Towards Data-Driven Requirements Engineering

Walid Maalej, Maleknaz Nayebi, Timo Johann, and Gunther Ruhe

***Abstract*—Nowadays, users can easily submit their 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. Outlining these trends, we envision a paradigm shift towards data-driven user-centered identification, prioritization, and management of software requirements. Future software and requirements engineering should be able to adopt the requirements of user masses when deciding on what to develop and when to release. We discuss how practitioners could systematically use explicit and implicit user data in an aggregated form to support their decisions about requirements – towards a data-driven requirements engineering by the masses and for the masses.**

***Index Terms*— App Reviews, Decision Support, Requirements Engineering, Software Analytics, Usage Data**

I. Introduction

Requirements Engineering is concerned with identifying, documenting, negotiating, and managing the desired properties and constraints of software-intensive systems, the goals to be achieved in a software project, and the assumptions about the environment [5]. Requirements can be considered 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.

Conventional requirements engineering typically involves users through interviews and workshops. Open source projects allow users to publicly report issues and ideas through issue trackers. Recently, software vendors also started collecting user feedback through social media channels, user forums, and review systems. In particular with the emergence of app stores as a software market place and a deployment infrastructure, users can now easily submit their feedback, review new releases, report bugs, rate the app and its features, or request new features [15].

Studies highlighted the importance of user feedback for the success of software products [10]. For instance, apps with more positive reviews get better rankings, better visibility, and higher download and sales numbers [7], [16]. In contrast, frustrated users who feel their voices are ignored might even harm the reputation of software by organizing social media campaigns against it [13]. As a large portion of software is delivered and used through app stores, more and more feedback from the user masses is available. Popular apps such as Dropbox or Facebook can get several thousand reviews in a single day [15]. Conventional environments such as desktop operating systems are following this trend in delivering and deploying software. Large systems such as MS Office, Facebook, Firefox, or Eclipse have their own plugin stores with similar rating and reviewing features.

In this article we discuss how software developers 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 the large amount of 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 thus call this automatically collected information about the software usage "implicit feedback". Finally, 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 when the next release should be offered and what requirements and features should be added or eliminated.

II. User Feedback Analytics

Analytics is the use of analysis, data, and systematic reasoning to make decisions [4]. Recently, app store operators such as Apple, Google, and Microsoft as well as specialized companies such as *AppAnnie* and *SimilarWeb* started offering analytics services to software vendors based on user feedback. A survey of 7000 developers in the beginning of 2014 found that about 40% of them use user analytics tools[1]. At the end of 2014 only 21% of surveyed developers claimed to still use these tools[2]. One of the main reasons was that, most analytics

W. Maalej and T. Johann are with the Department of Informatics, University of Hamburg, Germany (e-mail: {maalej,johann}@informatik.uni-hamburg.de).

M. Nayebi and G. Ruhe are with the Software Engineering Decision Support Laboratory, University of Calgary, Canada (e-mail: {ruhe,nayebi}@ucalgary.ca)

[1] Developer Economics Q1 2014: State of the Developer Nation— available online: http://www.visionmobile.com/product/developer-economics-q1-2014-state-developer-nation/

[2] Developer Economics Q3 2014: State of the Developer Nation— available online: http://www.visionmobile.com/product/developer-economics-q3-2014/

services only focus on sales and marketing decisions, showing download and sales numbers over time, user demographics, and sales behavior. Recent research has pointed out the potentials of feedback analytics for supporting software and requirements engineering decisions as well.

A. Classifying user feedback

User feedback includes a variety of information as Pagano and Maalej showed [15]. A user feedback might include a **bug report**, describing a problem with the app which should be corrected, such as a crash, an erroneous behavior, or a performance issue. A feedback might also include a **feature request**, where users ask for missing functionality (e.g., provided by other apps) or missing content (e.g., in catalogues and games). Users might share ideas on how to improve the app in future releases by adding or changing features. Some feedback describes a **user experience**, documenting a particular experience with the app or a specific feature (e.g. how the app helped in a certain situation). These reviews can be seen as documentation of the app, its requirements, and features. Finally, a pervasive type of information in user feedback is **rating**, which are simple text reflections of the numeric star rating. Ratings are less informative as they only include praise, dispraise, or a distractive critique, or dissuasion.

