End-user Driven Feedback Prioritization

Norbert Seyff[1,2], Melanie Stade[1], Farnaz Fotrousi[1,3], Martin Glinz[2], Emitza Guzman[2], Martina Kolpondinos-Huber[2,4], Denisse Munante Arzapalo[5], Marc Oriol[6], and Ronnie Schaniel[1]

[1] University of Applied Sciences and Arts Northwestern Switzerland, Switzerland
[2] University of Zurich, Switzerland
[3] Blekinge Institute of Technology, Sweden
[4] Swiss Federal Laboratories for Materials Science and Technology (Empa), Switzerland
[5] Fondazione Bruno Kessler, Italy
[6] Universitat Politècnica de Catalunya, Spain
norbert.seyff@fhnw.ch

Abstract. End-user feedback is becoming more important for the evolution of software systems. There exist various communication channels for end-users (app stores, social networks) which allow them to express their experiences and requirements regarding a software application. End-users communicate a large amount of feedback via these channels which leads to open issues regarding the use of end-user feedback for software development, maintenance and evolution. This includes investigating how to identify relevant feedback scattered across different feedback channels and how to determine the priority of the feedback issues communicated. In this research preview paper, we discuss ideas for end-user driven feedback prioritization.

Keywords: Requirements prioritization · End-user involvement · End-user feedback

1 Introduction

Building software systems which reflect the needs of end-users is crucial for their success. Hence it is important to gather end-user requirements and to include them in requirements prioritization and release planning activities. However, involving a *large number of end-users* in requirements elicitation activities can be cumbersome using traditional requirements elicitation methods (e.g., interviews), also because end-users are often out of organizational reach [1]. In such settings, end-user feedback gathering is one way that allows software companies to elicit and consider end-users' experiences and requirements for software development, maintenance and evolution activities. For this purpose, companies can make use of feedback communication channels such as app stores (e.g., Google Play, Apple's App Store), social networks

(e.g., Facebook, Twitter) and built-in feedback mechanisms which are part of the software application itself.

However, there are several issues regarding the elicitation and analysis of end-user feedback. Most of the feedback channels available (e.g., Twitter) *do not focus exclusively on the communication of feedback* relevant to software development, maintenance and evolution: they also allow for a *discussion* on various topics regarding a software application [2]. Therefore, identifying relevant feedback can be difficult in case a large amount of data is available and cannot be manually conducted within a reasonable amount of time and effort. Furthermore, feedback communicated using *natural language text*, does not follow any predefined structure or template. It is more the informal description of an issue or need than a formalized requirement. Some feedback channels also allow the documentation of *multi-modal* feedback descriptions which include screenshots and audio recordings. The lack of structure and the use of multi-modal descriptions can have a negative effect on the analysis of the feedback issue and can complicate the definition of a requirement [3], whether it is done manually or with the help of (semi-)automatic approaches.

Another critical issue is *trust* [4]. Most channels invite everybody to contribute and communicate feedback. It means that feedback issues can also be communicated from end-users with low reputation and might be misleading, not representing the opinion of the majority of end-users. In the worst case, some end-users might even negatively affect the evolution of a system by exaggerating or reporting wrong issues. These problems sometimes make it hard for a software company to decide which feedback issues actually are of high priority and should be considered in a next release of their software application.

The goal of our work is to provide approaches which go *beyond the elicitation of end-user feedback* and also establish an order of priority for the feedback issues gathered. We want to achieve this by involving end-users in the prioritization of feedback issues and by realizing (automated) feedback analysis approaches to generate priorities for feedback issues. This will, for example, allow software companies to generate adequate release plans [5].

2 Approaches for End-user Driven Feedback Prioritization

Our ongoing research within the EU project SUPERSEDE has revealed that there exist feedback channels (e.g., social networks) where a large number of feedback is communicated. Investigating end-user feedback communicated via Twitter, we have learned that the manual analysis of this feedback by software development companies is not feasible, because it would consume too much time and effort [2]. Hence, involving end-users in this task and making use of automated approaches for feedback analysis is needed to support the identification and prioritization of relevant feedback. In the following paragraphs we present several ideas the authors are working on to

allow for and support *End-user Driven Feedback Prioritization*. We structure these ideas by looking at *feedback gathering*, *feedback analysis* and the inclusion of end-user feedback in *decision making*.

2.1 Prioritization within Feedback Gathering

Elicitation of end-user feedback is a first crucial step for end-user involvement in software development, maintenance and evolution. There exists a broad range of methods and tools which allow end-users to communicate their needs (e.g., [6]). The following ideas allow prioritization within feedback gathering and are currently explored by the authors.

