How do Practitioners Capture and Utilize User Feedback during Continuous Software Engineering?

Jan Ole Johanssen*, Anja Kleebaum†, Bernd Bruegge*, and Barbara Paech†

*Technical University of Munich, Department of Informatics,
Garching bei München, Germany, {jan.johanssen, bruegge}@in.tum.de

†Heidelberg University, Institute of Computer Science,
Heidelberg, Germany, {kleebaum, paech}@informatik.uni-heidelberg.de

***Abstract*—Continuous software engineering (CSE) evolved as a process for rapid software evolution. Continuous delivery enables developers to frequently retrieve user feedback on the latest software increment. Developers use these insights for requirements validation and verification. Despite the importance of users, reports about user feedback in CSE practice are sparse. We conducted 20 interviews with practitioners from 17 companies that apply CSE. We asked practitioners how they capture and utilize user feedback. In this paper, we detail the practitioners' answers by posing three research questions. To improve continuous user feedback capture and utilization with respect to requirements engineering, we derived five recommendations: First, internal sources should be approached, as they provide a rich source of user feedback; second, existing tool support should be adapted and extended to automate user feedback processing; third, a concept of reference points should be established to relate user feedback to requirements; fourth, the utilization of user feedback for requirements validation should be increased; and last, the interaction with user feedback should be enabled and supported by increasing developer–user communication. We conclude that a continuous user understanding activity can improve requirements engineering by contributing to both the completeness and correctness of requirements.**

***Index Terms*—user feedback, usage monitoring, usage data, continuous software engineering, interview study.**

I. Introduction

The rapid development of software known as continuous software engineering (CSE) [1], [2] has become an established process for software evolution. CSE has enabled new capabilities to capture insights, for instance regarding decisions [3] or user involvement: Continuous delivery promotes user feedback on the latest changes to a software increment [4], [5]. While user feedback can be collected in an explicit form, such as written reviews, the monitoring of implicit user feedback improves understanding the need for new requirements [6]. CSE forms the basis for continuous experimentation, e.g., selecting the appropriate user audience for a feature, which opens new possibilities for requirements engineering [7].

Reports on current practices of how user feedback is captured and utilized in industry are sparse. While many practices, such as continuous integration [8], [9], continuous delivery [10], [11], or their combination [12], [13] have been extensively researched, the interaction with users in CSE environments is less understood. Rodríguez *et al*. identified a research gap for mechanisms that make use of feedback [14].

We conducted a semi-structured study by interviewing 24 practitioners at 17 companies during 20 interviews over the course of six months in 2017. Using their answers, we already investigated the state-of-the-practice of CSE [15] and evaluated our vision of improved integration of multiple types of knowledge into CSE [16]. To this end, we specifically asked practitioners about their use of decision knowledge during CSE [17] and their practices regarding user feedback. The results for the latter are presented in this paper.

We designed three research questions (RQs) to investigate user feedback in CSE. First, *which user feedback do practitioners consider for CSE?* This RQ investigates the types of user feedback on which the practitioners rely, as well as to which artifacts they relate the user feedback. The answers form the basis for the following RQs, as both of them affect how user feedback is captured and utilized. Second, *how do practitioners capture user feedback in CSE?* This RQ investigates how often practitioners capture user feedback, which tools they use, the sources on which they rely, and whether they capture the context of user feedback. Third, *how do practitioners utilize user feedback in CSE?* This RQ investigates the reasons why practitioners capture user feedback, whether they exploit changes over time, and whether they combine different user feedback types.

Based on the practitioners' answers to the RQs, we derived five recommendations that can guide improvement of continuous user feedback capture and utilization. We propose to establish a *continuous user understanding* activity that encompasses practices and tool support to gain insights from user feedback for requirements engineering.

The remainder of this paper is structured as follows. Section II provides an overview of RQs and their respective coding. Section III summarizes methodological aspects, i.e., the coding procedure and threats to validity of the interview study. Section IV provides brief descriptive data about the companies, the practitioners, and the projects. Section V details the practitioners' answers using quantitative analyses and qualitative explanations. Section VI discusses the results by combining results among RQs, summarizing observations, and deriving recommendations for requirements engineering. Section VII presents similar studies that address user feedback. Section VIII presents our conclusions.

Definitive version available at https://doi.org/10.1109/RE.2019.00026

TABLE I
OVERVIEW OF THE MAIN AND SUB-RESEARCH QUESTIONS FOLLOWED BY THEIR CODING AND SECTION THAT COVERS RESPECTIVE RESULTS.

Research Questions		Coding	Section
RQ 1	**Which user feedback do practitioners consider for CSE?**		
RQ 1.1	What types of user feedback do practitioners collect?	Explicit, Implicit	V-A1
RQ 1.2	Which artifacts do practitioners relate user feedback to?	Application, Feature	V-A2
RQ 2	**How do practitioners *capture* user feedback in CSE?**		
RQ 2.1	How often do practitioners capture user feedback?	Event-based, Periodic, Continuous	V-B1
RQ 2.2	Do practitioners rely on tool support to capture user feedback?	Tool support, Manual capture	V-B2
RQ 2.3	What are practitioners' sources for user feedback?	Internal, External	V-B3
RQ 2.4	Do practitioners capture the context of user feedback?	Capture context, Omit context	V-B4
RQ 3	**How do practitioners *utilize* user feedback in CSE?**		
RQ 3.1	Why do practitioners capture user feedback?	Planning, Support, Improvement	V-C1
RQ 3.2	Do practitioners exploit the change of user feedback over time?	Exploitation of change over time, No exploitation	V-C2
RQ 3.3	Do practitioners combine different user feedback types?	Pragmatic combination, No combination	V-C3

II. RESEARCH QUESTIONS

Table I lists RQs and respective sub-RQs. The coding was developed during data analysis. Answers to some sub-RQs qualify for more than one coding. In the following, we summarize the motivation and goals of the sub-RQs.

RQ 1.1 asks about the types of user feedback considered by practitioners. User feedback can be explicit or implicit, as described by Maalej *et al.* [6]: Explicit user feedback refers to any feedback the user is actively providing, such as an app review. Implicit user feedback is produced by users interacting with the artifact, such as a click on a button.

RQ 1.2 asks for artifacts to which practitioners relate user feedback. This can be either to the entire application or to a specific feature. The difference in this granularity indicates the maturity of the user feedback capture process, because feature-specific user feedback requires more processing efforts.

RQ 2.1 asks for the frequency with which practitioners capture user feedback. As the study investigates user feedback within the context of CSE, the results support an understanding whether practitioners also applying a continuous approach for the capture of user feedback during CSE.

RQ 2.2 asks about the tools used by practitioners. The capture of user feedback results in large amounts of unstructured data that needs to be processed to derive useful insights. This can be done either manually or with tool support.

RQ 2.3 asks about practitioners' sources of user feedback, i.e., from whom they capture user feedback. While we assume that practitioners rely on user- or developer-triggered explicit feedback or implicit feedback from monitored usage data, the intend of this question is to reveal more about the feedback provider, which may be a group of individuals.

RQ 2.4 asks about the capture of the context for the user feedback. In the human–computer–interaction domain, context is defined as any kind of information that is used to characterize an entity [18], [19]. For user feedback, this may be the time or location of the user when they are using the application.

RQ 3.1 asks about practitioners' reasons for the capture of user feedback. We are interested why they capture the user feedback and what it is used for with respect to the artifact.

