Continuous User Understanding for the Evolution of Interactive Systems

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

ABSTRACT

Continuous Software Engineering (CSE) activities, i.e., rapidly delivering new software functionality to software users and implementing received feedback, have became an established development practice for creating interactive systems. The frequency of software changes turns the feedback loop with users into a critical element of CSE that has not been addressed sufficiently; thus, it may be challenging for developers to understand users' software usage. This research project aims to enable a better understanding of users during CSE. We investigate relevant usage knowledge needs, the unobtrusive collection of usage data by software and hardware sensors, how usage data can be related to feature increments, and ways to externalize tacit usage knowledge. Leveraging these insights, we develop a platform to monitor, visualize, and understand usage knowledge to support developers during the design and development of interactive systems. The overall goal is to accurately fit the functionality of interactive systems to user needs.

CCS CONCEPTS

• **Information systems → Users and interactive retrieval**; • **Human-centered computing → HCI design and evaluation methods**; *Interactive systems and tools*; • **Software and its engineering → Software creation and management**; *Agile software development*; Designing software;

EICS '18, June 19–22, 2018, Paris, France

KEYWORDS

User Understanding; Usage Monitoring; User Behavior; Interactive System; Continuous Software Engineering

ACM Reference Format:
Jan Ole Johanssen. 2018. Continuous User Understanding for the Evolution of Interactive Systems. In *EICS '18: EICS '18: ACM SIGCHI Symposium on Engineering Interactive Computing Systems, June 19–22, 2018, Paris, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3220134.3220149

INTRODUCTION

The term Continuous Software Engineering (CSE) refers to various activities, such as continuous integration and continuous delivery, to rapidly iterate on software increments to facilitate continuous learning and improvement [3, 5]. The frequent delivery of functionality to users opens new possibilities and enables direct involvement of users in the software development process. Consequently, software is developed according to the users' needs.

However, CSE poses new challenges regarding the understanding of users. *First*, typically only early adopters or a subset of the development team access a new system version. Thus, the dataset obtained for analysis is sparse. *Second*, the frequent release of software increments makes qualitative feedback in the form of written reviews or interviews a less useful approach to understand how the software is being used. Generally, users tend to avoid explicit feedback because this requires additional efforts. *Third*, in consideration of interweaving development threads that are frequently deprecated by new software increments, relating implicit and explicit user feedback to the feature under development becomes a complex task. *Lastly*, developers do not have access to *tacit* knowledge that is hidden in user interactions and may become relevant for the development process. To illustrate the relationship between CSE and user understanding, consider the scenario in Sidebar 1 on the left side.

Scenario — A mobile navigation application provides a function to select different map views, such as terrain or satellite views. The individual views are selected by a menu item in the application's sidebar. As a feature improvement, this function is implemented as a dedicated button that is permanently visible. Here, the overall intention is to facilitate the change between map views. However, due to the change, users may no longer find the feature and consequently stop using it, perhaps even without their knowledge. This reduces the chances that users will report the issue, thereby potentially making the problem less detectable. In general, feature usage declines due to an unfavorable change in a software iteration; a situation that might be detectable. Developers require such information to support decisions and evaluate changes introduced in incremental software development.

Sidebar 1: A scenario that highlights the importance of user understanding during software development.

In summary, even though usage knowledge represents a key success factor for software development, compared to CSE activities such as continuous integration or continuous delivery, the activity of understanding users has not reached a similar maturity level. Therefore, we are investigating solutions to close the gap in the continuous feedback loop.

RELATED WORK

Users and their interactions, i.e., how they apply software, form the core of an active research area: for example, Röhm introduced the MALTASE framework to follow user interactions and help developers utilize the resulting information [12]. Pagano proposed the PORTNEUF framework to encourage user feedback and provided a method to assess the importance of individual feedback [11]. Similar to the work of Röhm and Pagano, we focus on a platform to support developers.

Several researchers have investigated the automatic generation of personas based on user interactions or social media data [10, 14]. Almeida *et al.* acknowledged a need for the human computer interaction community to consider usability smells, which are similar to code smells [6], to increase quality and maintainability of interactive applications [1]. Following this idea of *smells*, we address the problem from a user perspective, which is different from Almeida *et al.*'s approach, who described a developer-driven perspective. In other words, we attempt to derive *behavioral smells* that help us understand how users perceive the design and development of an interactive system.

Understanding users is an important source of information; existing approaches from other domains might be applied to software engineering—and vice versa. For example, Banovic highlighted how large amounts of behavior log data can be used to understand complex human routines [2].