Analytics tools can help to classify and filter the user feedback according to the information it contains. Researchers studied how text classification, natural language processing, as well as other heuristics such as the star rating, text length, and tense of the text can be used to classify the user reviews of apps into different categories [9], [12]. Recent preliminary results show that no single classifier works best for all review types and data sources [12]. Reviews might include information that falls under more than one category. One review might for example include different sentences, with one being bug report and another being a feature request. Therefore, analytics tools might first need to tokenize the reviews into parts (e.g., sentences or phrases) and extract the types of information separately.

Classifying user feedback would facilitate considering and processing it more efficiently, e.g. by forwarding single entries to appropriate stakeholders and development teams. Feedback only with text reflection of ratings can be summarized or filtered out. Ratings are very persuasive in app reviews (up to 70%) while rather uninformative for development teams [15]. Filtering them would save time. Feedback with bug reports can be forwarded in an aggregated form to developers and quality management representatives. Feedback with feature requests can be assigned to requirements analysts and managers to discuss them and identify requirements for future releases. User experience feedback can serve as ad-hoc documentation for the software or to derive user stories. These feedback entries can also be used as manuals or posted as Frequently Asked Questions to help other users. User experience feedback can also help requirements analysts to better understand the users, in particular, those who have never been considered as a target group. Some user experience feedback also describes workarounds and unusual situations that could inspire app designers, e.g. in developing test cases, capturing exceptions and alternate flows, or designing new features.

Automatically classifying the feedback can give an overall idea about an app's usage and types of user engagement (i.e. how many bug reports, feature requests, etc.). This could also be used for comparing the releases over time (in terms of provided and requested requirements, bug reports) and with similar apps.

B. Classifying stakeholders

Analytics can help to identify and classify various user groups and stakeholders with different characteristics and needs. In particular, deviant users and exceptional use cases can be of interest to requirements analysts and developers.

For instance, searching the iOS App Store for health related terms exposed that a magnifier app and a translator app - both not from the category Health & Fitness - have remarkably many positive reviews by nurses. Reading those reviews we found that nurses frequently use the magnifier app to read the small text package leaflet of medicine. They frequently used the translator app to communicate with patients who do not speak their language.

Analytics tools can mine the feedback to identify descriptions of usage patterns, uncommon vocabulary, or descriptions of roles and tasks. Observing "unintended use" of software, might lead to the repackaging of an app, to reordering the priorities, or to adding new requirements to support new target groups.

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 use the software), privacy (by identifying which privacy tradeoffs are acceptable for which stakeholders), or performance (by identifying which performance requirements apply for which user groups). Indeed a simple search for app reviews including sentences such as "as a blind person, I...", "as official, I ...", or "as a frequent traveller, I ..." will reveal reviews with such details.

C. Summarizing the reviews

Recently, researchers also suggested probabilistic approaches to summarize informative review content.

Feature-based summarization: Guzman and Maalej applied Natural Language Processing and sentiment analysis to extract software features from the user reviews together with a summary of the user opinions about each feature [8]. Figure 1 shows an example of review summarization. This can help engineering teams monitor the reaction of users to the software features by answering the following questions:

- Which features are users talking about most?
- Which features are perceived positively and which are perceived negatively?

Such information assists analysts with quantifying the importance of the software features and prioritizing their work for future releases. Breaking down the user evaluation to the feature level can also be used to monitor the "health condition" of features over time, e.g. before and after major releases. This can also be used to locate buggy parts or components of software more easily.

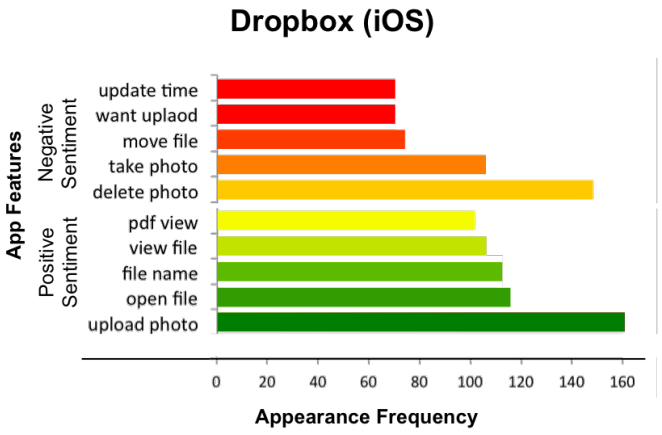

Figure 1: Feature-based feedback summarization. Extracted features with positive and negative average sentiments. Adopted from [8].