RQ 3.2 asks about the exploitation of changes in the user feedback over time. Investigating the evolution of user feedback can support the practitioners in revealing trends in how feature increments are applied and perceived by users.

RQ 3.3 asks about the combination of two or more types of user feedback to derive new insights. An example may be a written review, i.e., explicit feedback, that is augmented with a usage log, i.e., implicit feedback, from the same user.

III. RESEARCH METHOD

We outline the interview study with practitioners and discuss its threats to validity. More details are provided in [15], [16].

A. Interview Study

We conducted a semi-structured interview study [20], [21], as a means to study phenomena, actors, and their behavior in their natural context [22]. We structured the study into the phases design and planning, data collection, and data analysis.

Definitive version available at https://doi.org/10.1109/RE.2019.00026

1) Design and Planning: We created a questionnaire to acquire descriptive data (see Section IV) and interview questions that are similar to the sub-RQs presented in Table I. For reproduction of this study, we advise to use the sub-RQs as interview questions to maintain comparability with our results. We also compiled a list of companies that apply CSE.

2) Data Collection: Between April and September 2017, we conducted 20 interviews. In four interviews, two practitioners participated. We recorded the interviews with the practitioners' permission and transcribed the audio recordings. Transcripts were reviewed by the practitioners to ensure correctness. We guaranteed anonymity and aggregated results in case individual characteristics would otherwise have stood out.

3) Data Analysis: We analyzed the transcripts with the help of qualitative data analysis software [23]. In a first stage, we allocated practitioners' answers to RQs. This was necessary if a practitioner returned to a previous question after they provided an answer. In a second stage, we applied the coding listed in Table I. We derived the coding from related work on the capture and utilization of user feedback as well as emerging topics that we identified while analyzing answers. We analyzed the results quantitatively and qualitatively. We consolidated practitioners' responses as single answer in case an interview was attended by two practitioners.

B. Threats to Validity

We strived to combine and summarize answers from multiple practitioners on how they capture and utilize user feedback. As a result, our interview study may have several limitations [22]. Therefore, we discuss the four criteria for validity as they apply to empirical research in which researchers follow a positivist stance [21], [24].

1) Construct Validity: This criterion concerns the question of whether theoretical constructs were measured and interpreted correctly. The practitioners might have had a different understanding of the questions than what we had intended. We tried to minimize this aspect by encouraging the practitioners to ask questions at all time. Moreover, we conducted two interviews with experienced colleagues and discussed these interviews afterwards to reveal potential misinterpretations. We used open-ended questions to allow practitioners to provide extensive answers, which increases the amount of detail.

2) Internal Validity: This criterion concerns the conclusions that were drawn actually follow from the data, e.g., whether there are confounding factors that influenced the results. The practitioners might have answered questions in a way that does not reflect their daily practices, because they knew that the results would be published. We tried to mediate this possibility by ensuring the anonymity of interviewees and companies at any time. The practitioners' roles and the associated projects are context factors that influence their answers. We tried to address heterogeneity by describing observations instead of facts and deriving recommendations from them. The analysis of practitioners' answers might have been affected by the authors' *a priori* expectations and impressions. We addressed this threat by jointly discussing the coding of the transcripts.

3) External Validity: This criterion concerns the generalizability of the study. We contacted companies that we knew from other projects as there exists no central register of companies that apply CSE [25]. This might have resulted in a sampling bias in a way that the practitioners represent only a subgroup of the actual target population. We reduced this threat by relying on the industrial contacts of authors from two different universities. The interviews reflected subjective impressions, as they rely on practitioners' statements. We conducted 20 interviews to retrieve a broad spectrum of opinions. The number of answers varied for most of the interview questions; we indicate this in the figure captions. In some interviews, we skipped interview questions, either due to practitioners' time constraints or because they provided answers that made it clear that they cannot answer a subsequent question. However, we think that the diversity of projects and participants supports the generalizability of the answers.

4) Reliability Validity: This criterion concerns the study's dependency on specific researchers. We performed coding training and assessed the intercoder reliability. Then, two authors individually coded the transcripts, which might have affected the results. We mediated this threat by discussing questions during coding, especially in case of ambiguity. Also, the interview coding was supervised by a third author.

IV. Descriptive Data

We compiled descriptive data regarding the companies, practitioners, and projects. The following summarizes the information; more details and graphical data representations are provided in [15], [16]. We interviewed 24 individual practitioners during 20 interviews. The practitioners are affiliated with 17 companies. One company was interviewed twice; another one three times. We aimed for a high diversity of interviews by approaching companies of different size, practitioners with different backgrounds, and projects in different domains.

We categorized four companies that have a maximum staff headcount of 250 as small and medium-sized enterprises (SME). We refer to the remaining companies as corporations; eight employ up to 2,000, two around 50,000, and three 100,000 or more employees. Seven of the companies offer consultancy services. The remaining companies develop software products for consumer and business markets.

We used the practitioners' role descriptions to group them: five CSE specialists, e.g., a continuous deployment manager or a DevOps engineer, six developers, six project managers, six technical leaders, and one executive director. On average, the practitioners have spent 2 years in their respective roles, have 10 years' experience in information-technology (IT) projects, and have participated in 19 IT projects.

We asked the practitioners to relate their answers to a particular CSE project. On average, 20 employees are involved in such a project, with an average of 10 in projects at SMEs and 23 in corporations. All SME practitioners stated that their projects are cross-functional. Of the 20 practitioners, 15 reported to solely develop custom software, 3 focus on off-the-shelf products, and 2 develop both software types.

Definitive version available at https://doi.org/10.1109/RE.2019.00026

V. Results

For each sub-RQ, we present the practitioners' answers as a bar chart and detail them in an explanation. All charts extend to 20 interviews to enable direct visual comparisons among them. If an answer for a sub-RQ received two or more codings, we visualize the count as a grey bar and do not reduce it from the individual coding to maintain their expressiveness. In such cases, we add the number of answers that were exclusively (*excl.*) applied to a coding inside their respective bar.

A. Which User Feedback do Practitioners Consider for CSE?

This RQ investigates which user feedback types practitioners rely on for the capture and utilization as well as to which artifacts they relate the user feedback.

1) Types of User Feedback: Figure 1 visualizes the practitioners' answers.

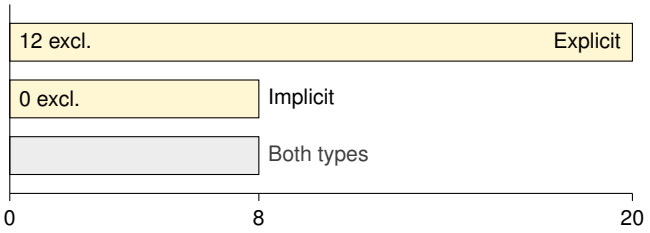

Fig. 1. Number of interviews in which practitioners addressed user feedback types. No practitioner reported to rely solely on implicit user feedback.

All 20 practitioners reported that they rely on *explicit* user feedback. Four practitioners did not further specify the kind of feedback. Five practitioners detailed how the explicit user feedback is produced, which allows conclusions about its kind. They described personal meetings in the form of presentations or design thinking workshops, in which they showed the user the latest version of the application. They collected either written or verbal feedback, using questionnaires or manually noting down how the users interact with the application. Three practitioners relied on explicit user feedback that is provided in the form of issue tickets. In some cases, the issue management system is publicly available. Then, the users themselves can create feature requests and bug reports that are directly passed to the developers. In other cases, the issues are reported by an intermediate layer such as first-level support.

