, such as continuous integration and delivery, require a step-wise introduction, the practitioners' statements suggest that CSE should be approached from multiple angles simultaneously.

The *Eye of CSE* can serve as a checklist for practitioners to tackle the subject of CSE by incrementally applying CSE elements and keeping an eye on potential next steps. The grouping of CSE categories allows practitioners to recognize relationships based on their proximity within the eye and the position of the CSE elements.

Example Whenever practitioners expand on *software management*, the categories *knowledge* and *quality* should be incorporated.

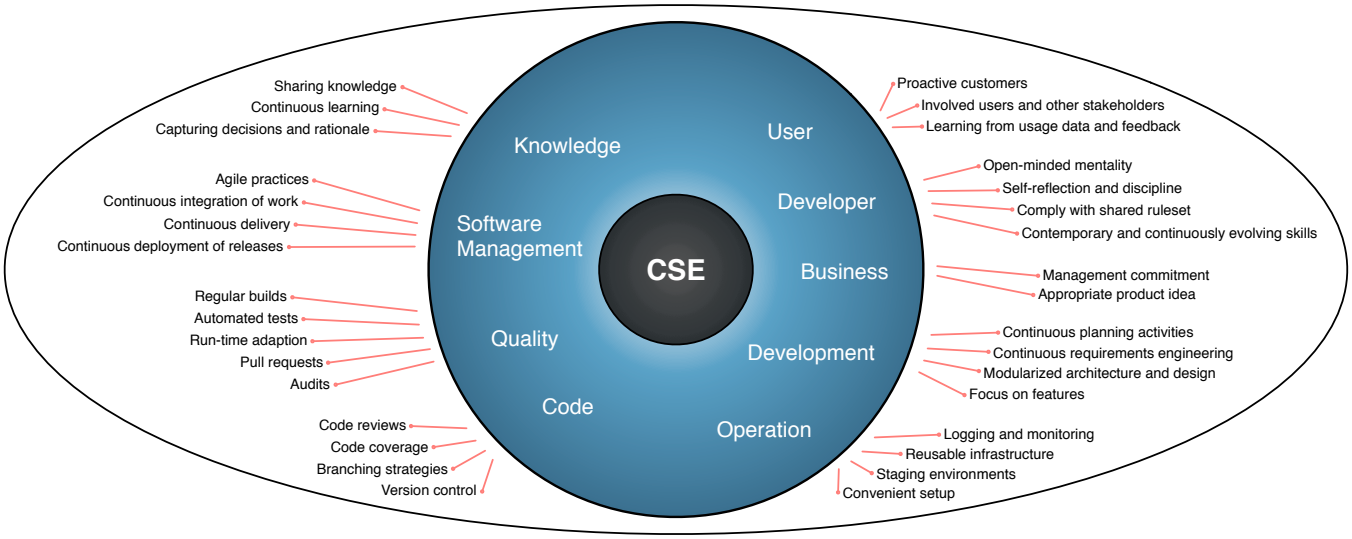

Figure 5: The model *Eye of CSE* consists of nine CSE categories and 33 CSE elements. The proximity of elements and categories suggests relationships between them. The model can open up ones' eyes to new ideas for additions to current CSE processes.

During the design of the *Eye of CSE*, we strived to accurately allocate CSE elements to categories by analyzing the practitioners' answers and carrying out internal discussions. By connecting CSE elements to the iris and not directly to CSE categories, we acknowledge that one CSE element can relate to one or more categories. The proximity of CSE elements within the *sclera*, the white of the eye, suggests a relation to a category and among CSE elements.

Example *Audits* can include *code reviews*, which makes them part of the *quality* category. Likewise, *learning from usage data and feedback* requires developers' *open-minded mentality*.

The model should open practitioners' eyes to new ideas when extending their CSE process. Furthermore, the relationships can start discussions for future consolidation of the model: The categories *user*, *developer*, and *business* could be further summarized as *stakeholders*, while the categories *business*, *development*, and *operation* share common characteristics and might be combined as *BizDevOps*, following the naming conventions from Fitzgerald and Stol [4]. As highlighted, we see structural dependencies between *software management* and *knowledge* and between *quality* and *code*.

6.2 Threats to Validity

In the following, we discuss the study's threats to validity according to the four aspects of validity given in Runeson *et al.* [17].

Construct validity concerns the disparity between the intended and actual study observations [17]. First, the practitioners resemble a heterogeneous group, each having an own point of view on CSE. The conformance between them might be small. We tried to address heterogeneity by describing observations instead of facts. Second, the questions may be interpreted by practitioners in a way different than intended by us. We tried to minimize this possibility by conducting two interviews with colleagues. We discussed these interviews afterwards to reveal potential misinterpretations. Also, the format of the interviews allowed practitioners to ask questions at any time. Third, the authors might have influenced the participants by asking specific questions. To mitigate this risk, we used open-ended questions to elicit as much information as possible from practitioners. Finally, the collection of CSE elements is based on a model that is an abstraction of reality and—to some extent—subjective. We asked practitioners to describe their experiences with the proposed set of CSE elements, which might biased them. We tried to mitigate this risk by collecting additional CSE elements.

