Crowdsourcing for Software Engineering

The Crowd in Requirements Engineering

The Landscape and Challenges

Eduard C. Groen, Fraunhofer Institute for Experimental Software Engineering

Norbert Seyff, University of Applied Sciences and Arts Northwestern Switzerland

Raian Ali, Bournemouth University

Fabiano Dalpiaz, Utrecht University

Joerg Doerr, Fraunhofer Institute for Experimental Software Engineering

Emitza Guzman, University of Zurich

Mahmood Hosseini, Bournemouth University

Jordi Marco and Marc Oriol, Polytechnic University of Catalonia

Anna Perini, FBK Center for Information and Communication Technology

Melanie Stade, University of Applied Sciences and Arts Northwestern Switzerland

Performing requirements engineering with the crowd of stakeholders (CrowdRE) turns it into a participatory effort supported by automation, leading to better requirements and software quality. Although any stakeholder can contribute, CrowdRE emphasizes one group whose role is often trivialized: users.

Engaging a large number of users in requirements engineering (RE) has always been a challenge with traditional RE methods.[1] This is especially true when RE should involve a large number of software product users (a crowd) who are beyond an organization's reach.[2]

Traditional RE approaches usually involve a limited number of representatives in interviews or focus groups. Advanced RE approaches applied in market-driven RE[3] enable companies to directly interact with key stakeholders using ad hoc feedback-gathering channels.[4] However, these approaches miss the opportunity to continuously involve large, heterogeneous groups of users who express their feedback through a variety of media.[2,5,6] This means developers can't consider the diverse backgrounds of user subgroups when they're developing a product's next version.[7,8] So, valuable resources for RE remain unused, and software products might not meet users' needs.

Crowd-based requirements engineering (CrowdRE) is an umbrella term for automated or semiautomated approaches to gather and analyze information from a crowd to derive validated user requirements. [9] Normally, the crowd is an undefined group of people.[10] But for CrowdRE, the crowd is in most cases a large group of current or potential users of a software product who interact among themselves or with representatives of a software company (for example, the product owner or development team).

CrowdRE strives to mobilize as many crowd members as possible to communicate and discuss their needs regarding the evolution of existing software products. We call the communication from users "user feedback," although such feedback can also come from other stakeholders. In addition, our vision of CrowdRE includes monitoring software application context and usage. It also strongly focuses on a participatory approach in which intrinsically motivated users become crowd members because they benefit from software products that meet their needs.

Eduard C. Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitza Guzman, Mahmood Hosseini, Jordi Marco, Marc Oriol, Anna Perini, Melanie Stade, "The Crowd in Requirements Engineering: The Landscape and Challenges", *IEEE Software*, vol. 34, no. 2, pp. 44-52, Mar./Apr. 2017, doi:10.1109/MS.2017.33

The CrowdRE Approach

Figure 1 presents our proposed CrowdRE approach. We consider the crowd the sender of the feedback and a software company (represented in Figure 1 as a development team) the receiver.

