Context-Aware User Feedback in Continuous Software Evolution

Dora Dzvonyar
Technical University of Munich
Munich, Germany
dzvonyar@in.tum.de

Stephan Krusche
Technical University of Munich
Munich, Germany
krusche@in.tum.de

Rana Alkadhi
Technical University of Munich
Munich, Germany
alkadhi@in.tum.de
King Saud University
Riyadh, Saudi Arabia

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

ABSTRACT

User feedback is an important means of validating requirements and discovering new requirements in continuous software evolution. However, users have a low motivation to provide feedback and prefer applications which do not interrupt their work. Due to missing context information, developers have difficulties to analyze feedback, and to integrate it into their development work.

In this paper, we describe CAFE, a context-aware feedback system which consists of: (1) a framework for collecting in-situ user feedback enriched with usage context data; and (2) a process for integrating feedback into a team's development activities. While the process is applicable to all kinds of applications, the implemented framework concentrates on mobile user feedback and its particular challenges.

We evaluated CAFE in a mobile application. Our results indicate that the system is a valuable step toward increasing user motivation to provide feedback and decreasing the developers' effort to integrate feedback, ultimately improving user involvement.

CCS Concepts

•Social and professional topics → Systems development; *Software maintenance;* **•Software and its engineering → Software evolution; •Human-centered computing** → *Ubiquitous and mobile computing systems and tools;* Participatory design;

Keywords

User Involvement, Requirements Traceability, Usage Context, Agile, Mobile Feedback

CSED'16, May 14-15 2016, Austin, TX, USA

ISBN 978-1-4503-4157-8/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2896941.2896952

1. INTRODUCTION

User involvement has been recognized as an important source of information for development teams [7, 11], and has a positive impact on user satisfaction as well as project and system success [2, 15]. In an agile software project, developers need to be able to react to quickly changing requirements and unclear specification [26]. Involving users in the software development process helps to validate requirements and can lead to the early recognition of necessary changes to the system [23].

Although feedback is integral to continuous software evolution, several challenges exist in this domain. Studies have shown that users generally have a low motivation to give feedback, and that users prefer that the application does not interrupt their work for feedback provision [1]. If the process is not fast, easy and integrated, their reluctance to feedback increases even further [24].

Additional problems arise after users have provided feedback. If feedback arrives through different channels and in heterogeneous form, it becomes cumbersome for the development team to collect and manage [20, 10]. Another challenge is missing context information in the feedback artifacts [15]. The more the developers know about the user's context, the easier they can understand his problem or request additional information [7]. Since end users have no background in software engineering, they do not know which information the development team needs, leading to a mismatch between what users provide and what developers require [4].

In addition to missing information, spontaneous feedback is difficult to analyze if it is written in unstructured natural language [24]. The author of the message might not specify which feature he is providing feedback on, or might comment on multiple aspects of the application. It is therefore difficult for developers to transform user feedback into a requirement or a reproducible bug which they can use as basis for future development iterations [20, 10]. Researchers claim that "a clear research gap exists for mechanisms to use customer feedback in the most appropriate way so that information can be quickly interpreted" [22]. Once the developers have extracted requirements from the received feedback, it is ben-

eficial to link the feedback to the implemented requirement to trace the evolution of the system [15].

These difficulties lead to low developer motivation to consider and integrate feedback. Evaluations have shown that developers are worried about the collection and analysis of feedback being overly time-consuming [16]. If developers do not change the application based on the received feedback, the users' motivation to provide feedback as well as the users' satisfaction decrease, because their opinion does not have an impact on the application [18]. In the long term, the development team misses a valuable opportunity to validate their system regularly and to discover new requirements [23].

Our aim is to design a mechanism for user feedback which addresses these challenges. We propose CAFE, a **C**ontext-**A**ware **FE**edback system which consists of a *framework* for collecting user feedback as well as a *process* for integrating it into the team's development activities. While the process is applicable to all kinds of applications, the implemented framework concentrates on mobile user feedback and its particular challenges [14].