The major challenge for feature-based opinion summarization is to identify the features in the reviews together with their associated opinions. Users might have different opinions about different features and mention them all in one review. Moreover, users might use different vocabulary to refer to the same feature, which makes purely syntactic approaches rather inaccurate. Finally, users often use sarcasm e.g. "Great. Now I lost all my data with the update" in their reviews, which makes automated sentiment analysis more challenging.

Topic-based summarization: Galvis Carreño and Winbladh [6] extracted word-based topics from reviews and assigned sentiments to them through a combination of topic modeling (LDA) and sentiment analysis. Similarly, Chen et al. [3] proposed AR-miner, a review analytics framework for summarizing informative app reviews. The tool first filters noisy and irrelevant reviews, such as ratings. Then, it summarizes and ranks the informative reviews using topic modeling and heuristics from the review metadata. The main use case for this work is to summarize and visualize discussion topics in the reviews. For instance, the topic can be the whole release, the app, the price, or the quality of the support.

D. Similar apps: looking beyond the fence

Modern app stores are structured into categories that include similar apps. Popular categories are "Games", "Social media", or "Productivity", but can also be more fine-grained such as "Email apps" or "Navigation systems". Combining the official information about similar apps such as the list of features, the release history, and prices, together with the user reviews and ratings can help with the identification and composition of user requirements for a new app. The feature definition and scoping can be seen as an optimization problem: what is the optimal set of features that the new app should provide, based on the experience from similar apps.

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. Looking into similar apps within each category or across categories [16] is another source for defining potential features. Recently, we conducted a study on 271 similar apps in the same category to create new app products from combining features across existing apps[3]. Applying crowdsourced evaluation of feature attractiveness among app users in conjunction with expert-based effort estimation, we synthesized user satisfaction from offering new combinations of the features. Combining the user opinion and priorities with the effort estimate, we were able to synthetize the set of features that should be combined into a new app. This can also be used to recommend removing or adding a set of features to an existing app. Overall, the results give a direction for the app vendor which features should be changed to improve the overall performance.

In practice, additional analysis dimensions should be taken into consideration, such as innovative features, business requirements, technical requirements, as well as the dependencies between the features and related requirements.

III. IMPLICIT FEEDBACK FROM USAGE DATA

Research has shown that developers and analysts can hardly use user feedback, in particular negative feedback, since it lacks context [2], [15]. While users tend to describe their experience with the software that has led to a positive rating, they tend to rather give short, uninformative, low quality reviews for negative ratings such as "fire the idiot who developed this app", or "worst experience since dating my ex".

To improve the understanding of the circumstances under which users submit their feedback, usage data and the interaction history can be very useful [13]. If captured continuously and presented to development teams, such information can be considered implicit feedback, enhancing the explicit feedback in the form of reviews and comments. While the explicit feedback summarizes the subjective opinions of users (i.e. what they say about the software), the implicit feedback can help with understanding their objective opinion (i.e. how and when they actually use the software and its features). With usage data, developers would be able to better understand the feedback or the decisions of users (e.g. to uninstall, reconfigure, or downgrade the software) and derive respective development decisions.

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. Graphical elements are typically associated with

[3] Details about the study: https://peerj.com/preprints/1354/

specific app features, components, or requirements and analyzing their usage frequency and usage sequence might help with understanding the user needs and improving the software. Other usage data include clusters of feature usage sequences, performance information, and information about the overall usage sessions. This might simply include the time, location, duration, or other co-used apps. However, more advanced situation descriptions based on physical sensors or explicit interactions (e.g. activate the private browsing button, or activate the offline state) can also reveal the usage situations and context.