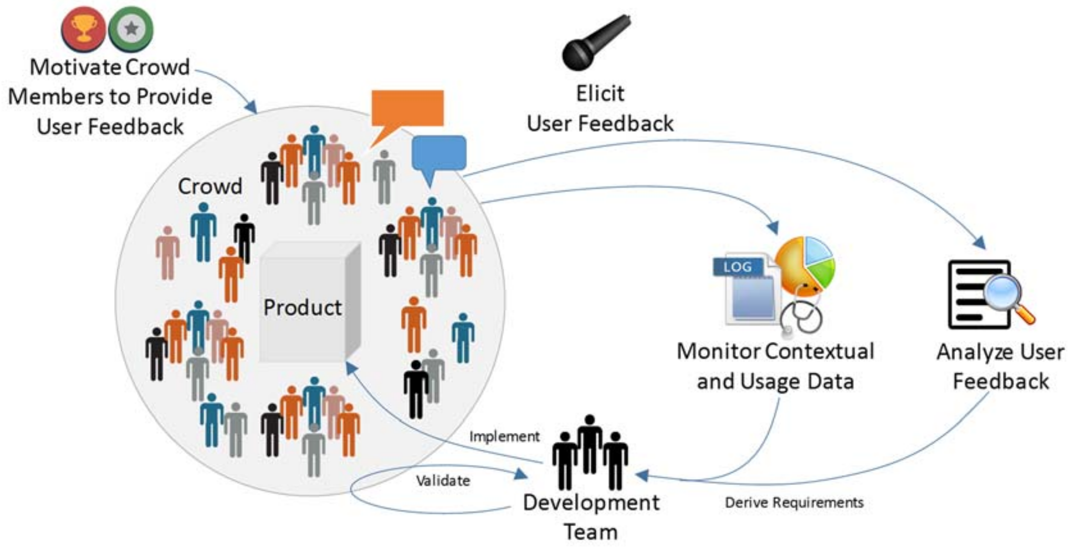

Figure 1. The relationships among the aspects of crowd-based requirements engineering (CrowdRE). CrowdRE strives to mobilize as many crowd members as possible to communicate and discuss their needs regarding the evolution of existing software products.

Pull feedback is when the software company explicitly asks the crowd for feedback. Push feedback is when the crowd initiates feedback.[2] For example, a crowd member could send feedback to an app store—“Hey, what’s wrong with the video quality?”—and rate the app with 2 out of 5 stars.

In this example, the feedback consists of linguistic and nonlinguistic documentation. Linguistic documentation includes natural-language text and audio messages; nonlinguistic documentation includes images, emoticons, and star ratings.11 Multimodal feedback combines multiple documentation formats, as in the previous example.

Ideally, through linguistic analysis, the feedback receiver will classify this feedback as a negative statement about an apparent performance issue (the video quality). The monitoring of context and usage data can gather additional information to help developers better understand the problem (for example, to identify low network bandwidth as the cause). The development team can use this information to resolve the issue. Next, we discuss in detail each key activity in Figure 1.

Motivating Crowd Members

To a considerable extent, CrowdRE depends on a continuous flow of user feedback. An adequate rate of flow can be achieved through motivating crowd members such that the amount and quality of their participation is sufficient. Motivation is intrinsic when crowd members have a genuine interest in contributing to software evolution; it’s extrinsic when it results from external interventions and incentives (for example, monetary rewards such as vouchers or nonmonetary rewards such as social recognition and playfulness).[12] Gamification and persuasive technology are two digital-motivation techniques for boosting task completion and influencing positive behavioral changes.

Regarding attitude and motivation toward giving feedback, we can categorize crowd members as these types:[5]

- *Privacy-tolerant and socially ostentatious* crowd members expect acknowledgment in return for their feedback.
- *Privacy-fanatical but generous* crowd members are motivated by respect for privacy.

- *Passive and stingy* crowd members are motivated by seeing others' feedback and contributing minimally.
- *Loyal and passionate* crowd members care about the software's sustainability and reputation but give less objective feedback.
- *Incentive seekers* care about monetary incentives and pay limited attention to feedback quality.
- *Perfectionists and complainers* are motivated by the self-satisfaction achieved after discovering and flagging a problem.
- *Impact seekers* are motivated by seeing their suggested changes implemented.

CrowdRE should cater to this diversity in backgrounds and expectations and avoid a one-size-fits-all motivation mind-set. For instance, although leaderboards appeal greatly to incentive seekers, privacy fanatics might get discouraged by seeing their names on one. One platform that takes this into account is REfine, which employs game elements to motivate users to express requirements and refine them by commenting, voting, and creating alternative requirements or subrequirements.[13]

Eliciting Feedback

Crowd members report on a variety of aspects, including software problems (for example, bugs), extension ideas (for example, feature requests), or new-product ideas. Although many feedback approaches don't clearly distinguish between these aspects, RE requires this distinction, which certain research prototypes readily provide. Some feedback channels also go beyond eliciting feedback; for instance, social networks such as Facebook can be used to gather, prioritize, and negotiate feedback.[14]

Crowd members must have easy access to feedback channels. In practice, user feedback appears in channels such as app stores,[5,15] product forums, and social media platforms such as Twitter.[8] Software companies also can build functionality into their software that lets crowd members give feedback in situ. Such functionality often focuses on (simple) linguistic feedback. However, multimodal approaches are available (for instance, AppEcho[16]), and the SUPERSEDE project (see the sidebar) is developing more advanced approaches. For example, a permanently visible feedback button lets users start the feedback process themselves. However, we also foresee that development teams will explicitly ask users for feedback.