CAFE provides a channel for users to give feedback with as little effort as possible, and to record this feedback in the issue tracking system, which is already used for development work. By saving feedback and development tasks in the same system, we can realize links between the two and thus document the evolution of requirements through user feedback, which we call *feedback traceability*. In addition, we enrich the received feedback by recording the usage context, and give the user the opportunity to further structure his message. By attaching this additional data to the feedback, CAFE provides developers with a better understanding of the user's situation.

This paper is organized as follows. In Section 2 we describe the main concepts and components of CAFE. This is followed by a first evaluation of the feedback mechanism described in Section 3. We discuss the results of this evaluation in Section 4 in terms of implications and limitations, and extract a conclusion and next steps in Section 5.

2. CONTEXT-AWARE FEEDBACK SYSTEM

CAFE combines an approach of integrating feedback into the development process with concepts from the domains of usage context tracking and feedback traceability. In the following, we summarize background and related work in each of these fields and explain how CAFE integrates these approaches into a comprehensive framework and process.

2.1 Integration of feedback into development

A successful user involvement strategy needs a clear process including infrastructure, procedures and communication channels as well as support by the management [10, 6]. Process models concerning the integration of user feedback into continuous software evolution presented by fellow researchers emphasize that user feedback should be taken into account regularly and have a set place in the software evolution process. Schneider presents a similar cycle of "transforming awareness to raw feedback, and further to valid requirements and business advantage" through four consecutive steps of feedback composing and sending, analysis by the development team, and the subsequent improvements to the system [24]. Pagano's domain-independent model for continuous user involvement, also advocates for the regular analysis of user feedback in both early and more advanced

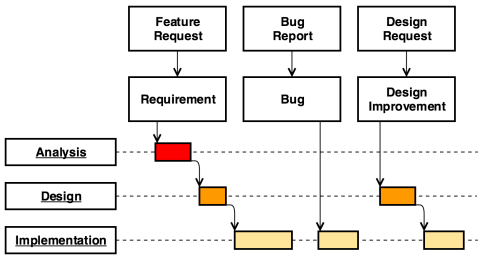

Figure 1: Integration of feedback into workflows of the development team, based on [13]

stages of a software project [20]. However, these models lack integration into specific processes and activities carried out by the development team [15, 20].

We therefore base CAFE on the feedback lifecycle by Krusche et al. [13], which is more concrete with regard to the integration of user feedback into specific activities of the development process. Depending on the kind of each feedback item, it is incorporated into the Analysis, Design or Implementation workflows. While a Bug Report can be implemented right away, a Design Request is first processed in the Design workflow, and a Feature Request has to be analyzed before it can proceed through the Design and Implementation activities, as shown in Figure 1. Feedback is stored in the Backlog in the form of Issues [13]. CAFE instantiates this feedback lifecycle with concrete tool support for gathering feedback and storing it in the issue tracking system. User feedback is gathered in situ, i.e., without the user leaving the application he is currently using [20].

Figure 2 gives an overview of CAFE's process from both the User's and the Developer's point of view. Members of the development team (shown in blue) perform all activities in the issue tracker, while users (in green) interact with the system through the application which integrates CAFE's feedback gathering component. Before a new release of the application, the Developer selects changes to be included in the new version and identifies the corresponding issues in the issue tracking system. If a change is visible to the user, the Developer specifies the usage context in which the change is noticeable. He also gives the change a title that the user is able to understand. These two activities can be performed in the issue tracker and are used to display context-relevant suggestions to the user, as described in Section 2.2. The application can now be released and installed by the User.

When the User has installed and used the new version and wants to send feedback to the development team, he initiates the feedback process through a UI element or gesture provided by the application which integrates CAFE. At this point, CAFE temporarily stores the usage context from which the process was started and displays the feedback screen where the user can immediately start sharing his opinion. The User types in his message and can send the feedback right away, but if he is more motivated, he can also decide to specify more details such as the type of his message (e.g. feature request vs. bug report). CAFE also presents the user with a list of changes relevant for his usage context by displaying issues which are important in the screen from which he initiated the feedback process, as described in detail in Section 2.2. He can select one of these suggestions as the subject of his feedback if he thinks that his message