None of the practitioners stated that they rely solely on *implicit* feedback; however, eight practitioners described the use of both types, partly depending on when and from whom they capture the user feedback. The practitioners start with the capture of explicit user feedback at an early stage. This may include video recordings, surveys, user interaction with prototypes in a controlled environment, i.e., applying observation techniques, as well as email and textual feedback that can be enriched with screenshots to receive qualitative, i.e., explicit, user feedback. After reaching a certain maturity level, they transition to collecting implicit user feedback and apply quantitative methods: the practitioners have mentioned A/B tests and the collection of usage statistics and analytics data.

One practitioner mentioned that they enrich the frequent implicit feedback by acquiring explicit user feedback through channels such as service hotlines or events such as exhibitions. Another practitioner mentioned primarily focusing on implicit user feedback, which allows them to gather information on how and when a user interacted with the application.

2) Artifacts for User Feedback Relationship: Figure 2 visualizes the practitioners' answers.

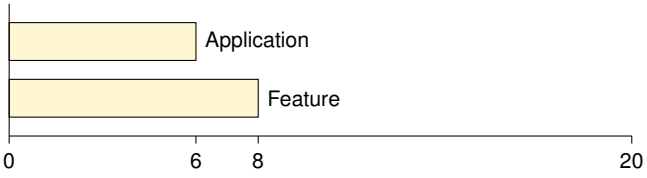

Fig. 2. Number of interviews in which practitioners addressed artifacts to which they relate user feedback. In six interviews, no answers were provided.

Six practitioners stated that they relate user feedback to the *application* itself, which serves as a high-level reference point. Two practitioners relate user feedback to operational or technical aspects of the application; they are only interested in the outcomes, i.e., what the application was used for. Two other practitioners stated that individual features are not of interest to them, that they instead are interested in the overall acceptance of the application. For individual feature aspects, they stated that there are always formal specifications that they fulfill and guarantee, such as by using test cases Therefore, they do not require additional verification through user feedback. This perspective is continued by one practitioner who emphasized that they are interested in finding situations in which users behave differently from what was expected—which could occur at any time while using the application. One other practitioner follows a similar approach, but is driven by the question of how the overall application could be improved.

Eight practitioners stated that they collect user feedback with a focus on the *feature* that is currently under development. Following an example by the practitioners, that may be a user feedback on how a new menu navigation is adapted by users. Five of them detailed how they establish the relationship between user feedback and the feature under development: One practitioner pointed out that this can be achieved by explicitly asking for feedback in a survey that allows users to describe the feature, which requires them to verbally relate their feedback. Another approach is having the user add reference points that guide developers to the feature in question; for example by attaching screenshots to the provided feedback. The practitioners explained that without such guidance, reproduction of the feedback and its understanding are difficult. To relate implicit user feedback to a feature, one practitioner reported that they create dedicated profiles for individual users to map a change in the feedback to the increment. One practitioner noted that they consider feature-specific user feedback, but only in an isolated surrounding which is meant to qualitatively improve a particular feature, rather than comparing it with an alternative implementation.

Definitive version available at https://doi.org/10.1109/RE.2019.00026

Another practitioner explained that their features correlate with services their company provides. Based on this relationship, they conclude that—if users activated a service—they also made use of a particular feature. This enables a high-level assessment if and how many users interact with a feature.

B. How do Practitioners Capture User Feedback in CSE?

This RQ investigates the frequency of user feedback capture, practitioners' use of tool support, feedback sources, and whether they capture the context of user feedback.

1) Frequency of User Feedback Capture: Figure 3 visualizes the practitioners' answers.

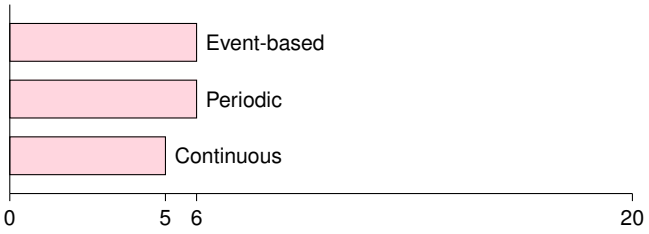

Fig. 3. Number of interviews in which practitioners addressed how often they capture user feedback. In three interviews, no answers were provided.

Six practitioners reported to capture user feedback *event-based*, which is reflected in a sporadic or unscheduled process. Three practitioners stated to capture the feedback ad-hoc or on-demand. This applies when they initiate the capture process on their own in case they require more insights. Likewise, two other practitioners mentioned making the capture process dependent on when the user provides explicit feedback after using the application. One practitioner described events that relate to a well-defined point in time, such as the beginning or end of the application development. This could also include the capture of user feedback that is triggered by external changes, such as updates of the operating system of the device running the application. Another practitioner highlighted that the capture of user feedback comes down to individual, rare, and irregular personal meetings.

Six practitioners collect user feedback in a *periodic* manner. They mentioned periods with a minimum length of two days; to one, two, and three weeks; up to a month. One practitioner criticized the testers always lagging behind the latest development, which leads to a time delay between feature development and the capture of user feedback. For this reason, the practitioner referred to the capture process as semi-agile. Another practitioner reported pilot phases that are dedicated to the capture and understanding of user feedback.

Five practitioners stated that they *continuously* capture user feedback. Only one of them continuously collects explicit user feedback, in the form of frequent meetings, phone calls, and video conferences with the users. All of the other practitioners rely on implicit user feedback during their continuous capture process. Notably, the five practitioners spoke with one voice in concluding that—while the user feedback is continuously captured—this does not imply that it is assessed or utilized; nor is it used to augment decisions or derive new insights.

2) Tool Support for User Feedback Capture: Figure 4 visualizes the practitioners' answers.

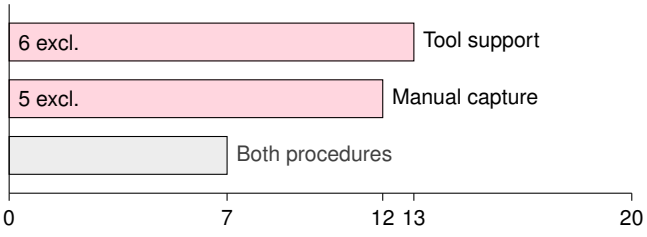

Fig. 4. Number of interviews in which practitioners addressed tool support to capture user feedback. In two interviews, no answers were provided.

We define tool support as a system designed to capture user feedback in the first place. In seven interviews, practitioners stated to use both tool support and manual capture approaches.

Overall, 13 practitioners rely on *tool support* to capture user feedback. All of them reported using standard tools; they listed various tools that are known for their analytics and experience analysis capability, of which are prominent products from the following companies: Google (4 mentions), Adobe (3 mentions), and Microsoft (2 mentions). Four other practitioners listed tools such as Atlassian JIRA or Redmine to capture explicit user feedback reports. On the one hand, it enables them to directly utilize the feedback for prioritization of requirements and development tasks. On the other hand, it helps them in finding major problems at times of peak activity. One practitioner stated that, while the possibility to use a particular tool exists, it is not being used. Besides standard software, 5 of the 13 practitioners additionally rely on custom software, i.e., tools that they developed for the sole purpose of capturing user feedback while their particular application is in use. This enables them to track individual features, as well as other aspects, such as monitor the system's run-time behavior. This may be the intensity of the workload that is managed by server-side components of the application. This helps practitioners make decisions on how to proceed with development. Practitioners also create custom scripts to track and understand users' adaption of their features.