Providing multiple feedback channels lets developers consider crowd members' individual backgrounds and needs regarding feedback communication. Ideally, this will lead to a large number of users being involved in requirements elicitation, because crowd members can communicate feedback anytime, even without a requirements engineer performing the elicitation. This makes it possible to gather requirements on a much larger scale. Such feedback can complement traditional requirements elicitation approaches such as interviews or workshops in which a limited number of users communicate their needs, supported by a requirements engineer.

Analyzing Feedback

The rise of Web 2.0 platforms such as social media and app stores has caused a surge in user feedback. Manually analyzing large amounts of feedback is time-consuming and cognitively demanding, and potentially suffers from bias (for example, an analyst might unintentionally focus on specific topics). Techniques to automatically analyze large amounts of feedback are necessary to achieve fast, iterative innovation cycles. A good basis for this exists; RE approaches to process large amounts of feedback through computational-linguistics techniques have existed since the early 2000s.[4]

To analyze the feedback gathered from different channels, CrowdRE predominantly uses linguistic analysis techniques such as text mining[2,15] or speech-act-based analysis.[17] This analysis filters out irrelevant data (for example, statements not discussing the product under analysis) and automatically classifies the remaining statements. This classification includes sentiment analysis, which assesses how positive or negative statements are, so that praise and complaints about product features and qualities can be identified.[15]

Furthermore, feedback can be classified into categories such as bug reports and feature requests,[15,18] using predefined feedback taxonomies[11] and topics, and similarities among statements can be identified.[19] Automated classification can also help identify whether feedback discusses certain product features or qualities, leading to the identification of functional and nonfunctional requirements.[9] Projects such as PRO-OPT (see the sidebar) and Opti4Apps (opti4apps.de) are developing such functionality. Researchers are also investigating how to automatically generate models that capture the key elements of natural-language requirements.[20]

Developers can apply further textual analyses to determine feedback reliability. For example, a considerable amount of feedback about a particular issue might indicate a problem's existence and importance. Metadata such as time stamps allow for identifying trends over time—for example, to determine whether a newer version of the product has resolved an issue that received many complaints.15 Furthermore, feedback can help companies compare their products to others—for example, by determining which product receives more positive feedback.

Researchers are also investigating automated analysis of nonlinguistic feedback, such as screenshots describing a problem context.

Monitoring Context and Usage Data

Future software-intensive systems as proposed in ubiquitous computing (for example, the Internet of Things) will deploy multiple sensors in highly distributed environments. This will allow for the comprehensive monitoring of software products and their context and usage data. This, in turn, will provide the capability to gather feedback from multiple sources, letting developers better understand the context and usage data. Such crowd-based monitoring[6] can provide user feedback in CrowdRE.

Crowd-based-monitoring systems are extensible and can aggregate new monitors from different providers at runtime, whereas in traditional monitoring, the monitored entities are usually known and developed at design time. New requirements can be derived from context and usage data gathered at runtime. This includes quantifying performance-related requirements and detecting context-dependent requirements. For instance, monitoring where a software product is used could lead to the requirement that a specific functionality is disabled because of local regulations on the storage of private data regarding underage users. Crowd-based monitoring can also help determine whether requirements are met at runtime (for example, whether a product's performance and reliability meet user expectations in different scenarios).

The monitoring results can then be aggregated with multimodal feedback from users to quantify and better understand similarities and differences, and to prioritize feedback.[21]

CrowdRE in Comparison

CrowdRE is similar to several other approaches. Here, we compare it to customer-specific RE (RE for tailor-made software), market-driven RE (RE for software products), and crowdsourcing.

Customer-Specific RE

Depending on the context, CrowdRE can complement or replace customer-specific RE. In this context, CrowdRE's greatest benefits arise when numerous users are involved. This is because customer-specific RE has difficulties considering the diverse backgrounds of user subgroups when the next version of a product is being developed.[7,8] Conversely, in settings with a limited number of users (for example, software that's tailor-made for a small company with some dozens of employees), traditional customer-specific RE techniques are sufficient because all the stakeholders are easily reached.

Market-Driven RE

Market-driven RE goes beyond the single-customer setting and enables serving a large market of customers.[3] This is typically the case with companies creating products such as office suites, operating systems, or enterprise-resource-planning systems. In market-driven RE, developers obtain information from known sets of stakeholders over longer periods of time through questionnaires, focus groups, and beta tests, which are scheduled at dedicated points in time according to the software release roadmap.