There is a large body of research on collecting and processing usage data for software engineering, mainly focusing on error reproduction and localization. Tools such as *Apple Crash Reporter*, *BugBuddy*, *Mozilla Talkback*, and *Google Breakpad* collect memory dumps and report to central failure repositories. These approaches do not capture user interactions and usage data and instead focus on performance and quality issues. Other approaches use interaction data and usage data to improve the usability of the systems, provide recommendations to the users, or conduct usability testing [11].

Finally, usage data can support the decisions of developers and requirements analysts, even without the explicit feedback. This can be used to study the sequence of feature usage, the duration, the time of the day and other context, to better understand the user needs. Thermal tracking of user finger movement on touch screens is a common approach for usage data analysis.

IV. FUTURE OF REQUIREMENTS ENGINEERING DECISION MAKING

Requirements engineering was formerly considered a decision-centric process – and nothing has changed or will change about that. Nevertheless, with the new analytics trends initiated from collecting masses of explicit and implicit user data from app stores, social media, or software usage loggers, the actual decisions about requirements and the process with which they are made will change. Overall, we project a transition from stakeholder-centered, intuition- or rationale-based decisions to group-based, mass-driven, user-centered decisions that are based on real-time analysis of a broad range of information sources [14]. These changes relate to *A*. how decisions are made and what information they are based on, *B*. who is making decisions, *C*. what is being decided, and *D*. when to make decisions.

A. How are requirements decisions made

In current RE practices, decision making is typically either based on the intuition and experience of stakeholders, or on rationale schemes (e.g. criteria, options, arguments. …), or both [1][5]. While intuition is subjective, potentially inconsistent, and lacks explanation, rationale has been proven to change over time and is difficult to capture and externalize. In the mass of available explicit and implicit user feedback about software as well as additional project and historical data, synergies between computational intelligence and human expertise are promising.

From systematically observing user communities [13], forums, social media channels, and review platforms, a wide range of information that supports requirements decisions is available. In the future, previously known and used data sources such as business requirements, system and technical requirements, stakeholders preferences, and requirements interdependencies, will be combined with aggregated user data. New ways of data analytics are needed to synthesize this information and provide constructive insights and recommendations. Technologically, we expect to have predictive modeling, scenario analysis, wideband Delphi analysis and adaptive group decision-making to support this process.

B. Who is involved in the decision process

The traditional setting with a well-defined decision maker and a set of nominated stakeholders will be transferred into a much more comprehensive setting which includes a group of unknown users. Social media and other communication channels (e.g. issue trackers and user communities like *UseVoice*) are used to involve users themselves in addition to their data – allowing them to participate in the decision-making process. Crowdsourcing is increasingly being discussed to enlarge the set of stakeholders and to elicit and manage requirements. A result of the 1st International Workshop on Crowd-Based Requirements Engineering (CrowdRE) is an agenda for crowd-based requirements engineering, which embraces event logging, usage mining, text mining, and motivational elements.

Communication with users will move from a unidirectional (from vendors to users) to a bidirectional mode including development teams in addition to conventional sales and marketing teams. To increase user involvement, the CrowdRE community discussed approaches to harness a social network to perform RE activities such as elicitation, prioritization, and negotiation. One other option is to customize e-democracy and e-participation approaches for requirements scenarios [17].

C. Topics and scope of RE decisions

With the increasing agility of development processes, the content of decisions to be made is changing as well. No longer is it a Boolean decision about selecting a requirement. Instead, requirements themselves are implemented incrementally. Decisions are related to enhancing or shrinking the functionality and to increasing the level of specified target quality. Software engineering for game development is a forerunner in this trend, as features are trialed and - depending on the user response - are either enhanced or eliminated. Also, the criteria for making these decisions are changing both in general and specifically over time. Explicit and implicit user feedback should and will be taken into more and more consideration when making decisions about what and when to implement and deliver.

Finally, the boundary between conventional requirements decisions, quality assurance, project management, or sales

decisions will continue to dissolve as the data is getting more and more mixed and the time between prior and post development is becoming of less importance.