Feedback Approaches Supporting the Communication of Priorities. There exist feedback communication channels which provide means for end-users to go beyond text-based feedback descriptions and to communicate a priority. This priority can be expressed, for example, by using *ratings* and it might cover different interpretations of what is actually meant by priority (e.g., importance for the individual end-user vs. importance for application success). Furthermore, some of these channels also allow end-users to specify the type of feedback they are communicating. For example, this could include a *selection* to define if a feedback issue is a "bug report" or a "feature request". Communicating a priority should be possible with little effort to avoid that end-users cancel the feedback communication. Allowing end-users to also communicate a priority regarding their request is a first important step towards End-user Driven Feedback Prioritization. However, such priorities need to be looked at with care, as end-users might have their own *interpretation* of what they actually want to communicate via the priority field; they often will just express their subjective opinion. It is essential to validate such priorities in order to gain trust.

Feedback Approaches Supporting the Discussion of Feedback amongst End-users. Some feedback channels allow end-users to discuss their feedback with peers. This includes, for example, social networks that enable end-users to communicate their agreement or disagreement (e.g., by using likes) regarding the feedback issues which have been gathered [7]. An example for a forum that focuses on requirements elicitation and prioritization is *Garuso* (*G*ame-b*a*sed *R*equ*i*rement*s* elicitati*o*n). It enables stakeholders outside organizational reach to communicate, discuss and rate their ideas over a forum-like online platform. Garuso makes use of gamification elements to motivate end-users to participate which was explored in previous research by the authors [8]. We consider approaches such as Garuso to allow for an early validation of relevance and priority of end-user feedback issues. However, ratings of end-users, for example expressed with "I like", might be ambiguous as it might not be clear what the end-user providing the rating is actually fond of. Apart from

commenting on the content, a like could also discuss the representation style of a feedback issue or simply show that one end-user is friendly with another one. To avoid such ambiguity, Garuso proposes a two-dimensional mechanism which covers grouping based on *relevance* (irrelevant, neutral, and relevant) and rating based on *popularity* (dislike, neutral, and like). Further, to avoid false contributions, i.e., grouping and rating of feedback issues only to earn points, Garuso offers the possibility to deliberately not taking any decision by selecting *no decision* (which gives the same amount of points to the end-users as if they grouped and rated a feedback issue).

2.2 Prioritization with the Help of Feedback Analysis

End-user feedback needs to be analyzed. Ideally there are automated methods and tools in place which support the analysis of the feedback gathered. We discuss how analysis can be used to generate priorities for end-user feedback issues.

Number of Feedback Issues Discussing the Same Issue. One possible option to analyses feedback issues is to *group* them and build *clusters* of end-user concerns. Obviously, the number of feedbacks within a cluster can help to indicate a priority. This could also mean that feedback from *different channels* is analyzed and clustered.

Sentiment of the Feedback Issues. We use the term sentiment to refer to the *feeling or view* reflected in a feedback text. A text could express a *negative* (e.g., "I hate uploading files with this app!") or *positive* (e.g., "The performance of this app is amazing!") sentiment. More negative sentiments could express higher end-user dissatisfaction and could therefore indicate a higher resolution urgency [9].

Type of Feedback Issues. The type of a feedback issue (e.g. bug report, feature request, general praise) might be explicitly communicated by end-users but there also exist automated approaches to do so [10] [11]. We assume that in most cases *certain* feedback types (e.g., bug report) might be more critical to be resolved and therefore have a higher priority than other issues (e.g., feature request).

Monitoring Data and Feedback Issues. Several other informations can help to automatically define or tailor the priority of a feedback issue. For example, there might be situations where one single end-user reports an issue (e.g., "My video stream is constantly buffering"). However, with the help of monitoring data, it is possible to analyze the *dimension* of this problem and detect if other end-users also experience a similar problem. This can help to prioritize and address the problem. Another example of the benefits of monitoring data is that monitoring can also be used to identify *how the software is being used* and detect which are the most used and

important functionalities. This information can be helpful to better understand the priority of incoming feedback. In case it addresses a widely and intensively used functionality, the feedback might be of higher priority.

2.3 Prioritization and Decision Making within the Software Company

The requirements prioritization decision making is a crucial activity in software development [12]. These decisions often involve different stakeholders including representatives of end-users, managers, and developers [13] and they are often based on different criteria such as development effort, user impact, costs, resource constraints. Several decision making techniques have been proposed to support requirements prioritization [14]. In the following paragraphs, we discuss ideas on how to involve end-users in the decision making process.

Other Requirements and Feedback Issues. In many software development organizations feedback issues might not be the only source for gathering requirements. In case feedback issues and requirements from *other sources* (e.g., interviews or workshops with different stakeholder groups) discuss the same need as incoming feedback issues, this indicates a high priority, in particular when different stakeholder groups are involved. Furthermore, it is important to define to what *extent* a specific source influences the overall priority of a requirement by e.g. defining weights for each source that contributes to a requirement. We are also envisioning mechanisms that assess the quality of the incoming feedback and generate a reputation score for particular end-users. Eventually this can also be used as weighting mechanisms which gives a higher priority to feedback from trusted senders with a solid reputation.