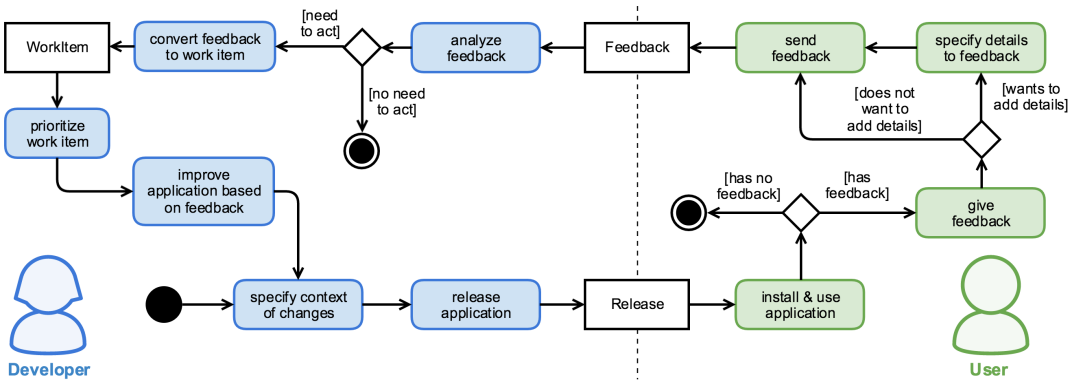

Figure 2: Simplified process model of CAFE

concerns one of the listed changes. If the User specifies additional details, these are attached to the resulting Feedback issue which is saved in the issue tracking system. In particular, the selected link between the new Feedback Issue and the previous change is established, which is the basis of feedback traceability as described in Section 2.3.

The process continues with the analysis of the received feedback by a member of the development team. Feedback analysis is possible directly in the issue tracking system, and the feedback is stored in a structure similar to the issues that the team already uses for development. Thus, team members are able to understand the feedback without much additional effort. The usage context which is stored by CAFE in the feedback issue also helps in the interpretation of the user's message. If the team needs to act based on the received message, the Developer can convert the feedback to a work item, e.g. a Bug or a Feature Request, as shown in Figure 1. The resulting work item is again linked to the feedback issue it originated from, thus realizing traceability as described in Section 2.3. The work item can now be prioritized and implemented. The subsequent improvement of the application based on the received user feedback closes the loop for continuous software evolution.

The practice of saving feedback in the issue tracker is not a new approach. For instance, Yetim et al. implemented a component which can be integrated into certain desktop applications [27]. Our work is similar because we provide a way to send feedback to the issue tracker, but we add an implementation for mobile devices. We extend their approach by incorporating context parameters and traceability of requirements through received feedback, as described in the following sections.

2.2 Context in user feedback

CAFE enriches user feedback with variables of the usage context. We follow the definition of Pagano, who defines the term 'context' to include "all current and past conditions and events which influence the interaction of a user with an application or the execution of an application on a system" [20]. Knowing about the user's context helps to reproduce the situation in which he encountered a problem and therefore facilitates feedback analysis [7]. Our goal is to enrich the gathered feedback with additional data and compensate for the user's potential failure to provide the necessary information to the development team.

Some approaches automatically save details of the usage context to user feedback [17, 20], while others provide possibilities for the user to manually structure his feedback, such as specifying the category of his message or selecting his concern from a list of pre-defined problems [25, 24]. CAFE bundles aspects of multiple approaches into one comprehensive user feedback framework. The system combines *automatic* tracking of the usage context with possibilities for the user to *manually* specify the context of his feedback, thus giving it additional structure.

For instance, CAFE saves the application screen from which the user initiated the feedback process along with details of the device the application is used on (e.g. model, OS version, language), and attaches it to the feedback in natural language given by the user. CAFE can also be configured to track important interactions of the user with the application, for example which screens the user visited or what UI elements he interacted with prior to giving feedback. In addition to this automatically gathered information, CAFE presents the user with a list of recent changes in the application which are relevant to his current usage context. After the user initiates the feedback process from a certain screen of the application, he gets a list of recently implemented changes which are visible in that screen because he is likely to comment on one of those changes. If the user selects one of these suggestions as the subject of his message, the resulting feedback is linked to the previously implemented change, as described in Section 2.3.