PROBLEM STATEMENT

CSE activities, such as continuous delivery, have transformed software evolution in a rapidly iterating process of developing new functionality and integrating feedback. However, the maturity level of user understanding is not keeping pace with recently evolving CSE activities. This has created a demand for development support in CSE environments, i.e., the need for a *continuous user understanding activity*.

RESEARCH QUESTIONS

To investigate user understanding in the CSE context, we aim to answer four research questions:

RQ1 *What is relevant usage knowledge that stakeholders lack about users and their interactions?*

Understanding usage knowledge needs represents the foundation of this work. Answers to this question identify actual usage knowledge needs and strengthen ideas about how to integrate a continuous user understanding activity into CSE activities.

RQ2 *How to enable the unobtrusive collection of usage data from the silent majority of users?*

Most users refrain from providing explicit feedback; thus, we want to explore various approaches to collect *implicit* user feedback by utilizing a variety of software and hardware sensors.

RQ3 *How to relate usage data to features or individual software increments?*

After usage data are collected, the data must to be linked to the feature under development. We explore various ways to establish such links, e.g., commit-based references or more fine-grained approaches on code level.

RQ4 *How to externalize tacit usage knowledge?*

By bringing together usage knowledge and other context information about the application, we want to determine to which extent we can extract tacit knowledge, i.e., knowledge that is deeply ingrained in the mind of users and difficult to verbalize.

RESEARCH APPROACH

An on-going literature review is being performed to answer the research questions, in particular **RQ1**. This strengthens our understanding of how the user understanding domain is currently approached. Since CSE is also heavily influenced by industry, we are continuously analyzing available tools on the market in terms of their functionalities and the way they integrate with existing CSE activities.

Furthermore, we have performed an extensive empirical study with 24 industry practitioners regarding different aspects of CSE. One of our goals was to better understand practitioners' needs for user understanding and derive requirements for a platform to enable continuous user understanding.

To explore the full spectrum of user understanding, we are working on a prototypical implementation of a platform for continuous user understanding. To evaluate the platform and assess theoretical concepts, we are planning to perform multiple case studies in industry settings.

RESULTS TO DATE

The analysis of practitioners' knowledge needs has formed a major element of our research to date, establishing a basis for continuous user understanding. Therefore, regarding **RQ1**, we published the initial results of the empirical study with practitioners [9] and plan to evaluate the remaining parts of the interviews in the future. The study's results point to interesting findings in currently applied approaches as well as needs for future additions to CSE activities.

To explore the unobtrusive collection of usage data as part of **RQ2**, we investigated human cognitive load during knowledge work measured by consumer wearable sensors [13]. Although this research was performed in smart environments, a setting apart from software engineering, this reflects a first step in understanding users based solely on implicit data. The gained insight will be transformed into CSE practices as one of our next steps.

To address **RQ3**, we explored ideas about how usage knowledge can be combined with other types of knowledge [7], and described a proposal for usage knowledge visualization in the CSE context using a knowledge dashboard and widgets reflects our first contribution to answering **RQ4** [8].

CURRENT STATUS AND NEXT STEPS

After clearing up the CSE setting and knowledge needs, we will enter the pivotal part of this research project, i.e., continuously understanding users in the context of software evolution.

In practice, this involves the extraction of information from usage data, e.g., automatically classifying users into different groups, and system-related information, such as usability problems, based solely on implicit data, such as taps. We also strive to integrate other aspects from the affective computing domain, since we have obtained promising results for involving user emotions. However, this aspect requires more work to become a valuable contribution relative to addressing **RQ2** and **RQ3**.

Currently, we are working on an implementation of the platform on a prototypical basis to evaluate the concepts and explore user behavior in CSE settings. We refer to this as the Continuous User Understanding plattform (CUU, pronounced *"see you"*). CUU enables developers to inspect usage knowledge and thereby addresses **RQ2** to **RQ4**. The implementation follows an incremental approach. A first major version of CUU was developed in recent months and will be ready to function in multiple real-world industry settings in the 2018 summer term: We will deploy CUU in our capstone course [4], in which up to 100 students work in development teams on problems posed by industrial partners, which we consider to be an ideal testbed. Based on the insights derived from this evaluation, we will continue to implement a second major iteration, incorporate lessons-learned, followed by a potential second deployment in the capstone course during the 2018/19 winter term.

EXPECTED CONTRIBUTIONS

The expected contributions of this research are diverse. Some build on each other, given the structure of the research questions. For example, **RQ1** forms a foundation for **RQ2** to **RQ4**.