In CrowdRE, feedback comes from a crowd of users or their representatives. This crowd has a weaker bond with the software company, and its feedback data can be obtained using several unobtrusive automated means, without explicit interaction. So, CrowdRE allows continuous collection of feedback from a larger group of stakeholders, which makes it a logical upscale form of market-driven RE, just as market-driven RE is an adaptation that enables customer-specific RE to transcend the organization's boundaries.

Crowdsourcing

Crowdsourcing distributes a workload by outsourcing activities in the form of microtasks to an anonymous crowd that isn't necessarily intrinsically motivated to participate. The motivation of crowd members is usually driven by extrinsic motivators such as pay or the prospect of winning a bounty. In contrast, CrowdRE has a genuine interest in the personal opinion of the users in the crowd. It also aims to provide benefits for participating crowd members in terms of improved software products, increasing user satisfaction.

By involving many crowd members and collecting their opinions and usage data, CrowdRE gives a voice to users. This has been described as a form of social participation,[22] which goes beyond outsourcing simple problem-solving tasks. Moreover, CrowdRE uses automation techniques such as text analysis and monitoring, and applies crowdsourcing strategically in select phases. This way, it can mitigate several threats to crowdsourcing scalability.[10]

Challenges

Although CrowdRE seems promising and practitioners can already use CrowdRE solutions to obtain information from users regarding feature and quality improvements, certain challenges exist, which we discuss separately for each key activity in Figure 1.

Motivating Crowd Members

There's a fine line between motivating crowd members and trivializing their job. Ad hoc introduction of digital motivation might be seen as undermining the task and might adversely affect feedback's usefulness and truthfulness. Therefore, to gather high-quality user feedback, digital-motivation techniques should be adaptive to the context and adaptable by crowd members. Such adaptation and adaptability seem promising to sustain crowd members' motivation and get them to engage in demanding tasks such as argumentation and negotiation of requirements.

Eliciting Feedback

Key elicitation challenges are privacy and personalization. For all feedback channels, from existing platforms to novel built-in feedback channels and monitors, users should be able to influence their level of privacy. For example, a user could allow other users to read a review he or she wrote but not explore the context data gathered. Users should be supported in their decisions regarding when (push or pull), where (the feedback channel and device), and how (which feedback functionalities and to what level of structure) to give feedback. Adaptive approaches addressing this diversity seem promising5 but must be established and evaluated more.

Analyzing Feedback

The input, processing, and output of the feedback data all introduce challenges to feedback analysis. Because feedback comes from online platforms with anonymous users, it's hard to identify user subgroups (for example, by age) and prevent minority groups from being overlooked. Current techniques have difficulties identifying all the relevant data, automatically analyzing multimodal feedback, and estimating the quality of the (text-based) analysis. Because crowd interaction often isn't aimed at achieving consensus, the analysis results require careful interpretation. In addition, exclusively focusing on the frequency of, for example, certain topics can cause important results to be overlooked. Early comparisons with traditional RE and crowdsourcing show promising results, although further analysis must prove that users from minority groups are being heard.

Monitoring Context and Usage Data

The adaptation of monitors to the (changing) characteristics of the crowd and software products constitutes a main challenge. Monitors and sensors must be reconfigurable at runtime and automatically replaced when failing, and the context can be better understood through distributed pluggable sensors.[23] Interpreting contradictory monitoring data (for example, only some users might struggle with a feature) is difficult, but a comprehensive understanding can be obtained by aggregating the data with user feedback from other sources. If the users' privacy is considered, crowd monitoring promises benefits for the industry, including the ability to gather feedback from a large number of representative users.

Introducing CrowdRE in Practice

An overarching challenge is setting CrowdRE up. Companies who plan to apply CrowdRE must first tailor and fine-tune it to their particular usage context. Furthermore, little is known about CrowdRE's successful application in industry. Through user participation and automation, CrowdRE could result in an early return of investment, but unforeseeable issues might exist that prevent its successful application in a particular context. So, more empirical research and case studies are needed to validate CrowdRE and show that it provides the promised benefits.

We expect to see more elaborate solutions in the coming years as researchers and companies adopt CrowdRE. We're currently investing in CrowdRE tools and techniques to validate their potential in real-world settings (see the sidebar).