2.3 Feedback traceability

We derive the term *Feedback Traceability* from the domain of Requirements Traceability, which is defined as "the ability to describe and follow the life of a requirement [...] from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of on-going refinement and iteration in any of these phases" [9]. Requirements traceability is considered as important for a development team to verify the system against the original requirements and to assess the impact of prospectives change to the system [8]. Since user feedback can be both the *source* of a new requirement, and the *consequence* of an implemented requirement, it is important to represent this bi-directional trace between feedback and requirements. CAFE records and visualizes the links between feedback and requirements, which addresses a deficit in current user involvement practice reported by developers [16].

Related work in this domain includes Nagel et al.'s approach to link voice recordings to development tasks resulting from comments [19]. Panichella et al. use classification techniques to suggest relevant links between source code and other artifacts such as requirements or models [21]. However, these approaches lack a concrete and integrated representation of the relationship between user feedback and subsequent development work, which CAFE achieves by incorporating feedback into the issue tracking system as described in Section 2.1. Since issue trackers support establishing links between issues and are already used for knowledge representation by many development teams [3], we consider them a viable solution to address this shortcoming.

CAFE establishes a link between a user feedback item and a previously implemented requirement by presenting context-relevant suggestions to the user as described in Section 2.2. These suggestions correspond to changes that the developers have configured in the issue tracker, as described in Section 2.1. If the user selects one of these items as the subject of his feedback, the resulting feedback issue is linked to the issue of the implemented requirement. The requirements resulting from the feedback can then be linked to the feedback message directly in the issue tracker.

Feedback traceability is useful to analyze what impact a particular change had on the users, or to assess the importance of a potential new feature for the user base by showing how many users suggest that feature. It also serves as a knowledge base for the team, e.g. by documenting the reasons why a change was implemented. We did not find any other feedback mechanism which realizes a link between requirements and user feedback in an integrated system that can be used for development work. We therefore consider feedback traceability to be a major contribution of CAFE.

3. EVALUATION

To demonstrate the applicability of CAFE, we conducted a formative evaluation within a university environment and evaluated the usage of the system from both the user and the developer side. We are examining the following hypotheses in the evaluation of CAFE:

H1 CAFE increases users' motivation to provide feedback.

- **H1.1** The introduction of CAFE into an application leads to more user feedback.
- **H1.2** Users take the opportunity provided by CAFE to specify additional information in their feedback.

H2 CAFE decreases developers' effort to integrate feedback.

- **H2.1** The usage context collected by CAFE decreases the effort to analyze user feedback.
- **H2.2** Storing feedback in the issue tracker decreases the effort to integrate it into the development.

3.1 Setting

The formative evaluation took place in the context of a multi-project capstone course conducted twice per year at the Chair for Applied Software Engineering at the Technical University in Munich [5, 12]. In this course, participants produce mobile applications in real software projects for customers from industry based on an agile process model called Rugby which is a variation of Scrum [13].

We integrated CAFE into an application of this course and evaluated its usage from the user and the developer side. This application, named Conada, is designed to promote collaborative fitness at the workplace. We set up CAFE to be triggered from every screen of the application through a shake gesture because this did not require a new element to be added to the user interface, which would not have fit in every screen. We configured the feedback system to track user interaction and prepared it to show suggestions of implemented functionality to the user so that he can link his feedback, as described in Sections 2.2 and 2.3.

We integrated CAFE into the application Conada and divided our evaluation into two consecutive stages. First, we conducted experiments with users to collect feedback, which the Conada developers then used for further development. Second, we held semi-structured interviews with members of the development team. Both evaluation phases are described in the following subsections.

3.1.1 User Experiments

We first evaluated CAFE from a user's perspective in the form of quasi-experiments conducted on two consecutive days in July 2015. Participants were either students or persons involved with the university. In total, 15 subjects took part in this phase of the evaluation, in sessions either on their own or with another person present to evaluate the collaborative workout functionality of Conada.