We categorized 12 responses as *manual capture*, in which practitioners described procedures that require no software at all, or only tools that are not primarily designed for the capture of user feedback. This includes practitioners' relying on pen and paper, on meetings or workshops that allow verbal exchange with respect to a prototype, or on indirect contact with end users via a product owner who serves as a proxy. Likewise, explicit user feedback capture via Skype or video recordings were mentioned by two practitioners. Two other practitioners stated to manually capture and externalize observations in wiki pages that are used for knowledge management, such as on Atlassian Confluence. Three practitioners highlighted that they rely on intensive email interactions to capture user feedback. Finally, two practitioners rely on online tools, such as Google Forms, that allow them to conduct surveys to capture user feedback and initiate interviews.

Definitive version available at https://doi.org/10.1109/RE.2019.00026

3) Sources for User Feedback: Figure 5 visualizes the practitioners' answers.

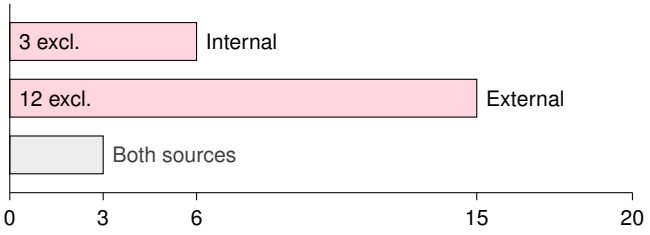

Fig. 5. Number of interviews in which practitioners addressed sources for user feedback. In two interviews, no answers were provided.

We distinguish between internal and external sources of feedback. The developers themselves or team members are internal sources, whereas external sources encompass stakeholders, i.e., customers or end users. We describe the answers in the respective explanations if they address both sources.

Six practitioners mentioned user feedback capture from *internal* sources. One practitioner explicitly highlighted user feedback capture from other departments as a key factor for successful implementation. In their case, their marketing department always uses the latest version of the application under development and thereby becomes an internal customer. This also allows them to directly discuss the feedback in case something is unclear, either via instant messaging or direct "water cooler talk", mostly on a sporadic basis. One practitioner emphasized the trusted relationship with fellow colleagues that allows exchange of honest feedback.

Three practitioners stated that they rely on multiple extensions of the development team, which include dedicated teams that focus on testing, providing feedback, and suggesting improvements. One of those practitioners noted that the decoupling of requirements implementation and testing results in a waterfall-like, non-agile procedure, which, however, might also be related to their product context. The two others described user interface-focused teams that are generally able to provide instant feedback. However, dependent on the extent of the changes, they may not be able to keep pace with the development process and therefore lag behind. The developers consequently receive delayed user feedback, which tends to increase their workload. In the worst case, new feature improvements build upon features that have not yet been tested or validated by user feedback.

Fifteen practitioners addressed user feedback capture from *external* sources. For three practitioners, this means sitting down with the customer and presenting the latest changes to the application under development. According to them, this requires tool support and typically results in high capture and assessment efforts. However, through the personal contact, the customer directly provides feedback to the developer in charge. The interpersonal aspect, i.e., an increased sense of responsibility by the developer and a high appreciation by the customer, was rated as important by the practitioners, as it contributes to the quality of feedback.

Four practitioners noted that the way they deal with user feedback depends on the reporting source. While they review user feedback from time to time, they continuously capture it from external sources without a particular goal in order to be ready to provide on-demand insights when requested. Three practitioners noted to capture user feedback from sources depending on their current development progress. For them this means to start with paper prototypes and collect explicit user feedback in the beginning and transition to more elaborate capturing of implicit user feedback as the product matures. Two other practitioners described a phase-driven capture process, whereby a limited number of users, or the customers themselves, are actively using the latest version of the application, in what is called a beta or pilot phase, while the practitioners are capturing user feedback. Two practitioners reported to focus their user feedback capture process on clearly defining a test scenario which they use to investigate a particular aspect of the feature under development. Three practitioners stated that they follow the provided specification and rely on user feedback that is handed to them by the customer or the product owner and thereby externalize the capture process to another stakeholder. Finally, three practitioners introduced a follow-up approach to capture user feedback; after they implement a change, they actively request feedback from sources they expect to be future users or who have requested a similar functionality before. They inform them in case something is changed on the basis of their feedback.

4) Context of User Feedback: Figure 6 visualizes the practitioners' answers.

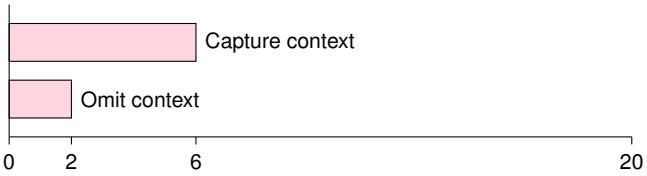

Fig. 6. Number of interviews in which practitioners addressed context of user feedback. In 12 interviews, no answers were provided.

In total, six practitioners stated to *capture context* of user feedback. One technical leader described their option to inspect the environment in which the users interact within the application. Another project leader, who pointed out a less technical, yet marketing-related view of the users' context, described that insights such as a users' purchase history allows them to get a deeper understanding of possible needs and the user environment. A developer noted that context information that allows them to derive helpful insights may be as simple as the local time, indicating whether the application is used during lunch break or typical working hours. Another practitioner reported that the time may also be used to systematically segment user feedback. This supports user feedback comparison, as detailed in Section V-C2. Another practitioner stated that the captured context is similar to a snapshot, rather than a comprehensive user profile. Two practitioners acknowledged to capture, but do not use context information.

Definitive version available at https://doi.org/10.1109/RE.2019.00026

While two other practitioners stated that they consciously *omit context* capture, only one of them provided a reason: They primarily rely on user feedback from fellow team members and colleagues; formal regulations and company guidelines apply legal limits to what information they may capture. In particular, monitoring additional data to capture the context could reveal personal data, such as individual preferences.

C. How do Practitioners Utilize User Feedback in CSE?

This RQ investigates reasons why practitioners capture user feedback, their ability to exploit changes over time, and whether they combine different types of user feedback.

1) Reasons for User Feedback Capture: Figure 7 visualizes the practitioners' answers.

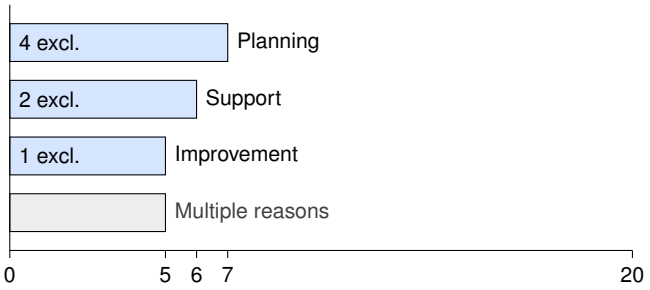

Fig. 7. Number of interviews in which practitioners addressed why they capture user feedback. In eight interviews, no answers were provided.