Internal validity concerns correlations between the investigated factors and other factors [17]. Practitioners might have provided answers that do not fully reflect their daily work, since they were aware that results would be published. We addressed this possibility by guaranteeing the full anonymity of interviewees and companies. Further, the interpretation of answers might be biased by the authors' *a priori* expectations and subconscious impressions. We addressed this threat by coding the transcriptions and discussing the codes. Finally, the slides might biased the practitioners' perception of CSE. We perceive this as a minor threat, since it can only affect RQ1 and practitioners might prepare beforehand anyway.

External validity addresses the generalizability of the study results [17]. We contacted companies that we already knew, which affects the sampling and might result in a selection bias. However, there is no central register of companies that apply CSE [22]. Clearly, this amounts to a risk to the representativeness of the participants. It is mitigated by the fact that the authors are from two different universities. Further, the diversity of projects and participants reinforces the generalizability. Finally, interviews are subjective, since they rely on the practitioners' statements. To reduce subjectivity, we conducted 20 interviews, to acquire a wider set of opinions.

Reliability validity concerns the study's dependency on specific researchers [17]. After we carried out coding training and checked intercoder reliability, two authors individually coded different transcripts. We address this threat by discussing questions during coding; a third author of this paper supervised the interview analysis.

7 RELATED WORK

To the best of our knowledge, there are no previous studies with practitioners that address CSE as a process. Thus, we present and discuss studies that followed research approaches similar to our work and addressed specific CSE elements and CSE categories. Kuhrmann *et al.* research development approaches in practice [10]. They highlight the importance of traditional frameworks, a factor that is supported by our observation 12. Their results indicate that companies apply hybrid approaches, which are defined stepwise, in ways similar to the strategies we identified in Section 5.4. Mäkinen *et al.* report the widespread adoption of version control and continuous integration, based on semi-structured interviews with 18 organizations [12]. Our observations confirm this situation in practice. In fact, we found that most positive experience reports were stated in the respective CSE categories (Figure 4). Ståhl and Bosch interviewed practitioners to assess their experience of continuous integration and discuss benefits [20]. We can confirm most of their findings, e. g., that it increases developer productivity (observation 4). The same observation partially supports their finding that practitioners see improvements in project predictability. Further literature studies list benefits and challenges regarding continuous integration, delivery, and deployment [16, 19]. Kevic *et al.* reveal the positive impact of experiments; this points towards a relationship between continuous deployment and a change in user behavior [8], and supports our model (Figure 5). Dybå and Dingsøyr highlight human and social factors in working settings that are similar to CSE [3]. Ayed *et al.* conclude that the success of agile practices depends on various social and inter-cultural factors [1]. Larusdottir *et al.* note the importance of teams' ability to trust in their capabilities [11]. Based on these reports and multiple answers in our study, we added the category *developer*, as described in Section 6.

8 CONCLUSION

We describe the method and observations of a semi-structured interview study involving 24 practitioners from 17 companies during 20 interviews. The study provides an overview of the current state of practice in CSE, as a way of assisting practitioners to understand CSE. We found that practitioners have different perspectives, i. e., *tool*, *methodology*, *developer*, *life cycle*, and *product management* perspectives. Most relevant elements of CSE are *user* and *team commitment*, as well as a high degree of maturity of *automated loops*. With respect to CSE elements, practitioners report more positive experiences than negative ones, but more than half of the responses were neutral. This might indicate that many practitioners are currently only testing various CSE element in the field. For the future, practitioners focus on three strategies: *enhancement*, *expansion*, and *on-demand adaption*. To support practitioners in their approaches to establish CSE in companies, we created a model—the *Eye of CSE*—based on our observations during the interviews. The *Eye of CSE* allows practitioners to identify relevant CSE elements, the mutual relationships between them, and to devise future additions to their own projects. We do not claim that the model is complete. We see a continuation of the study in form of a survey to further validate the perception of the *Eye of CSE* to reach for a more quantitative and comprehensive understanding. Insights from practitioners from different domains might open up new research directions.

ACKNOWLEDGMENTS

This work was supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future – Managed Software Evolution (CURES project). We thank the practitioners for their participation in the interviews and sharing their insights, as well as Rana Alkadhi for her valuable feedback.