Each session lasted 30-45 minutes depending on the participants' speed and was organized as follows: we explained to the participants that they were about to give feedback on a mobile application. We made it clear that while we were developing a feedback system, they would not evaluate the feedback system, but use the Conada application and communicate their thoughts to the developers through the feedback system. Moreover, we assured them that the nature of their feedback (e.g. negative vs. positive comments) would not influence the outcome of the evaluation to ensure that they provide their honest opinion. To ensure that each participant tries out the main functionality of the application, we provided a sheet with a minimal usage scenario which the participants were supposed to go through during the experiment. In addition, we encouraged them to use the application freely.

On the first day of evaluation, we put 7 participants into an 'uninformed group' to which we did not introduce CAFE. The 8 participants of the second day were put into an 'informed group', for which we opened CAFE's feedback screen once, told them that they could type a message and indicated where they could add details to it. However, we were careful to only give them a neutral introduction without implying that these details were important to the developers. We did not reveal the context-tracking functionality of CAFE to either group, and merely told them to shake the device whenever they thought of something to tell to the developers. We created the two groups to examine whether there are differences in the usage of the feedback system between uninformed and informed participants.

After each participant had spent on average 20 minutes using Conada and giving feedback to the developers of the application, we gave them a questionnaire asking for their

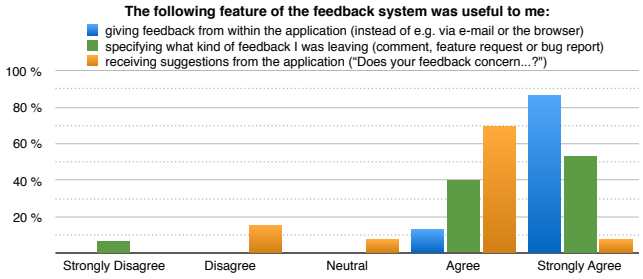

Figure 3: Users' opinion about three key functionalities of CAFS

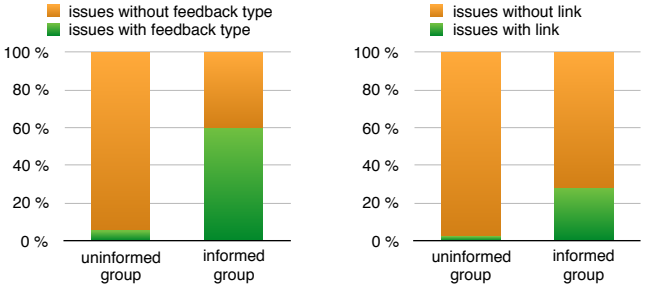

Figure 4: Feedback issues uninformed vs. informed group

opinion on different aspects of the feedback system. This controlled environment gave us the opportunity to obtain in-depth data on the users' experience which would not have been possible in a real-world setting.

3.1.2 Developer Interviews

The above-described experiments resulted in user feedback which we used for evaluating CAFE from the developers' perspective. The Conada development team was asked to analyze and consider the received feedback in their final development iteration. We then conducted semi-structured, one-on-one interviews with six members of the development team as well as their project managers.

The interviews were audio recorded and included questions on the team member's experience with the feedback system during the project. In addition, we asked their opinion on how it could be used in future projects with more time to consider and implement feedback. We asked each interviewee the same questions, based on a five point Likert scale and gave them the opportunity to justify their responses freely. This semi-structured format allowed us to ask follow-up questions if we wanted to explore a particular topic in greater detail. We also asked the participants open questions about their ideas for future improvements to CAFE and possible projects in which they would use the system. This evaluation approach led to a combination of categorical and unstructured data which was useful as a base for the further development.

3.2 Results

In this section, we summarize the results from both the user experiments and the developer interviews.

3.2.1 User Experiments

Each of the 15 participants gave feedback at least four times while using the Conada application, resulting in 93 total feedback items which we analyzed along with the questionnaire data. The first questions referred to the participants' background in software engineering as well as their general attitude towards feedback. 73 % of the subjects claimed that they never or almost never gave feedback on applications, although they knew that it was valuable to a development team. However, when asked for the main reasons for providing feedback, the answers showed a clear consensus on negative motives. Encountering a problem while using an application or being dissatisfied with the current functionality were the main reason.