In seven interviews, practitioners provided answers that are linked to *planning* activities during different phases of an application development cycle. Two practitioners noted to capture user feedback with the goal of performing business modeling, e.g., exploring new technologies, and defer other utilization of the user feedback, such as usability engineering. One of them added that, while there is a need to draw conclusions from user feedback particularly during early phases of product development, the overall goal should be diversification based on user feedback, which is provided for different reasons. As a result, user feedback should be considered but not necessarily cause fundamental changes to the project. Four practitioners stated that they utilize user feedback to prioritize features. For two of them, this primarily means deciding which feature needs to be implemented first or with a higher urgency, while two others use this insight to remove old features. Notably, one practitioner indicated the possibility to also detect the need for new features. One practitioner emphasized the possibility of assessing the acceptance of a feature from a marketing perspective. This allows conclusions to be drawn regarding business-focused metrics, such as answering the question of whether invested efforts were worth it.

Six practitioners reported a use of user feedback with a main focus on *support* activities, such as bug fixing. One of them called this a customer-centered view. Four practitioners outlined that, when user feedback is used to identify bugs, it helps them to draw conclusions about the problem, e.g., by extracting and reconstructing the flow of the events that caused an error. One lead developer emphasized that implicit user feedback in particular is used to ensure that the application and its functionality are working as intended.

Five practitioners described user feedback utilization with the goal of *improving* the quality of a feature and as a decision-support for doing so. For another practitioner, this means to continuously monitor and observe the way the changes are adapted by users, which helps to develop a gut feeling of how the software is performing. Another practitioner observed a change in how they utilize user feedback; they are currently transitioning from a focus on reconstructing errors toward developing an understanding of how the users use the application. This will help them to assess whether the users are using the feature as intended. Two other practitioners continued this idea and stated that sometimes the wrong requirements are used for implementation of a feature—based on incorrect assumptions made previously. In that case, user feedback is the key for adapting requirements and improving the feature.

2) Change of User Feedback Over Time: Figure 8 visualizes the practitioners' answers.

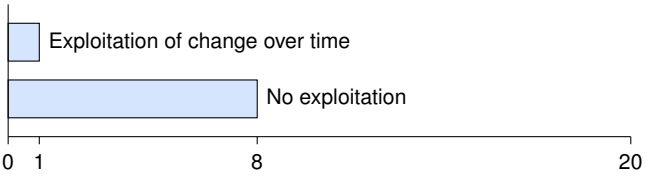

Fig. 8. Number of interviews in which practitioners addressed change of user feedback over time. In 11 interviews, no answers were provided.

Only one practitioner reported to actively *exploit* changes in user feedback. However, this ability is highly correlated to the fact that they are using the data to understand and optimize the business value that the application is offering to its users.

Eight practitioners reported that they do *not exploit* changes in user feedback, while three practitioners elaborated that they are generally able to do that kind of investigations but have not yet done so. One practitioner stated that they have not yet encountered a situation in which an analysis over time would have been beneficial but sees that it could be useful in case of major changes in the user interfaces. Another practitioner shared similar thoughts and added that one always needs to consider the cost effectiveness, as this kind of analysis is something that could require major effort but yield only minor benefit. While stating that their tool probably allows for such a comparison over time, yet another practitioner indicated that they already struggle to cope with most of the basic functionality offered by the tool.

Out of those same eight practitioners, five responded that they lack the technical ability to exploit user feedback over time, but their general impression is that they would like to have such a functionality in place. For example, one practitioner supposed coarse-grained feedback at an early stage, while, after some iterations, they expected more fine-grained feedback to be provided by the user. An increase of the amount of user feedback may support understanding urgent feedback. This enables the detection of user feedback change events.

3) Combination of User Feedback Types: Figure 9 visualizes the practitioners' answers.

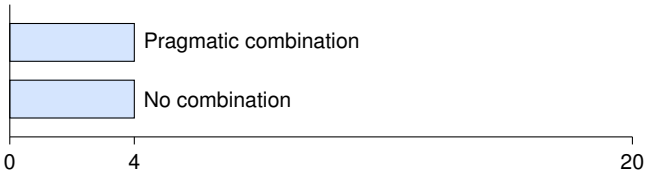

Fig. 9. Number of interviews in which practitioners addressed combination of user feedback types. In 12 interviews, no answers were provided.

Four practitioners described *pragmatic combination* practices regarding user feedback. One practitioner reported being able to combine user profiles with written reviews, such as ones that originate from emails. Another practitioner stated that, while user feedback types are usually handled separately, different types of internal data can be combined. However, it is not being used for feature development. Two other practitioners emphasized that their starting point for the combination of different types of user feedback would be explicit feedback: if they observe peaks in email responses, they initiate a deeper investigation and include other user feedback sources.

Four practitioners indicated that they do *not combine* different user feedback types during development. However, while one practitioner did not see a benefit herein, two are interested in the ability to combine user feedback. Reasons why they do not combine feedback sources are a low complexity level of the application and limited availability of feedback.

VI. Discussion

This section discusses results from Section V by combining practitioners' answers among RQs and relating them to prior-published research. We summarize our interpretation of results in **observations** that form the basis for **recommendations**, in which we reflect on either promising practices that are applied by a few practitioners but should be investigated in more detail, or challenges and untapped potential that were addressed in answers from multiple practitioners. The recommendations guide future work toward continuous capture and utilization of user feedback to benefit requirements engineering.

A. Continuous User Feedback Capture

More than a quarter of the practitioners rely on colleagues and team members in close proximity to retrieve an initial set of user feedback (Figure 5). While reports identified gaps in the requirements communication between departments [26], we found that the practitioners in our study drew positive conclusion regarding collaboration. They highlighted the benefits of direct contact that result in honest and instant responses. Fellow team members are more likely to provide detailed and structured feedback, as they have an intrinsic motivation to contribute to producing and maintaining high-quality software. Feedback from colleagues from outside their project is particularly welcomed too, as they provide new perspectives. The close proximity to team members allows for frequent explicit user feedback capture with almost no delay.

Observation: *Practitioners gain many insights by retrieving user feedback frequently from internal sources, such as fellow team members or colleagues from other teams.*

Recommendation: *Continuous user feedback capture from internal sources should be expanded, as it provides honest responses and leads to diversified requirements.*

The recommendation is an extension to the theory that testing with five users in multiple, consecutive, yet small experiments is most efficient to identify problems regarding requirements such as usability [27]. However, this is clearly insufficient: Feedback must also be captured from external sources, as relying on a selected user group over a long period increases the chances of bias [28]. A challenge remains in the transition from qualitative, low-quantity feedback from the limited scope of "users" in close proximity toward the high-quantity capture of user feedback from a large audience.

This challenge becomes apparent in the light of Figure 1, which reveals that the majority of user feedback is explicit. The practitioners praised the value of face-to-face interactions and the richness of explicit user feedback. It comes as no surprise that explicit user feedback is preferred, as it allows practitioners to capture expressive insights with a high level of detail [28]. This is further emphasized by the high number of manual capture practices (Figure 4). Even during rapid development practices, practitioners rely heavily on manual approaches to capture user feedback, such as collecting user feedback from emails, questionnaires, or video recordings. However, as one practitioner pointed out, such user feedback capture and processing result in a non-agile procedure that slows down overall development and puts the feature's quality at risk. This is also reflected in the frequency with which practitioners capture user feedback. As depicted in Figure 3, only one quarter of practitioners continuously capture user feedback, while more than half rely on event-based and periodic approaches.