REFERENCES

[1] Hajer Ayed, Benoit Vanderose, and Naji Habra. 2017. Agile cultural challenges in Europe and Asia: insights from practitioners. In *Proc. of the 39th Int. Conf. on Software Eng.: SEIP Track*. IEEE, 153–162.
[2] Jan Bosch. 2014. *Continuous Software Engineering: An Introduction.* Springer.
[3] Tore Dybå and Torgeir Dingsøyr. 2008. Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol.* 50, 9-10 (2008), 833–859.
[4] Brian Fitzgerald and Klaas-Jan Stol. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123 (2017), 176–189.
[5] Watts S. Humphrey. 1988. The Software Engineering Process: Definition and Scope. *SIGSOFT Softw. Eng. Notes* 14, 4 (April 1988), 82–83.
[6] Jan Ole Johanssen, Anja Kleebaum, Bernd Bruegge, and Barbara Paech. 2017. Towards a Systematic Approach to Integrate Usage and Decision Knowledge in Continuous Software Engineering. In *2nd Workshop on Continuous Software Engineering*. 7–11.
[7] Jan Ole Johanssen, Anja Kleebaum, Bernd Bruegge, and Barbara Paech. 2017. Towards the Visualization of Usage and Decision Knowledge in Continuous Software Engineering. In *2017 IEEE Working Conference on Software Visualization (VISSOFT)*. 104–108.
[8] Katja Kevic, Brendan Murphy, Laurie Williams, and Jennifer Beckmann. 2017. Characterizing Experimentation in Continuous Deployment: A Case Study on Bing. In *Proc. of the 39th Int. Conf. on Software Eng.: SEIP Track*. 123–132.
[9] Stephan Krusche and Bernd Bruegge. 2017. CSEPM - A Continuous Software Engineering Process Metamodel. In *2017 IEEE/ACM 3rd International Workshop on Rapid Continuous Software Engineering (RCoSE)*. 2–8.
[10] Marco Kuhrmann, Philipp Diebold, Jürgen Münch, Paolo Tell, Vahid Garousi, Michael Felderer, Kitija Trektere, Fergal McCaffery, Oliver Linssen, Eckhart Hanser, and Christian R. Prause. 2017. Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond. In *Proceedings of the 2017 International Conference on Software and System Process (ICSSP 2017)*. ACM, 30–39.
[11] Marta Larusdottir, Jan Gulliksen, and Åsa Cajander. 2017. A license to kill – Improving UCSD in Agile development. *J. Syst. Softw.* 123 (2017), 214–222.
[12] Simo Mäkinen, Marko Leppänen, Terhi Kilamo, Anna-Liisa Mattila, Eero Laukkanen, Max Pagels, and Tomi Männistö. 2016. Improving the delivery cycle: A multiple-case study of the toolchains in Finnish software intensive enterprises. *Information and Software Technology* 80 (2016), 175 – 194.
[13] Michael D Myers and Michael Newman. 2007. The Qualitative Interview in IS Research: Examining the Craft. *Information and Organization* 17, 1 (2007), 2–26.
[14] Helena Holmström Olsson, Hiva Alahyari, and Jan Bosch. 2012. Climbing the "Stairway to Heaven" – A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In *2012 38th Euromicro Conf. on Soft. Eng. and Advanced Applications*. 392–399.
[15] Akond Ashfaque Ur Rahman, Eric Helms, Laurie Williams, and Chris Parnin. 2015. Synthesizing Continuous Deployment Practices Used in Software Development. In *2015 Agile Conference*. 1–10.
[16] Pilar Rodríguez, Alireza Haghighatkhah, Lucy Ellen Lwakatare, Susanna Teppola, Tanja Suomalainen, Juho Eskeli, Teemu Karvonen, Pasi Kuvaja, June M. Verner, and Markku Oivo. 2017. Continuous deployment of software intensive products and services: A systematic mapping study. *J. Syst. Softw.* 123 (2017).
[17] Per Runeson, Martin Host, Austen Rainer, and Björn Regnell. 2012. *Case Study Research in Software Engineering: Guidelines and Examples.* John Wiley & Sons.
[18] Johnny Saldaña. 2009. *The Coding Manual for Qualitative Researchers* (2 ed.). SAGE Publications.
[19] Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. 2017. Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access* 5, Ci (2017), 3909–3943.
[20] Daniel Ståhl and Jan Bosch. 2013. Experienced Benefits of Continuous Integration in Industry Software Product Development: A Case Study. In *Proceedings of the 12th IASTED International Conference on Software Engineering, SE 2013*. 736–743.
[21] Daniel Ståhl, Torvald Mårtensson, and Jan Bosch. 2017. The continuity of continuous integration: Correlations and consequences. *Journal of Systems and Software* 127 (2017), 150–167.
[22] Marco Torchiano and Filippo Ricca. 2013. Six reasons for rejecting an industrial survey paper. In *2013 1st International Workshop on Conducting Empirical Studies in Industry (CESI)*. 21–26.