Validation of End-user Feedback Priority. Even though we have identified several ideas on how to determine the priority of a feedback issue, decision makers will need to be involved to finally decide on the priority of a feedback issue. We foresee that a high number of evidences on a feedback's priority might be helpful for decision makers to make a final decision. Therefore, we recommend that a decision maker (e.g., a product owner) has *access* to and can *visualize* end-user driven feedback prioritization derived from different sources. Furthermore, feedback issues can be used as input for other requirements elicitation and negotiation activities in order to be discussed with other stakeholders to ensure their validity.

Validation of Requirements from other Sources with the Help of Feedback Channels. We also foresee that feedback channels will be used to validate requirements from other sources. For example, in case the Product Owner is unsure about some of her ideas and the importance of these ideas, she can *ask end-users for*

their priority (e.g., via a new discussion thread in an online forum or a popup-window within an application).

3 Conclusion and Next Steps

End-user feedback is becoming more important for the evolution of software systems. However, it is crucial to better understand the importance and relevance of end-user feedback. Prioritized end-user feedback can help to build software systems which are widely accepted by end-users and to make the software evolution process more efficient.

In this paper, we presented first ideas on how to engage end-users in feedback prioritization. Furthermore, we explored (automated) analysis approaches which exploit information gathered from end-users to prioritize their feedback. We are currently realizing the bespoke prioritization approaches within the SUPERSEDE EU project. Next steps will include an investigation into the consequences of involving end-users in feedback prioritization. We expect that transparent prioritization approaches will help end-users to understand why a feedback issue is eventually considered for implementation and will help to continuously engage end-users in feedback gathering and prioritization activities.

4 Acknowledgment

The research described in the paper was in part funded by the EU project SUPERSEDE (grant agreement no 644018).

References

1. Todoran, I., Seyff, N., Glinz, M. (2013) How cloud providers elicit consumer requirements: An exploratory study of nineteen companies. In: International Requirements Engineering Conference , pp. 105–114.
2. Guzman, E., Alkadhi, R., Seyff, N. (2016) A Needle in a Haystack: What Do Twitter Users Say about Software?. In: International Requirements Engineering Conference, pp. 96–105.
3. Gärtner, S., Schneider, K. (2012) A method for prioritizing end-user feedback for requirements engineering. In: International Workshop on Co-operative and Human Aspects of Software Engineering, pp. 47–49.
4. Dalpiaz, F. (2011) Social threats and the new challenges for Requirements Engineering. In: International Workshop on Requirements Engineering for Social Computing, pp. 22–25.
5. Greer, D., Ruhe, G. (2004) Software release planning: an evolutionary and iterative approach. Information and Software Technology 46(4), pp. 243–253.
6. Seyff, N., Ollmann, G., Bortenschlager, M. (2014) AppEcho: a user-driven, in situ

feedback approach for mobile platforms and applications. In: MOBILESoft, pp. 99–108.
7. Seyff, N., Todoran, I., Caluser, K., Singer, L., Glinz, M. (2015) Using popular social network sites to support requirements elicitation, prioritization and negotiation. J. Internet Services and Applications 6(1), 7:1–7:16.
8. Huber, M. Z., Hilty, L.M. (2015) Gamification and Sustainable Consumption: Overcoming the Limitations of Persuasive Technologies. In: ICT Innovations for Sustainability, pp. 367–385.
9. Chen, N., Lin, J., Hoi, S. Ch., Xiao, X., Zhang, B. (2014) AR-miner: mining informative reviews for developers from mobile app marketplace. In: International Conference on Software Engineering, pp. 767–778.
10. Guzman, E., El-Haliby, M., Bruegge, B. (2015) Ensemble methods for app review classification: An approach for software evolution. In: Automated Software Engineering (ASE), pp. 771–776.
11. Panichella, S., Di Sorbo, A., Guzman, E., Vissagio, C. A., Canfora, G., Gall, H. (2015) How can I improve my app? classifying user reviews for software maintenance and evolution. In: Software Maintenance and Evolution (ICSME), pp. 281–290.
12. Firesmith, D. (2004) Prioritizing requirements. Journal of Object Technology 3(8), pp. 35–48.
13. Ruhe, G., Saliu, M. (2005) The art and science of software release planning. IEEE Software 22(6), pp. 47–53.
14. Achimugu, P., Selamat, A., Ibrahim, R., Mahrin, M.N. (2014) A systematic literature review of software requirements prioritization research. Inf. Softw. Technol. 56(6), pp 568–585.