D. When are requirements decisions taken

The paradigm shift from reactive to real-time and even proactive decision-making is of key importance in the development of modern software-intensive products. Incrementally building and deploying products relying on deep customer insight and real-time feedback is expected in order to create products faster and with a higher customer acceptance. Changes to objectives, priorities, and dependencies as well as performance and other constraints is happening at real-time. Significant changes in any of these parameters trigger the need to adjust the underlying development processes. As a consequence, an infrastructure and methodology for real-time (or at least just-in-time) decision-making will become more and more important for future software engineering projects.

V. FUTURE AND CHALLENGES

In the above, we have outlined a number of new trends that have been initiated from the emergence of app stores and social media. We foresee requirements elicitation, negotiation, and analysis as an open process with a stronger degree of interaction between developers, analysts, and users. While this has been proven to be true for mobile app stores and marketplaces, we see this as an essential trend for a broader class of software products and services, without saying that this is applicable to all types of software development.

In particular, we foresee a paradigm shift in requirements engineering and software evolution towards data-driven user-centered development, prioritization, planning, and management of requirements. This shift is enabled by the ease of access to user feedback and usage data, the pro-active participation of users through social media, and the maturity of analytics tools and algorithms.

The process of attracting stakeholder priorities in software projects has always been complex and reliable information is hard to get. However, without this information, product development becomes very risky. In data-driven requirements engineering, users and developers can be seen as two groups of stakeholders working on the same objective. User reviews and comments about software can be systematically collected and analyzed to derive work items, define requirements, and prioritize items for the next releases.

Moving towards a data-driven requirements engineering, by the masses and for the masses, bears challenges for researchers and practitioners beyond the technical challenges reported in this paper. **Scaling the analysis** and management of user input to an Internet scale requires a large amount of human resources and sophisticated support tools. **Motivating users** to qualitatively contribute to tasks for which they are not accountable can be very hard. Giving the right incentives to receive evenly distributed and qualitative feedback will be an important task. When making decisions based on user data, the **representativeness** of the participating sample of the whole user population and the quality of data received from the masses must be ensured. Moreover, user contributions are neither necessarily based on rationale nor on logical reasoning and are rather subjective. **Subjectiveness** can hinder innovation. This will require means for communicating with users and explaining requirements decision to them. A stronger participation of the users themselves in addition to using their data is necessary. Similarly, masses of crowd data will most likely lead to **conflicting contributions**. Users themselves are concerned when accessing the crowd data. Analyzing the data may cause **misuse** in the sense of extracting personal information of users. On the other hand it is important to ensure the authenticity of the participation.

Finally, **changing the mindset** in software engineering teams and accepting users as equal stakeholders with potentially good ideas and suggestions is an important cultural challenge.

REFERENCES

[1] A. *Aurum and C. Wohlin, "The fundamental nature of requirements engineering activities as a decision-making process," Information and Software Technology, vol. 45, no. 14, pp. 945–954, Nov. 2003. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S095058490300096X*

[2] *N. Bettenburg, S. Just, A. Schr¨oter, C. Weiss, R. Premraj, and T. Zimmermann, What makes a good bug report? New York, New York, USA: ACM Press, Nov. 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1453101.1453146*

[3] *N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "Ar-miner: Mining informative reviews for developers from mobile app marketplace," in Proceedings of the 36th International Conference on Software Engineering, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 767–778. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568263*

[4] *T. Davenport, J. Harris, and R. Morison, Analytics at Work: Smarter Decisions, Better Results, ser. Harvard Business School Press. Harvard Business Press, 2010. [Online]. Available: https://books.google.de/books?id=2otJuvfyflgC*

[5] *A. Davis, "The art of requirements triage," Computer, vol. 36, no. 3, pp. 42–49, Mar. 2003. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1185216*

[6] L. V. Galvis Carre ˜no and K. Winbladh, "Analysis of user *comments: an approach for software requirements evolution," in ICSE '13 Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, May 2013, pp. 582–591. [Online]. Available: http://dl.acm.org/citation.cfm?id=2486788.2486865*

[7] *T. W. Gruen, T. Osmonbekov, and A. J. Czaplewski, "ewom: The impact of customer-to-customer online know-how exchange on customer value and loyalty," Journal of Businessresearch, vol. 59, no. 4, pp. 449–456, 2006.*