We asked the participants for their opinion on the key features of CAFE using a five point Likert scale. An excerpt of the results is shown in Figure 3. 87 % strongly agreed that giving feedback without the need to leave the application was useful. 53 % strongly agreed that they liked being able to specify the category of their feedback (comment, feature request or bug report). Regarding the suggestions offered by CAFE which allow users to link feedback to a specific change in the application, the answers were not as clear, but still positive (77 % agreed or strongly agreed).

As a next step, we analyzed how participants used the features of CAFE by inspecting the feedback stored in the issue tracker. We noticed a clear difference between the 'informed group' which had prior knowledge of the functionality of CAFE and the 'uninformed group' which was not introduced to the feedback system. The effect of simply mentioning these features to the participants is shown in Figure 4. Both charts contain the proportion of feedback items in the issue tracker with and without additional data, compared between the informed and uninformed group.

The prior introduction to CAFE's functionality had a strong effect on the usage of both features. While only 6 % of the issues had a type (comment, feature request, bug report) assigned in the uninformed group, the informed participants picked a type for 60 % of their feedback messages. The indication of the suggestion functionality of CAFE described in Section 2.3 also led to a strong increase of its usage, raising the proportion of feedback items linked to a previously implemented requirement by a factor of 10.

To investigate whether CAFE creates a feeling of involvement with users, we asked the participants how convinced they were that their feedback would be considered by the app creators, compared to when they would give feedback via the AppStore or email. 60 % are more convinced of their impact on the application through CAFE compared to the AppStore, and 73 % stated the same opinion when they compare CAFE to providing feedback via email.

3.2.2 Developer Interviews

The developers of Conada were pleased to get feedback from users because it helped them to prioritize the tasks for the last development iteration before the final presentation of their project. 75 % of the interviewed developers and project managers strongly agreed that the collection of feedback in the issue tracker was useful to their team. Developers and project managers agreed that CAFE made it "less of a hassle" to look at feedback, because it was "just there" and they did not need to leave the system they already used regularly to find it. They also remarked that being able to see what kind of feedback a particular change evoked could be useful for analysis. 83 % of interviewees agreed that CAFE is applicable in industry projects they

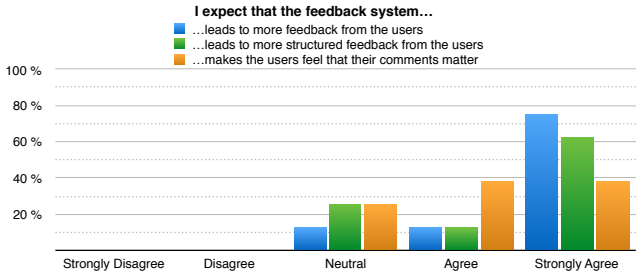

Figure 5: Developers' opinion about the implications of using CAFE

were involved, as they were already using an issue tracker and integrating the system would not lead to a large organizational overhead.

All interviewees strongly agreed that they benefited from the attached usage context information. Two of the developers mentioned the example of one particular feedback item containing a simple question "What does the switch do?", which they could not have understood without the attached information of the screen which the feedback came from.

As shown in Figure 5, 75% of the participating developers agreed or strongly agreed that CAFE would lead to more feedback from users, and over half of them agreed that CAFE would also lead to more structured feedback. The majority reasoned that this was due to the simplicity of the feedback process. While they agreed that CAFE would increase user involvement, they made this dependent on the way the development team communicated with the users. If they reacted to feedback accordingly, user involvement and satisfaction would rise, otherwise the feedback system would have no impact on user involvement.

4. DISCUSSION

In this section, we discuss the implications of our work for both users and developers as well as its current limitations.

4.1 Implications

The evaluation results confirm part of our research hypotheses. In particular, we were able to gather in-depth information on the usage of CAFE on the user side through the controlled experiments and the analysis of the questionnaire data. We cannot yet confirm **H1.1** because we cannot determine whether including CAFE in the application led to more feedback, considering that at the time of our evaluation, the Conada development team had not yet collected feedback through other channels. This hypothesis has to be evaluated in a long-term setting, preferably by following the amount of feedback over several consecutive releases of an application after introducing CAFE. However, the fact that the interviewed developers view CAFE to be beneficial for user involvement provides first anecdotal evidence for **H1**.