First, a formalized process of integrating usage knowledge into a CSE pipeline will be presented and made available for integration in an applied setting. Thereby, the process loop of committing code followed by automatically testing and delivering software increments is being completed with a continuous user understanding activity.

Second, the CUU platform including a meta model of how to integrate and combine it with other CSE processes, e.g., continuous delivery, is being developed on a prototypical basis. CUU is expected to facilitate the understanding of user behavior between software builds, which is enabled by visual representation of different knowledge types. Further, an evaluation and assessment of CUU's functionality will reflect an important contribution to validate its benefits.

Third, a set of so-called *user anti-patterns* that describe typical user behavior during interactive system usage will be developed. These user anti-patterns should allow a simplified understanding of users aiming at increased comprehension of software evolution, thereby resulting in fast iterations with target-oriented improvements. Grouped and organized contextually, these user anti-patterns will support developers reactions to changes in feature usage by users. The user anti-patterns can be applied by software engineers and other domains experts.

This research project started two years ago. We expect to finish developing the CUU platform over the course of this year and to complete this phase with an analysis and assessment of the results.

ACKNOWLEDGMENTS

I thank my supervisor Professor Bernd Bruegge as well as Professor Barbara Paech and Anja Kleebaum. This research is supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future – Managed Software Evolution (CURES project).

REFERENCES

[1] Diogo Almeida, José Creissac Campos, João Saraiva, and João Carlos Silva. 2015. Towards a Catalog of Usability Smells. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC '15)*. ACM, 175–181. https://doi.org/10.1145/2695664.2695670

[2] Nikola Banovic. 2017. Method for Understanding Complex Human Routine Behaviors from Large Behavior Logs. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17)*. ACM, 254–258. https://doi.org/10.1145/3027063.3027135

[3] Jan Bosch. 2014. *Continuous Software Engineering: An Introduction*. Springer International Publishing, Cham, 3–13. https://doi.org/10.1007/978-3-319-11283-1_1

[4] Bernd Bruegge, Stephan Krusche, and Lukas Alperowitz. 2015. Software Engineering Project Courses with Industrial Clients. *ACM Transactions on Computing Education* 15, 4, Article 17 (Dec. 2015), 31 pages. https://doi.org/10.1145/2732155

[5] Brian Fitzgerald and Klaas-Jan Stol. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123 (2017), 176 – 189. https://doi.org/10.1016/j.jss.2015.06.063

[6] Martin Fowler and Kent Beck. 1999. *Refactoring: improving the design of existing code*. Addison-Wesley Professional.

[7] Jan Ole Johanssen, Anja Kleebaum, Bernd Bruegge, and Barbara Paech. 2017. Towards a Systematic Approach to Integrate Usage and Decision Knowledge in Continuous Software Engineering. In *Proceedings of the 2nd Workshop on Continuous Software Engineering*. 7–11.

[8] Jan Ole Johanssen, Anja Kleebaum, Bernd Bruegge, and Barbara Paech. 2017. Towards the Visualization of Usage and Decision Knowledge in Continuous Software Engineering. In *2017 IEEE Working Conference on Software Visualization (VISSOFT)*. 104–108. https://doi.org/10.1109/VISSOFT.2017.18

[9] Jan Ole Johanssen, Anja Kleebaum, Barbara Paech, and Bernd Bruegge. 2018. Practitioners' Eye on Continuous Software Engineering: An Interview Study. In *Proceedings of the International Conference on Software and System Processes*. Accepted for publication (May 2018).

[10] Soon-Gyo Jung, Jisun An, Haewoon Kwak, Moeed Ahmad, Lene Nielsen, and Bernard J. Jansen. 2017. Persona Generation from Aggregated Social Media Data. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17)*. ACM, 1748–1755. https://doi.org/10.1145/3027063.3053120

[11] Dennis Pagano. 2013. *Portneuf - A Framework for Continuous User Involvement*. Dissertation. Technical University of Munich.

[12] Tobias Röhm. 2015. *The MALTASE Framework For Usage-Aware Software Evolution*. Dissertation. Technical University of Munich.

[13] Florian Schaule, Jan Ole Johanssen, Bernd Bruegge, and Vivian Loftness. 2018. Employing Consumer Wearables to Detect Office Workers' Cognitive Load for Interruption Management. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 1, Article 32 (March 2018), 20 pages. https://doi.org/10.1145/3191764

[14] Xiang Zhang, Hans-Frederick Brown, and Anil Shankar. 2016. Data-driven Personas: Constructing Archetypal Users with Clickstreams and User Telemetry. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, 5350–5359. https://doi.org/10.1145/2858036.2858523