LEVERAGING THE INTERNAL CROWD FOR CONTINUOUS REQUIREMENTS ENGINEERING - LESSONS LEARNED FROM A DESIGN SCIENCE RESEARCH PROJECT

Pascal Vogel  
*University of Hamburg*, pascal.vogel@uni-hamburg.de

Christian Grotherr  
*University of Hamburg*, christian.grotherr@uni-hamburg.de

Martin Semmann  
*University of Hamburg*, martin.semmann@uni-hamburg.de

Follow this and additional works at: [https://aisel.aisnet.org/ecis2019_rp](https://aisel.aisnet.org/ecis2019_rp)

Recommended Citation

[https://aisel.aisnet.org/ecis2019_rp/4](https://aisel.aisnet.org/ecis2019_rp/4)

This material is brought to you by the ECIS 2019 Proceedings at AIS Electronic Library (AISeL). It has been accepted for inclusion in Research Papers by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
LEVERAGING THE INTERNAL CROWD FOR CONTINUOUS REQUIREMENTS ENGINEERING – LESSONS LEARNED FROM A DESIGN SCIENCE RESEARCH PROJECT

Research paper

Vogel, Pascal, University of Hamburg, Hamburg, Germany, pascal.vogel@uni-hamburg.de
Grotherr, Christian, University of Hamburg, Hamburg, Germany, christian.grotherr@uni-hamburg.de
Semmann, Martin, University of Hamburg, Hamburg, Germany, martin.semmann@uni-hamburg.de

Abstract

Open phenomena including open resources, processes such as crowdsourcing and their effects have initiated fundamental shifts in the way organizations conduct their business. With increasing recognition of the value of openness, represented by principles such as transparency, access, participation and democracy, crowdsourcing has established itself as a suitable mechanism for various use-cases ranging from decision-making to idea generation, microtasking and problem-solving. Increasingly, organizations have begun to crowdsource to their employees instead of external crowds. However, few studies analyze concrete use-cases of internal crowdsourcing and further research is needed. We propose that internal crowdsourcing is particularly useful for software requirements engineering (RE) within organizations. Crowd-based RE alleviates shortcomings of traditional RE approaches such as requiring co-presence or lacking representativeness. Research on crowd-based RE remains scarce, particularly concerning intra-organizational settings and the post-implementation phase of software projects. Defining high-quality requirements demands contextual expertise as well as experience and internal crowds seem to be decidedly well-suited for solving RE-related tasks as they exhibit these exact traits. Following a design science research approach, we develop design principles for internal crowd-based RE and instantiate them in a public-sector organization leading to a holistic evaluation. Subsequently, we formulate recommendations for establishing crowd-based RE in intra-organizational settings.

Keywords: internal crowdsourcing, requirements engineering, crowd-based requirements engineering, design principles, design science research
1 Introduction

In recent years, open phenomena including open resources, open processes such as crowdsourcing and their opening effects have set in motion fundamental shifts in the way organizations conduct their business (Schlagwein et al., 2017). Moreover, with organizations increasingly recognizing the value of openness, represented by principles such as transparency, access, participation and democracy, crowdsourcing has established itself as a suitable mechanism for a variety of use-cases ranging from decision-making to idea generation, microtasking and problem-solving (Schlagwein et al., 2017; Prpic and Shukla, 2016). Crowdsourcing, the act of an organization outsourcing a task formerly performed by its employees to an undefined group of people via an open call (Howe, 2006), enables organizations to flexibly access a large workforce of individuals including its knowledge, creativity and experience (Blohm et al., 2013). Increasingly, organizations have begun to crowdsource to their own employees instead of an external crowd (Benbya and Leidner, 2016; Zuchowski et al., 2016). Internal crowdsourcing gives employees a voice and enables them to actively engage in debates and share their perceptions and ideas to improve current work practices (Zuchowski et al., 2016). This form of employee empowerment fosters competitiveness in organizations due to enhanced decision making capabilities and heightened innovativeness (Zhu et al., 2016).