[8] *E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews,"*

[9] *C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in MSR '13 Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, May 2013, pp. 41–44. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487094*

[10] *H. Li, L. Zhang, L. Zhang, and J. Shen, "A user satisfaction analysis approach for software evolution," in Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on, vol. 2. IEEE, 2010, pp. 1093–1097.*

[11] *W. Maalej, T. Fritz, and R. Robbes, "Collecting and processing interaction data for recommendation systems," in Recommendation Systems in Software Engineering, M. P. Robillard, W. Maalej, R. J. Walker, and T. Zimmermann, Eds. Springer Berlin Heidelberg, 2014, pp. 173–197. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-45135-5 7*

[12] *W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in Proceedings of the 23th IEEE International Requirements Engineering Conference. IEEE, 2015, pp. 116–125.*

[13] *W. Maalej and D. Pagano, "On the Socialness of Software," in 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing. IEEE, Dec. 2011, pp. 864–871. [Online]. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6118890*

[14] *M. Nayebi and G. Ruhe, "Analytical product release planning," in The Art and Science of Analyzing Software Data. Morgan Kaufmann, 2015, pp. 550–580.*

[15] *D. Pagano and W. Maalej, "User feedback in the appstore: an empirical study," in 21st IEEE International Conference on Requirements Engineering - RE '13, 2013, pp. 125–134.*

[16] *F. Sarro, A. Al-Subaihin, M. Harman, Y. Jia, W. Martin, and Y. Zhang, "Feature lifecycles as they spread, migrate, remain, and die in app stores," in Proceedings of the 23rd IEEE International Requirements Engineering Conference. IEEE, 2015, pp. 76–85.*

[17] *T. Johann, W. Maalej, "Democratic Mass Participation of Users in Requirements Engineering?" in IEEE 23rd International Requirements Engineering Conference, 2015, pp. 256-261.*

Walid Maalej is a professor of informatics at the University of Hamburg with a chair on Applied Software Technology. Previously, he was leading a research group on context and human aspects in software engineering at the TU Munich, where he received his M.Sc. in 2006 and his doctoral degree in 2010 – both with distinction. His current research interests include data-driven software engineering, context-aware adaptive systems, human aspects of software development and application, e-participation/crowdsourcing, and software engineering impact on society. He is a member of the editorial board of the Journal of Systems and Software and a Junior Fellow of the German Computer Science Society (GI). Dr. Maalej has also served as consultant for several companies including Siemens, Tata Consultancy Services, and Rohde und Schwarz.

Maleknaz Nayebi is a PhD candidate at the Software Engineering Decision Support lab at University of Calgary in Canada. She is pursuing a PhD in mining analytical release patterns. She has six years of professional software engineering experience in different industrial fields. Her main research interests are in mining software repositories, release engineering, open innovation and empirical software engineering. Maleknaz co-chaired OISE 2015 workshop on Open Innovation in Software Engineering being co-located with ICSSP conference. Maleknaz is a student member of the IEEE and ACM.

Timo Johann is a research associate and a doctoral candidate at the Mobile Services and Context Aware Software Engineering Group at University of Hamburg in Germany. He holds a Master of Science from the University of Applied Science in Trier.

Previously he worked in the Green Software Engineering research Project, where he has published several papers on sustainable aspects of software and software engineering. Current research interests include user involvement, crowdsourcing requirements engineering, social aspects of software and social software engineering. Timo is a student member of the IEEE and ACM.

Gunther Ruhe holds an Industrial Research Chair in Software Engineering at University of Calgary. Dr. Ruhe received a doctorate habil. nat. degree (Computer Science) from University of Kaiserslautern. From 1996 until 2001, he was the deputy director of the Fraunhofer Institute for Experimental Software Engineering Fh IESE. Since 2016, he serves as the Editor in Chief of the journal of Information and Software Technology, published by Elsevier. His main research interests are in the areas of Product Release Planning, Software Project Management, Decision Support, Data Analytics, Empirical Software Engineering as well as Search-based Software Engineering. He is a Senior member of IEEE and a member of the ACM. Dr. Ruhe is the Founder and CEO of Expert Decisions Inc., a University of Calgary spin-off company created in 2003.