References

1. V. Dheepa, D.J. Aravindhar, and C. Vijayalakshmi, "A Novel Method for Large Scale Requirement Elicitation," *Int'l J. Eng. and Innovative Technology*, vol. 2, no. 7, 2013, pp. 375–379.
2. W. Maalej, H.-J. Happel, and A. Rashid, "When Users Become Collaborators: Towards Continuous and Context-Aware User Input," *Proc. 24th ACM SIGPLAN Conf. Object-Oriented Programming Systems, Languages, and Applications* (OOPSLA 09), 2009, pp. 981–990.
3. B. Regnell and S. Brinkkemper, "Market-Driven Requirements Engineering for Software Products," *Engineering and Managing Software Requirements*, Springer, 2005, pp. 287–308.
4. J. Natt och Dag et al., "A Feasibility Study of Automated Natural Language Requirements Analysis in Market-Driven Development," *Requirements Eng.*, vol. 7, no. 1, 2002, pp. 20–33.
5. M. Almaliki, C. Ncube, and R. Ali, "Adaptive Software-Based Feedback Acquisition: A Persona-Based Design," *Proc. IEEE 9th Int'l Conf. Research Challenges in Information Science* (RCIS 15), 2015, pp. 100–111.
6. M. Arlitt et al., "Passive Crowd-Based Monitoring of World Wide Web Infrastructure and Its Performance," *Proc. IEEE Int'l Conf. Communications* (ICC 12), 2012, pp. 2689–2694.
7. E.C. Groen, "Crowd Out the Competition: Gaining Market Advantage through Crowd-Based Requirements Engineering," *Proc. 1st Int'l Workshop Crowd-Based Requirements Eng.* (CrowdRE 15), 2015, article 3.
8. E. Guzman, R. Alkadhi, and N. Seyff, "A Needle in a Haystack: What Do Twitter Users Say about Software?," *Proc. 24th IEEE Int'l Requirements Eng. Conf.* (RE 16), 2016, pp. 96–105.
9. E.C. Groen, J. Doerr, and S. Adam, "Towards Crowd-Based Requirements Engineering: A Research Preview," *Requirements Engineering: Foundation for Software Quality,* LNCS 9013, 2015, pp. 247–253.
10. K.-J. Stol and B. Fitzgerald, "Two's Company, Three's a Crowd: A Case Study of Crowdsourcing Software Development," *Proc. 36th Int'l Conf. Software Eng.* (ICSE 14), 2014, pp. 187–198.
11. I. Morales-Ramirez, A. Perini, and R.S.S. Guizzardi, "An Ontology of Online User Feedback in Software Engineering," *Applied Ontology*, vol. 10, nos. 3–4, 2015, pp. 297–330.
12. M. Hosseini et al., "The Four Pillars of Crowdsourcing: A Reference Model," *Proc. IEEE 8th Int'l Conf. Research Challenges in Information Science* (RCIS 14), 2014, pp. 1–12.
13. R. Snijders et al., "REfine: A Gamified Platform for Participatory Requirements Engineering," *Proc. 1st Int'l Workshop Crowd-Based Requirements Eng.* (CrowdRE 15), 2015, pp. 1–6.

14. N. Seyff et al., “Using Popular Social Network Sites to Support Requirements Elicitation, Prioritization and Negotiation,” *J. Internet Services and Applications*, vol. 6, no. 1, 2015; jisajournal.springeropen.com/articles/10.1186/s13174-015-0021-9.
15. W. Maalej and H. Nabil, “Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews,” *Proc. 23rd IEEE Int’l Requirements Eng. Conf.* (RE 15), 2015, pp. 116–125.
16. N. Seyff, G. Ollmann, and M. Bortenschlager, “AppEcho: A User-Driven, In Situ Feedback Approach for Mobile Platforms and Applications,” *Proc. 1st IEEE/ACM Int’l Conf. Mobile Software Eng. and Systems* (MOBILESoft 14), 2014, pp. 99–108.
17. I. Morales-Ramirez, A. Perini, and M. Ceccato, “Towards Supporting the Analysis of Online Discussions in OSS Communities: A Speech-Act Based Approach,” *Information Systems Engineering in Complex Environments*, Springer, 2014, pp. 215–232.
18. S. Panichella et al., “How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution,” *Proc. 31st Int’l Conf. Software Maintenance and Evolution* (ICSME 15), 2015, pp. 281–290.
19. L.V. Galvis Carreño and K. Winbladh, “Analysis of User Comments: An Approach for Software Requirements Evolution,” *Proc. 35th Int’l Conf. Software Eng.* (ICSE 13), 2013, pp. 582–591.
20. M. Robeer et al., “Automated Extraction of Conceptual Models from User Stories via NLP,” *Proc. 24th IEEE Int’l Requirements Eng. Conf.* (RE 16), 2016, pp. 196–205.
21. D. Pagano, PORTNEUF: A Framework for Continuous User Involvement, Verlag Dr. Hut, 2013.
22. R. Ali et al., “Social Adaptation: When Software Gives Users a Voice,” *Proc. 7th Int’l Conf. Evaluation of Novel Approaches to Software Eng.* (ENASE 12), 2012, pp. 75–84.
23. A. Kumar, H. Kim, and G.P. Hancke, “Environmental Monitoring Systems: A Review,” *IEEE Sensors J.*, vol. 13, no. 4, 2013, pp. 1329–1339.

