Data-Driven Requirements Engineering – An Update

Walid Maalej
University of Hamburg
maalej@informatik.uni-hamburg.de

Maleknaz Nayebi
Ecole Polytechnique Montreal
mnayebi@polymtl.ca

Guenther Ruhe
University of Calgary
ruhe@ucalgary.ca

***Abstract*—Nowadays, users can easily submit feedback about software products in app stores, social media, or user groups. Moreover, software vendors are collecting massive amounts of implicit feedback in the form of usage data, error logs, and sensor data. These trends suggest a shift toward data-driven user-centered identification, prioritization, and management of software requirements. Developers should be able to adopt the requirements of masses of users when deciding what to develop and when to release. They could systematically use explicit and implicit user data in an aggregated form to support requirements decisions. In this talk we will present and discuss most recent achievements in this direction since the paper's original publication. We will also show to mine data sets mobile apps, give a few success/failure stories and a few practical advises.**

***Index Terms*—Requirements engineering, Data analytics, Mining software repositories, Stakeholders, Feature extraction**

I. Motivation and Context

Requirements engineering identifies, documents, negotiates, and manages the desired properties and constraints of software-intensive systems, the goals to be achieved in a software project, and the assumptions about the environment [7]. We can consider requirements as a verbalization of decision alternatives regarding the functionality and quality of a system [1]. A large part of requirements engineering is concerned with involving system users, capturing their needs, and getting their feedback.

Since its original publication [7], an increasing number of studies showed that the predicted direction is relevant. In the European Horizon2020 project Q-Rapid [4], a data-driven, quality-aware rapid software development framework was developed in which quality and functional requirements are managed in conjunction. As another example, social media support for deciding on new or updated features is studied [10]. It was shown that app store mining is an important source of information, but should be complemented by other sources such as tweets to provide the best and most up-to-date input for decision-making.

The notion of platform-mediated software engineering on two sided-markets (as in mobile apps stores) closed the feedback loop to users and added transparency to the market status of software products. As of today, the "market" has significant impact on the release decisions. Release engineers are concerned with the factors to evaluate success in these two-sided markets [11]. A survey with release engineers showed that 90% believed that *customer feedback* has the highest importance for evaluating success and failure of mobile apps [11]. This imposes the importance of understanding user requirements trough the public feedback loops and in a more systematic way.

In the original article as well as in this talk, we discuss how software developers, managers, and analysts can use this data to identify, prioritize, and manage the requirements for their software products. We see three major future directions in practice.

First, tools for feedback analytics will help deal with a large number of heterogeneous and unstructured user comments by classifying, filtering, and summarizing them.

Second, automatically collected usage data, logs, and interaction traces could improve the feedback quality and assist developers with understanding the feedback and reacting to it. We call this automatically collected information about the software usage implicit feedback.

Third, with all the explicit and implicit feedback now available in an (almost) continuous way, the following question arises: How can practitioners use this information and integrate it into their processes and tools to decide about what should be done, e.g. when the next release should be offered or what requirements and features should be added or eliminated [7].

II. Highlights of the Talk

The rapid growth of app stores and repositories has led to an increasing number of publications in app store analysis through data mining. A big portion of these studies is concerned with analyzing and prioritizing user requirements and planning for app releases [8]. Having synthesized these studies, our experience with industry, and lessons learned in the working tutorial[1], this talk will cover the following topics:

Identifying and classifying stakeholders: Analytics can help to identify and classify various user groups, stakeholders, or personas with different characteristics and needs. For instance, deviant users and exceptional use cases can be of interest to requirements analysts and developers. Classifying stakeholders might also help analysts to identify, understand, and specify non-functional requirements more concretely, such as accessibility (by identifying how people with disabilities

[1] https://sites.google.com/view/mining2planning/home

use the software), privacy (by identifying acceptable trade-offs for stakeholders), or performance (by identifying which performance requirements apply for which user groups) [6].

Classifying and summarizing user feedback: User feedback includes a variety of information as shown by Pagano and Maalej [13]. Automatically classifying feedback can give an overall idea about an apps usage and types of user engagement (i.e. how many bug reports, feature requests, etc.). This could be used for comparing the releases over time (in terms of sentiments, requested requirements, reported bugs) and with other apps [5]. Researchers also suggested probabilistic approaches to summarize informative review content [3].