Regarding **H1.2**, we discovered that there was a strong effect to the usage of CAFE's features when users are informed about the functionality of CAFE to attach additional data to feedback. This indicates that users who know about this functionality are willing to specify this information, while participants of our uninformed group did not seem to find these features. Therefore, future versions of CAFE will include adjustments of the user interface to counter this effect.

The interviewed members of the development team unanimously agreed that the collected context information was useful when working with feedback, which provides anecdotal evidence for **H2.1**. The results along with the developers' comments also validate our choice to save feedback directly in the issue tracker as opposed to an external database (**H2.2**).

4.2 Limitations

While the results of our first evaluation support our hypotheses, there are certain limitations of this study. With regard to the user experiments, our sample was homogeneous concerning their knowledge in application development and user feedback. This along with the fact that only 15 people participated in our experiments limits the generalizability of our results.

In addition to this external threat, the internal validity is influenced by the controlled setting of the experiments. Our participants claimed that they rarely give feedback, but our study actively instructed them to do so. Thus, it is possible that they like CAFE simply because they have rarely used other feedback channels and have not experienced their advantages. However, we believe that every participant had used other feedback channels at least once and could thus compare it to the experience of CAFE.

The validity of the developer interviews is limited by the fact that the developers of Conada could only use the received feedback in one iteration. The evaluation should be repeated after the participants have spent more time using CAFE for development in order to get a more meaningful response. The fact that the project took place in a university setting can also be regarded as a limitation, although the nature of the capstone course is as real as possible in a student project, and 75 % of our participants characterized themselves as semi-professional developers.

5. CONCLUSION AND FUTURE WORK

CAFE addresses several challenges that both users and developers face in the current practices of providing and analyzing user feedback. The system integrates an easy way for users to give feedback without leaving the currently used application. By storing user feedback in the issue tracking system, it simplifies feedback collection, and makes it easier for developers to consider feedback regularly because they can analyze it in the system which they already use for their daily tasks. Moreover, CAFE enriches the user feedback with automatically collected information about usage context such as the last interaction steps, and gives the user the possibility to structure his feedback, e.g. by specifying the feedback type. This helps to close the information gap between users and developers.

CAFE goes further than other tools available by representing the connection between requirements and feedback, which we call feedback traceability. User feedback can be both the consequence of an implemented requirement, and the source for new requirements, a connection which is represented in CAFE as links in the issue tracking system. This offers an integrated overview over the evolution of requirements in the software development process in the same system, which we consider a substantial benefit of CAFE.

A first evaluation provides anecdotal evidence regarding the impact of CAFE on users and developers. Users reported CAFE to be simple to use and are willing to specify further

details to their text-based feedback if they are aware of the possibility. Developers highlighted the fact that feedback is stored in the issue tracker as particularly beneficial. They further reported that both the automatically collected usage context as well as the additional data specified by users such as the feedback type helped them to analyze the feedback and extract the underlying requirement.

We are currently conducting a second long term evaluation with an AppStore application to obtain results in a setting with a larger user base, a more experienced development team and objective measurements for efficiency. In particular, we investigate whether developers consider user feedback more regularly with CAFE, and how our system impacts their efficiency of integrating feedback based on process metrics such as time spent on feedback analysis. We also want to analyze the value of feedback traceability by studying the impact of CAFE on rationale management. In the long term, we want to gain insight into whether CAFE has an impact on user satisfaction by improving user involvement through examining the AppStore ratings of the application over consecutive releases.

6. REFERENCES

[1] M. Almaliki, C. Ncube, and R. Ali. The design of adaptive acquisition of users feedback: An empirical study. In *RCIS '14*, pages 1–12. IEEE, 2014.

[2] M. Bano and D. Zowghi. User involvement in software development and system success: a systematic literature review. In *EASE '13*, pages 125–130. ACM, 2013.