Reducing manual steps in favor of (semi-)automated processes may be one way of dealing with this issue. Specifically, this could be achieved by extending tool support, which has already been adopted by the requirements engineering community. For example, there are approaches that enable continuous context-aware user feedback [6], [29]. In addition, with respect to the combination of types of user feedback, FAME is an approach that combines explicit user feedback and usage monitoring to continuously elicit requirements [30].

Observation: *Practitioners rely on explicit over implicit user feedback and manual capture of the former reduces the efficiency of an ongoing development process.*

Recommendation: *Continuous user feedback capture should adapt and extend tool support, e.g., by increasing the use of automatization for processing, to ensure high-quality and expressive requirements in a timely manner.*

B. Continuous User Feedback Utilization

Practitioners reported the utilization of different types of user feedback, while their process on how to derive high-level and actionable insights is unsystematic: On the one hand, as visualized in Figure 2, many practitioners reported to relate user feedback to the entire application, leaving the feedback scattered across multiple features and thereby reducing its value for individual requirements. On the other hand, multiple practitioners indicated that they strive for a feature relationship while capturing user feedback, such as through directly asking the user, or relying on user profiles and feature isolation. However, following the terminology defined in [31], we assess the practitioners' perceived ease of use of such approaches as low, which may be why many practitioners refrain from applying them in practice. It remains a challenge for developers to relate captured user feedback to a feature.

The Quamoco approach addresses the systematic mapping of quality metrics and abstract quality criteria [32], but only recently usage data-driven approaches are proposed [33]. Durán-Sáez *et al*. approached the challenge of relating requirements to fine-grained user events by logging them to derive implicit user feedback for various criteria [34]. For explicit user feedback, Guzman and Maalej developed a natural language processing technique that identifies an application feature from reviews and that allows attaching sentiments to it [35].

The users' context may provide additional support when relating user feedback to a feature under development. Knauss acknowledged that there is no specific definition of context in requirements engineering [36], which may explain the low number of responses in Figure 6. However, she went on to say that context is important to consider for requirements elicitation [36]. For example, increasing the availability of context information could further contribute to the extraction of *tacit requirements* [37], a challenge that developers face when users assume that their needs are obvious [38]. Generally, as one practitioner remarked, trivial context information, such as data points in the form of time stamps, already can significantly contribute to an increase in the value of user feedback.

Observation: *Practitioners' use of tool support for user feedback capture is limited, as they struggle to employ specific user feedback for the feature under development.*

Recommendation: *Continuous user feedback utilization should offer a lightweight concept for creating reference points to relate user feedback to requirements at hand.*

User feedback is necessary for both the validation and verification of requirements [39]. Requirements validation is concerned with making sure that the specified requirements correctly represent the users' needs, i.e., that the elicited requirements are correct and complete. Requirements verification ensures that the design and implementation of a requirement correctly represent its specification. Agile development processes reportedly have a stronger focus on requirements validation than on their verification [40].

From the practitioners answers we cannot derive a unique reason why they capture user feedback, as many provided more than one reason (Figure 7). The answers for the two prevalent codings, i.e., planning and support, suggest that practitioners tend to capture user feedback as a means for requirements verification. They reported to employ the user feedback to iterate on their previous decisions, which contains a re-prioritization of whether to remove an existing feature or not. According to them, these are high-level decisions, which is also reflected in the responses for Figure 2. Many practitioners, however, stated to apply implicit user feedback to guide support inquiries and work on bug fixes. Overall, the utilization of user feedback for a systematic validation of requirements is sparsely applied by practitioners, leaving opportunities unused. In particular the use for quality requirements was only addressed by a few practitioners. This, however, is a viable concept, as it has been shown by Groen *et al*. that users are mentioning quality requirements, such as usability and reliability, when providing explicit feedback [41].

Notably, one practitioner mentioned work on an application for which neither a customer nor users yet exist, which made it difficult to elicit concrete requirements. While they defined the application's functionality, their vision as to how it was to be presented to user was still vague. This meant that their software development was driven by understanding attractive requirements as opposed to must-be or one-dimensional requirements as defined in the Kano model [42]. As a consequence, the practitioners imagined a potential user, put themselves in their role, and used the application to then derive requirements for the feature.

Observation: *Practitioners predominantly rely on user feedback for requirements verification instead of validation.*

Recommendation: *Continuous user feedback utilization should enable the understanding of user feedback as a means to improve existing and to explore new requirements.*

Fewer than half of the practitioners provided an answer to the question as to whether they address change of user feedback over time or its combination (Figure 8 and Figure 9). Only one and four practitioners, respectively, reported that they have a working approach. This reduces the validity of assumptions derived from their answers; however, it strengthens a common theme among practitioners' answers: While many of them respond to be generally able to utilize user feedback, they are not making use of this possibility. We further observe this in answers regarding the capture of context (Figure 6).

Expanding the involvement of the users during feedback utilization may help to improve this aspect. Stade *et al*. reported how tool support helped a software company to increase the communication with users [43]. Gómez *et al*. described a next generation architecture for app stores that enables more actionable feedback from users [44]. Maalej *et al*. predicted that *"[c]ommunication with users will move from being unidirectional [...] to being bidirectional [...]"* [45].

This vision is reflected in a few answers by practitioners who indicated that they are moving toward a dialog-based utilization of user feedback. For example, regardless of user feedback sources (Figure 5), practitioners stated that the ongoing and active exchange between developers and users emerges as successful in practice. One practitioner emphasized the value of retaining contact with active users and informing them about new feature increments. This increases the possibility to retrieve more feedback and serves as a way to validate the changes to ensure that they are in line with the users' needs. Similarly, another practitioner described a dedicated checkbox allowing users to request updates on the latest changes.

With respect to Figure 7 and the goal to utilize user feedback for feature improvement, one practitioner envisioned a functionality that allows to define a mechanism that triggers as soon as a user starts interacting with a feature. This mechanism should enable developers to initiate a well-defined scenario that guides the user throughout their exploration of a new feature increment and to observe them while doing so. This would help the practitioners to pinpoint the users' attention to a detail on which they would like to have feedback.

Observation: *Most practitioners utilize user feedback individually, however, there is a trend toward its consolidation.*

Recommendation: *Continuous user feedback utilization should cover interactive practices to explore user feedback and increase the developer–user communication to exploit unused potential for enhancing the quality of requirements.*

VII. Related Work

We list empirical studies that addressed how practitioners deal with user feedback and report results similar to ours.

Heiskari and Lehtola conducted a case study to investigate the state of user involvement in practice [46]. They found explicit user feedback as the most common type of insight for development—which is supported by our results in the Section V-A1. They identified a need to involve users in a more organized way than in the state-of-the-practice and to establish defined feedback processes. This is reflected in the number of manual capture procedures described in Section V-B2.

Ko *et al*. performed a case study to identify constraints on capturing and utilizing post-deployment user feedback [47]. They found the heterogeneity of intended use cases of an application as the major constraint for the utilization of user feedback, as it makes prioritization difficult. Similar to this finding, the practitioners in our study noted that more context about the user feedback, such as user profile in Section V-B4, is needed to support the utilization of the feedback.