Implicit feedback analysis: Research has shown that developers and analysts can hardly use user feedback, in particular negative feedback, since it lacks context [2], [13]. To improve the understanding of the circumstances under which users submit their feedback, usage data and the interaction history can be very useful. Similar to explicit feedback, usage data need to be analyzed, filtered, summarized, and visualized to add value for developers and analysts. Implicit usage data might include the individual clicks of the users and the interactions with the user interface. We elaborate on techniques that can be used in practice to support the decisions of developers and requirements analysts.

Designing super apps by looking beyond the fence: Functionality of software products often does not match user needs and expectations. The closed set-up of systems and information is replaced by wide access to data of users and competitor products. This shift offers completely new opportunities to approach requirements elicitation and subsequent planning of software functionality. Having wide accessibility to the information of similar apps in conjunction with their user reviews and ratings can be a great source to identify the user requirements for a new app and later to compose new market-driven apps [12].

The changing process of RE Decision-making: Requirements engineering eamins a centric process, but the decisions to be made and the process how they are made have changed and will further change. Overall, we project a transition from single person and intuition based decision-making to a group-based process with decisions that are based on real-time analysis of a broad range of information sources. These changes relate to (i) how decisions are made and based on which information, (ii) who is making them, (iii) what is decided about, (iv) when to make them. There is no ”one size fits all solution” that should be expected, but a trend to openness op the whole process, which is aligned with the general trend to *open innovation in software engineering* [9].

III. Presenters

WALID MAALEJ is a professor of informatics at the University of Hamburg, chair for Applied Software Technology, and a member of the tech-transfer institute HITeC. His research interests include data-driven software engineering, context-aware adaptive systems, e-participation and crowd-sourcing, and software engineering's impact on society. Maalej received a PhD from the Technical University of Munich. He served as Program Chair of RE'18 and Industry Co-Chair for RE'17 and RE'10. Currently he is leading a 5MEUR project on Requirements Engineering and Data Science.

MALEKNAZ NAYEBI is holding an excellence research chair as an assistant professor at the University of Montreal. Maleknaz was a Postdoctoral fellow at the University of Toronto. She got her PhD at the Software Engineering Decision Support lab from The University of Calgary. The PhD was on *Analytical Release Management for Mobile Apps*. She has six years of professional software engineering experience. Her main research interests are in mining software repositories, release engineering, open innovation and empirical software engineering. Maleknaz co-chaired RE data track 2018, IWSPM 2018, IASESE 2018 advanced school, and OISE 2015.

GUENTHER RUHE is the Industrial Research Chair in Software Engineering at the University of Calgary. His research focuses on product release planning, software project management, decision support, data analytics, empirical software engineering, and search-based software engineering. Ruhe has a proven track record of working with industry and transfer technologies to practice. Hes the editor in chief of Information and Software Technology and was the General Chair of the Requirements Engineering conference RE'18. Hes a senior member of IEEE and a member of ACM.

References

[1] A. Aurum and C. Wohlin. The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology*, 45(14), 2003.

[2] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In *Proc. FSE*, pages 308–318. ACM, 2008.

[3] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Proc. RE*. IEEE, 2014.

[4] L. Guzmán, M. Oriol, P. Rodríguez, X. Franch, A. Jedlitschka, and M. Oivo. How can quality awareness support rapid software development?–a research preview. In *Proc. REFSQ*, pages 167–173. Springer, 2017.

[5] T. Johann, C. Stanik, B. M. A. AlirezaM.Alizadeh, and W. Maalej. Safe: A simple approach for feature extraction from app descriptions and app reviews. *Proc. RE*, pages 21–30, 2017.

[6] Z. Kurtanović and W. Maalej. On user rationale in software engineering. *Requirement Engineering Journal*, 23(3):357–379, Sept. 2018.

[7] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe. Toward data-driven requirements engineering. *IEEE Software*, 33(1), 2016.

[8] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman. A survey of app store analysis for software engineering. *TSE*, 43(9):817–847, 2017.

[9] H. Munir, K. Wnuk, and P. Runeson. Open innovation in software engineering: a systematic mapping study. *Empirical Software Engineering*, 21(2):684–723, 2016.

[10] M. Nayebi, H. Cho, and G. Ruhe. App store mining is not enough for app improvement. *Empirical Software Engineering*, pages 1–31, 2018.

[11] M. Nayebi, H. Farahi, and G. Ruhe. Which version should be released to app store? In *Proc. ESEM*. IEEE, 2017.

[12] M. Nayebi and G. Ruhe. Optimized functionality for super mobile apps. In *Proc. RE*. IEEE, 2017.

[13] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *Proc. RE*. IEEE, 2013.