[3] D. Bertram, A. Voida, S. Greenberg, and R. Walker. Communication, collaboration, and bugs. In *CSCW '10*, page 291. ACM, 2010.

[4] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In *SIGSOFT '08*, page 308. ACM, 2008.

[5] B. Bruegge, S. Krusche, and L. Alperowitz. Software engineering project courses with industrial clients. *ACM Transactions on Computing Education*, 15(4):17:1–17:31, 2015.

[6] L. Damodaran. User involvement in the systems design process - a practical guide for users. *Behaviour and Information Technology*, 15:363–377, 1996.

[7] M. De Sá and L. Carriço. Designing and evaluating mobile interaction: challenges and trends. *Foundations and Trends in Human-Computer Interaction*, 4(3):175–243, 2011.

[8] A. Egyed and P. Grunbacher. Automating requirements traceability: Beyond the record & replay paradigm. In *ASE '02*, pages 163–171. IEEE, 2002.

[9] O. Gotel and A. Finkelstein. An analysis of the requirements traceability problem. In *RE '94*, pages 94–101. IEEE, 1994.

[10] J. Heiskari and L. Lehtola. Investigating the state of user involvement in practice. *APSEC '09*, pages 433–440, 2009.

[11] K. Holtzblatt. Designing for the mobile device: Experiences, challenges, and methods. *Communications of the ACM*, 48(7):32–35, 2005.

[12] S. Krusche and L. Alperowitz. Introduction of Continuous Delivery in Multi-Customer Project Courses. In *Companion Proceedings of the 36th ICSE*, pages 335–343. IEEE, 2014.

[13] S. Krusche, L. Alperowitz, B. Bruegge, and M. Wagner. Rugby: an agile process model based on continuous delivery. In *RCoSE '14*, pages 42–50. ACM, 2014.

[14] S. Krusche and B. Bruegge. User feedback in mobile development. In *Proceedings of the 2nd International Workshop on MobileDeLi*, pages 25–26. ACM, 2014.

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

[16] S. Kujala, M. Kauppinen, L. Lehtola, and T. Kojo. The role of user involvement in requirements quality and project success. In *RE '05*, pages 75–84. IEEE, 2005.

[17] W. Maalej, H.-J. Happel, and A. Rashid. When Users Become Collaborators: Towards Continuous and Context-Aware User Input. In *OOPSLA '09*, page 981. ACM, 2009.

[18] W. Maalej and D. Pagano. On the socialness of software. In *DASC '11*, pages 864–871. IEEE, 2011.

[19] M. Nagel, J. Helming, M. Koegel, and H. Naughton. Audio recording in software engineering. *ICSE South Africa*, 2010.

[20] D. Pagano. *Portneuf - A Framework for Continuous User Involvement*. PhD thesis, Technische Universität München, 2013.

[21] A. Panichella, A. De Lucia, and A. Zaidman. Adaptive user feedback for ir-based traceability recovery. In *SST '15, 2015*, pages 15–21. IEEE, 2015.

[22] P. Rodríguez, A. Haghighatkhah, L. Lwakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J. Verner, and M. Oivo. Continuous deployment of software intensive products and services: A systematic mapping study. *Journal of Systems and Software*, 2016.

[23] J. C. Sampaio do Prado Leite and P. A. Freeman. Requirements validation through viewpoint resolution. *IEEE Transactions on Software Engineering*, 17(12):1253–1269, 1991.

[24] K. Schneider. Focusing spontaneous feedback to support system evolution. In *RE '11*, pages 165–174. IEEE, 2011.

[25] K. Schneider, S. Meyer, M. Peters, F. Schliephacke, J. Mörschbach, and L. Aguirre. Feedback in context: Supporting the evolution of IT-ecosystems. *Lecture Notes in Computer Science*, 6156:191–205, 2010.

[26] K. Schwaber and M. Beedle. *Agile Software Development with Scrum*. Prentice Hall, New Jersey, USA, 2001.

[27] F. Yetim, S. Draxler, G. Stevens, and V. Wulf. Fostering Continuous User Participation by Embedding a Communication Support Tool in User Interfaces. *AIS Transactions on Human-Computer Interaction*, 4(2):153–168, 2012.