Eduard C. Groen *is a researcher at the Fraunhofer Institute for Experimental Software Engineering. His research interests include deriving functional and nonfunctional requirements from natural-language texts and developing task-oriented development practices. Groen received a master’s in psychology with a specialization in engineering psychology from the University of Twente. Contact him at eduard.groen@iese.fraunhofer.de.*

Norbert Seyff *is a professor of requirements engineering at the University of Applied Sciences and Arts Northwestern Switzerland and a senior research associate at the University of Zurich. His research focuses on requirements engineering and software modeling. Seyff received a PhD in computer science from Johannes Kepler University Linz. Contact him at norbert.seyff@fhnw.ch.*

Raian Ali *is an associate professor of computing at Bournemouth University. His research focuses on the engineering of social informatics. Ali received a PhD in software engineering from the University of Trento. Contact him at rali@bournemouth.ac.uk.*

Fabiano Dalpiaz *is an assistant professor of software systems at Utrecht University. He’s the principal investigator at the university’s Requirements Engineering Lab, and his research focuses on the development and use of semiautomated techniques that help stakeholders improve software requirements. Dalpiaz received a PhD in software engineering from the University of Trento. Contact him at f.dalpiaz@uu.nl.*

Joerg Doerr *is the head of the Information Systems division at the Fraunhofer Institute for Experimental Software Engineering and a lecturer at the University of Kaiserslautern. His research interest is software engineering for information systems, focusing on requirements engineering, especially nonfunctional requirements. Doerr received a PhD in computer science from the University of Kaiserslautern. He’s a member of the German Informatics Society. Contact him at joerg.doerr@iese.fraunhofer.de.*

Emitza Guzman *is a postdoctoral researcher at the University of Zurich. Her research focuses on increasing user involvement during software evolution. Guzman received a PhD in informatics from Technische Universität München. Contact her at guzman@ifi.uzh.ch.*

Mahmood Hosseini *is a lecturer in business computing at Bournemouth University. His research interests include crowdsourcing in requirements engineering and the analysis of transparency requirements.*

Hosseini received a PhD in engineering social informatics from Bournemouth University. Contact him at mhosseini@bournemouth.ac.uk.

***Jordi Marco** is an associate professor in the Computer Science Department at the Polytechnic University of Catalonia (UPC). His research interests include service-oriented computing, quality of service, cloud computing, and monitoring. Marco received a PhD in informatics from UPC. Contact him at jmarco@cs.upc.edu.*

***Marc Oriol** is a postdoctoral researcher of the GESSI research group at the Polytechnic University of Catalonia (UPC). His research interests include service-oriented computing, quality of service, cloud computing, and monitoring. Oriol received a PhD in computing from UPC. Contact him at moriol@essi.upc.edu.*

***Anna Perini** is a researcher in the Software Engineering research unit of the Center for Information and Communication Technology. Her research interests include goal-oriented requirements engineering, regulatory compliance, decision support systems, and collaborative requirements engineering. She's the project coordinator of the SUPERSEDE project. Perini received her Dr. degree in physics from the University of Trento. Contact her at perini@fbk.eu.*