Stade *et al*. reported on a case study and online survey on how software companies elicit end-user feedback [48]. They found a need for more context, which again corresponds to our results in Section V-B4. They identified feedback channels such as hotlines or phone calls, email, and meetings being used the most. This confirms our results in Section V-A1.

Olsson and Bosch investigated post-deployment data usage in embedded systems during three case studies [49]. They found that such data was neither used to acquire insights on individual features, nor to innovate new functionality [49]. Our results in Section V-C1 support the predominant use of user feedback for planning and bug fixing, rather than feature improvement in the sense of innovation.

Pagano and Bruegge reported on a case study with five companies in which they investigated how professionals deal with *"post-deployment end-user feedback"* [50]. They stated that the utilization of user feedback is mainly a manual process, which is reflected in our results in Section V-B2. They conclude that tool support is required to utilize user feedback for other practices, such as for requirements engineering [50].

VIII. Conclusion

Continuous software engineering evolved as a process for software evolution [1], [2]. While most of the practices, such as continuous integration or delivery, are well-researched [8]–[13], the use of user feedback, in particular with respect to requirements engineering, has fallen behind [14].

We conducted 20 interviews with practitioners from 17 companies to understand the state-of-the-practice of user feedback capture and utilization during CSE. In this paper, we provide results to nine areas of interest. While some of the results appear obvious [25], we strive to provide empirical evidence for them. All practitioners rely on explicit user feedback. Fourteen practitioners described how they relate user feedback to an artifact. While most practitioners rely on tool support, the numbers for manual and non-continuous capture of user feedback are high. Most feedback is captured from external sources. User feedback is predominantly utilized for requirements verification. Practitioners sparsely consider user feedback context, exploration over time, and its combination.

From the results, we derived five recommendations, based on either promising practices or challenges. We conclude that continuous user feedback capture benefits from user feedback provided by internal sources as well as by adapted and extended tool support. Continuous user feedback utilization can be improved through user feedback reference points. This establishes a focus on requirements validation, which could be further advanced by more user feedback interaction.

The recommendations guide our current and future work. We designed a user feedback reference concept [51], developed an approach to automate user feedback capture [52], and explored a way to validate requirements [53]. We will consolidate the continuous user feedback capture and utilization in a *continuous user understanding* activity and provide tool support. This aims to improve requirements engineering regarding the completeness and correctness of requirements.

Acknowledgment

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 insights and the anonymous reviewers for their valuable feedback to improve this work.

REFERENCES

[1] J. Bosch, *Continuous Software Engineering: An Introduction*. Springer, 2014, pp. 3–13.

[2] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: A roadmap and agenda," *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.

[3] A. Kleebaum, J. O. Johanssen, B. Paech, R. Alkadhi, and B. Bruegge, "Decision Knowledge Triggers in Continuous Software Engineering," in *Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering*, ser. RCoSE '18. ACM, 2018, pp. 23–26.

[4] S. Krusche and B. Bruegge, "User Feedback in Mobile Development," in *Proceedings of the 2nd International Workshop on Mobile Development Lifecycle*, ser. MobileDeLi '14. ACM, 2014, pp. 25–26.

[5] S. Krusche, L. Alperowitz, B. Bruegge, and M. Wagner, "Rugby: An Agile Process Model Based on Continuous Delivery," in *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, ser. RCoSE '14. ACM, 2014, pp. 42–50.

[6] W. Maalej, H.-J. Happel, and A. Rashid, "When Users Become Collaborators: Towards Continuous and Context-Aware User Input," in *Proceedings of the 24th Conference Companion on Object Oriented Programming Systems Languages and Applications*, ser. OOPSLA '09. ACM, 2009, pp. 981–990.

[7] G. Schermann, J. Cito, and P. Leitner, "Continuous Experimentation: Challenges, Implementation Techniques, and Current Research," *IEEE Software*, vol. 35, no. 2, pp. 26–31, 2018.

[8] M. Meyer, "Continuous integration and its tools," *IEEE Software*, vol. 31, no. 3, pp. 14–16, 2014.

[9] D. Ståhl and J. Bosch, "Modeling continuous integration practice differences in industry software development," *Journal of Systems and Software*, vol. 87, pp. 48–59, 2014.

[10] E. Laukkanen, J. Itkonen, and C. Lassenius, "Problems, causes and solutions when adopting continuous delivery—A systematic literature review," *Information and Software Technology*, vol. 82, pp. 55–79, 2017.

[11] L. Chen, "Continuous Delivery: Overcoming adoption challenges," *Journal of Systems and Software*, vol. 128, pp. 72–86, 2017.

[12] M. Shahin, M. Ali Babar, and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," *IEEE Access*, vol. 5, pp. 3909–3943, 2017.

[13] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, vol. 50, no. 9, pp. 833–859, 2008.

[14] P. Rodríguez, A. Haghighatkhah, L. E. Lwakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J. M. Verner, and M. Oivo, "Continuous deployment of software intensive products and services: A systematic mapping study," *Journal of Systems and Software*, vol. 123, pp. 263–291, 2017.

[15] J. O. Johanssen, A. Kleebaum, B. Paech, and B. Bruegge, "Practitioners' eye on continuous software engineering: An interview study," in *Proceedings of the 2018 International Conference on Software and System Process*, ser. ICSSP '18. ACM, 2018, pp. 41–50.

[16] ——, "Continuous software engineering and its support by usage and decision knowledge: An interview study with practitioners," *Journal of Software: Evolution and Process*, vol. 31, no. 5, p. e2169, 2019.

[17] A. Kleebaum, J. O. Johanssen, B. Paech, and B. Bruegge, "How do Practitioners Manage Decision Knowledge during Continuous Software Engineering?" in *Proceedings of the 31st International Conference on Software Engineering and Knowledge Engineering*, ser. SEKE'19. KSI Research Inc., 2019, pp. 735–740.

[18] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, ser. HUC '99. Springer-Verlag, 1999, pp. 304–307.

[19] A. K. Dey, "Enabling the Use of Context in Interactive Applications," in *Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '00. ACM, 2000, pp. 79–80.

[20] M. D. Myers and M. Newman, "The Qualitative Interview in IS Research: Examining the Craft," *Information and Organization*, vol. 17, no. 1, pp. 2–26, 2007.

[21] P. Runeson, M. Host, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, 2012.

[22] K.-J. Stol and B. Fitzgerald, "The ABC of Software Engineering Research," *ACM Transactions on Software Engineering and Methodology*, vol. 27, no. 3, pp. 1–51, 2018.

[23] J. Saldaña, *The Coding Manual for Qualitative Researchers*, 2nd ed. SAGE Publications, 2009.

[24] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, *Selecting Empirical Methods for Software Engineering Research*. Springer London, 2008, ch. 11, pp. 285–311.

[25] M. Torchiano and F. Ricca, "Six reasons for rejecting an industrial survey paper," in *Proceedings of the 1st International Workshop on Conducting Empirical Studies in Industry*, ser. CESI '2013, 2013, pp. 21–26.

[26] L. Karlsson, Å. Dahlstedt, J. Natt och Dag, B. Regnell, and A. Persson, "Challenges in market-driven requirements engineering—An industrial interview study," in *Proceedings of the 8th International Workshop on Requirements Engineering: Foundation for Software Quality*, 2002, pp. 37–49.

[27] J. Nielsen and T. K. Landauer, "A Mathematical Model of the Finding of Usability Problems," in *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, ser. CHI '93. ACM, 1993, pp. 206–213.