***Melanie Stade** is a doctoral candidate in the Cognitive Psychology and Cognitive Ergonomics research group at Technical University Berlin and works in the SUPERSEDE project at the University of Applied Sciences and Arts Northwestern Switzerland. Her research interests include longitudinal usability and user experience assessment, and user behavior during feedback acquisition. Stade received a Diplom in psychology from Humboldt University Berlin. Contact her at melanie.stade@fhnw.ch.*

Examples of Application Potential

Crowd-based requirements engineering (CrowdRE) has application potential in almost all domains in which software products have many stakeholders from whom usage data and user feedback are obtainable. For example, in the information systems domain, enterprise-resource-planning systems have many users within organizational reach. Furthermore, mass markets exist in which a software product's users are unknown to the software company (for example, mobile apps). In the embedded-systems domain, vehicle manufacturers can exploit monitoring and log data and analyze feedback provided by service personnel and car drivers. In emerging smart domains (for example, smart cities, smart health, and smart energy), the targeted group of stakeholders is very large.

Here, we give examples of CrowdRE in practice and describe two projects that develop and use CrowdRE techniques.

In industry, one platform for gathering and discussing feedback with the user crowd is the Requirements Bazaar (requirements-bazaar.org). Another platform called StakeSource illustrates how stakeholder analysis can directly benefit from crowdsourcing—for example, by predicting ratings of requirements on the basis of similarities in crowd members' voting behavior.[1] By using more classic crowdsourcing instruments, CrowdRE provides the potential to

- obtain user feedback on features scheduled to be incorporated into a new product and
- validate the user requirements derived from this feedback through the crowd's social participation.[2]

The PRO-OPT (Big Data Production Optimization in Smart Ecosystems; pro-opt.org) project aims to enable companies to effectively analyze large business datasets across company boundaries, thereby improving their current and future products, including embedded systems. PRO-OPT uses CrowdRE in a market-oriented setup with automotive manufacturers and suppliers. Reports of car drivers in user portals are analyzed by natural-language analysis and compared to diagnostic data (reflecting context and usage data) obtained at automobile service stations. Through the aggregation of these data, potential root causes of systematic problems (for example, an engine problem occurring sooner in landscapes that tax the engine's performance) can be revealed or even anticipated. The car manufacturer can then fix the problem to prevent the failure, at least in other vehicles of the same model. Also, the requirements derived from this analysis can be used for later models, ultimately benefiting current and future drivers.

The SUPERSEDE (Supporting Evolution and Adaptation of Personalized Software by Exploiting Contextual Data and End-User Feedback; supersede.eu) project is developing multimodal-feedback functionalities that will let a crowd of users provide unobtrusive in situ feedback on software products.

Furthermore, the project is establishing comprehensive techniques to monitor software products and obtain environmental and context data through sensors. The obtained feedback and data will be analyzed to identify relevant information to support decision making during software evolution. Informed decisions based on the feedback and monitoring data will lead to products that better meet user needs and improve the user experience.

References

1. S.-L. Lim and A. Finkelstein, "StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation," *IEEE Trans. Software Eng.*, vol. 38, no. 3, 2012, pp. 707–735.
2. R. Ali et al., "Social Adaptation: When Software Gives Users a Voice," *Proc. 7th Int'l Conf. Evaluation of Novel Approaches to Software Eng.* (ENASE 12), 2012, pp. 75–84.

Section Title: Crowdsourcing for Software Engineering

Article Title: The Crowd in Requirements Engineering: The Landscape and Challenges

Abstract: Crowd-based requirements engineering (CrowdRE) could significantly change RE. Performing RE activities such as elicitation with the crowd of stakeholders turns RE into a participatory effort, leads to more accurate requirements, and ultimately boosts software quality. Although any stakeholder in the crowd can contribute, CrowdRE emphasizes one stakeholder group whose role is often trivialized: users. CrowdRE empowers the management of requirements, such as their prioritization and segmentation, in a dynamic, evolved style through collecting and harnessing a continuous flow of user feedback and monitoring data on the usage context. To analyze the large amount of data obtained from the crowd, automated approaches are key. This article presents current research topics in CrowdRE; discusses the benefits, challenges, and lessons learned from projects and experiments; and assesses how to apply the methods and tools in industrial contexts.

Keywords: crowd-based requirements engineering, CrowdRE, user feedback, requirements engineering, software requirements, crowdsourcing, software development, software engineering

Content Type: orig-research