[28] M. J. Gallivan and M. Keil, "The user–developer communication process: a critical case study," *Information Systems Journal*, vol. 13, no. 1, pp. 37–68, 2003.

[29] D. Dzvonyar, S. Krusche, R. Alkadhi, and B. Bruegge, "Context-aware User Feedback in Continuous Software Evolution," in *Proceedings of the International Workshop on Continuous Software Evolution and Delivery*, ser. CSED '16. ACM, 2016, pp. 12–18.

[30] M. Oriol, M. Stade, F. Fotrousi, S. Nadal, J. Varga, N. Seyff, A. Abello, X. Franch, J. Marco, and O. Schmidt, "FAME: Supporting Continuous Requirements Elicitation by Combining User Feedback and Monitoring," in *Proceedings of the 26th International Requirements Engineering Conference*, ser. RE '18. IEEE, 2018, pp. 217–227.

[31] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, "User Acceptance of Computer Technology: A Comparison of Two Theoretical Models," *Management Science*, vol. 35, no. 8, pp. 982–1002, 1989.

[32] S. Wagner, A. Goeb, L. Heinemann, M. Kläs, C. Lampasona, K. Lochmann, A. Mayr, R. Plösch, A. Seidl, J. Streit, and A. Trendowicz, "Operationalised product quality models and assessment: The Quamoco approach," *Information and Software Technology*, vol. 62, no. 1, pp. 101–123, 2015.

[33] X. Franch, C. Ayala, L. López, S. Martínez-Fernández, P. Rodríguez, C. Gómez, A. Jedlitschka, M. Oivo, J. Partanen, T. Räty, and V. Rytivaara, "Data-driven requirements engineering in agile projects: The q-rapids approach," in *Proceedings of the 25th International Requirements Engineering Conference Workshops*, ser. REW '17. IEEE, 2017, pp. 411–414.

[34] R. Durán-Sáez, X. Ferré, H. Zhu, and Q. Liu, "Task Analysis-Based User Event Logging for Mobile Applications," in *Proceedings of the 25th International Requirements Engineering Conference Workshops*, ser. REW '17. IEEE, 2017, pp. 152–155.

[35] E. Guzman and W. Maalej, "How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews," in *Proceedings of the 22nd International Requirements Engineering Conference*, ser. RE '14. IEEE, 2014, pp. 153–162.

[36] A. Knauss, "On the usage of context for requirements elicitation: End-user involvement in IT ecosystems," in *Proceedings of the 20th International Requirements Engineering Conference*, ser. RE '12. IEEE, 2012, pp. 345–348.

[37] V. Gervasi, R. Gacitua, M. Rouncefield, P. Sawyer, L. Kof, L. Ma, P. Piwek, A. de Roeck, A. Willis, H. Yang, and B. Nuseibeh, *Unpacking Tacit Knowledge for Requirements Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 23–47.

[38] J. A. Bubenko, "Challenges in requirements engineering," in *Proceedings of International Symposium on Requirements Engineering*, ser. RE '95. IEEE, 1995, pp. 160–162.

[39] K. Pohl and C. Rupp, *Requirements Engineering Fundamentals*, 2nd ed. Rocky Nook, 2015.

[40] B. Ramesh, L. Cao, and R. Baskerville, "Agile requirements engineering practices and challenges: an empirical study," *Information Systems Journal*, vol. 20, no. 5, pp. 449–480, 2010.

[41] E. C. Groen, S. Kopczyńska, M. P. Hauer, T. D. Krafft, and J. Doerr, "Users — The Hidden Software Product Quality Experts?: A Study on How App Users Report Quality Aspects in Online Reviews," in *Proceedings of the 25th International Requirements Engineering Conference*, ser. RE '17. IEEE, 2017, pp. 80–89.

[42] E. Sauerwein, F. Bailom, K. Matzler, and H. H. Hinterhuber, "The Kano model: How to delight your customers," in *Proceedings of the 9th International Working Seminar on Production Economics*, ser. WSPE '96, R. W. Grubbström, Ed., vol. 1. Innsbruck: Elsevier, 1996, pp. 313–327.

[43] M. Stade, M. Oriol, O. Cabrera, F. Fotrousi, R. Schaniel, N. Seyff, and O. Schmidt, "Providing a User Forum is not enough: First Experiences of a Software Company with CrowdRE," in *Proceedings of the 25th International Requirements Engineering Conference Workshops*, ser. REW '17. IEEE, 2017, pp. 164–169.

[44] M. Gómez, B. Adams, W. Maalej, M. Monperrus, and R. Rouvoy, "App Store 2.0: From Crowdsourced Information to Actionable Feedback in Mobile Ecosystems," *IEEE Software*, vol. 34, no. 2, pp. 81–89, 2017.

[45] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward Data-Driven Requirements Engineering," *IEEE Software*, vol. 33, no. 1, pp. 48–54, 2016.

[46] J. Heiskari and L. Lehtola, "Investigating the State of User Involvement in Practice," in *Proceedings of the 6th Asia-Pacific Software Engineering Conference*. Penang, Malaysia: IEEE, 2009, pp. 433–440.

[47] A. J. Ko, M. J. Lee, V. Ferrari, S. Ip, and C. Tran, "A case study of post-deployment user feedback triage," in *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '11. Waikiki, Honolulu, HI, USA: ACM Press, 2011, pp. 1–8.

[48] M. Stade, F. Fotrousi, N. Seyff, and O. Albrecht, "Feedback Gathering from an Industrial Point of View," in *Proceedings of the 25th International Requirements Engineering Conference*, ser. RE '17, A. Moreira and J. Araújo, Eds. Lisbon, Portugal: IEEE, 2017, pp. 63–71.

[49] H. Holmström Olsson and J. Bosch, "Towards Data-Driven Product Development: A Multiple Case Study on Post-deployment Data Usage in Software-Intensive Embedded Systems," in *Lecture Notes in Business Information Processing*, B. Fitzgerald, K. Conboy, K. Power, R. Valerdi, L. Morgan, and K.-J. Stol, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 167, pp. 152–164.

[50] D. Pagano and B. Brügge, "User Involvement in Software Evolution Practice: A Case Study," in *Proceedings of the International Conference on Software Engineering*, ser. ICSE '13. IEEE Press, 2013, pp. 953–962.

[51] J. O. Johanssen, A. Kleebaum, B. Bruegge, and B. Paech, "Feature Crumbs: Adapting Usage Monitoring to Continuous Software Engineering," in *Product-Focused Software Process Improvement*, M. Kuhrmann, K. Schneider, D. Pfahl, S. Amasaki, M. Ciolkowski, R. Hebig, P. Tell, J. Klünder, and S. Küpper, Eds. Springer International Publishing, 2018, pp. 263–271.

[52] J. O. Johanssen, L. M. Reimer, and B. Bruegge, "Continuous thinking aloud," in *Proceedings of the Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution*, ser. RCoSE-DDrEE '19. IEEE Press, 2019, pp. 12–15.

[53] J. O. Johanssen, J. P. Bernius, and B. Bruegge, "Toward Usability Problem Identification Based on User Emotions Derived from Facial Expressions," in *Proceedings of the 4th International Workshop on Emotion Awareness in Software Engineering*, ser. SEmotion '19. IEEE Press, 2019.