However, studies analyzing various use-cases of internal crowdsourcing are still scarce and further research in this area is needed (Zhao and Zhu, 2014; Zuchowski et al., 2016). We propose that internal crowdsourcing mechanisms are particularly useful for software requirements engineering (RE) within organizations. As defining high-quality requirements requires contextual expertise and experience, internal crowds seem to be decidedly well-suited for the tasks the RE process entails as their members exhibit these exact traits (Zuchowski et al., 2016). Past research demonstrates the ability of crowdsourcing to tap into the collective intelligence of large, distributed, heterogeneous groups of individuals to engineer requirements for customer-centered information systems (Renzel et al., 2013; Johann and Maalej, 2015; Groen et al., 2017). Involving users in the RE process has a positive effect on requirements quality as well as user satisfaction and general system success (Zowghi et al., 2015). Crowd-based RE approaches attempt to alleviate shortcomings of traditional RE approaches such as a need for co-presence, selection bias as well as a lack of scalability due to their effort and time intensiveness and promise to involve large, heterogenous groups of stakeholders in the RE process (Sharma and Sureka, 2017). However, as with internal crowdsourcing in general, there is little design knowledge available for crowd-based RE concerning both internal and external crowds (Knop et al., 2017). Also, as of yet, most studies on crowd-based RE are focused on the integration of external crowds of users, and studies concerned with internal crowds remain scarce (Snijders et al., 2015). Moreover, studies in the context of crowd-based RE which integrate internal crowds do not extend their scope into the post-project phase of the software product lifecycle. The post-project phase does, however, represent an important crossroads when it comes to the long-term realization of benefits of software projects, an area in which organizations exhibit deficiencies (Markus, 2004; Semmann and Böhmann, 2015). With increasing openness and user empowerment (Rashid et al., 2008), research calls for improving traditional RE approaches to be able to elicit requirements for entire software ecosystems, spanning multiple organizations and covering several application domains (Villela et al., 2018). This lack of design knowledge and applications of internal crowdsourcing within organizations lead us to the following research question:

RQ: What are design principles for continuous internal crowd-based requirements engineering?

In this study, we therefore follow a design science research (DSR) approach to develop nascent design theory (Gregor and Hevner, 2013) in the form of design knowledge. We propose design principles for crowd-based RE approaches that are continuous, i.e. stretch into the post-project phase of software projects and integrate an internal crowd of employees. To demonstrate and evaluate our design principles, we develop a process and corresponding platform for continuous internal crowd-based RE within a public organization. We arrive at a socio-technical artifact embedded in a context-aware process which strikes a balance between transparency and participation enabled by applying internal crowdsourcing to the RE process and the limited organizational resources available for implementing these requirements. The remainder of this paper is structured as follows. Following this introduction, we summarize related work on internal crowdsourcing and crowd-based RE (Section 2). Subsequently, the applied DSR approach and related methodology are presented (Section 3). We then define design principles for
continuous internal crowd-based RE (Section 4) which we apply by developing design artifacts (Section 5). We evaluate these artifacts (Section 6), derive recommendations for conducting continuous internal crowd-based RE and discuss the implications of our findings (Section 7). This paper concludes with a summary of its results, contribution and limitations (Section 8).

2 Related Work

2.1 Internal Crowdsourcing

The term crowdsourcing was first popularized by Howe (2006, p. 1) who defined it as “the act of a company or institution taking a function once performed by employees and crowdsourcing it to an undefined (and generally large) network of people in the form of an open call”. Crowdsourcing can enable organizational value creation in a variety of areas (Durward et al., 2016), for instance generating new product ideas via crowd ideation (Leimeister et al., 2009; Ebner et al., 2008), financing these ideas via crowdfunding (Bretschneider et al., 2014) or evaluating them via crowd testing (Zogaj et al., 2014; Leicht et al., 2016). Increasingly, organizations have begun to crowdsource tasks to their own employees rather than an external crowd. This form of crowdsourcing – internal crowdsourcing – is defined as “an (a) IT-enabled (b) group activity based on an (c) open call for participation (d) in an enterprise” (Zuchowski et al., 2016, p. 168). Recent studies have investigated a variety of application domains for internal crowdsourcing such as internal crowdfunding (Muller et al., 2013; Feldmann and Gimpel, 2016), organizational learning (Schlagwein and Bjørn-Andersen, 2014; Zuchowski, 2016), employee engagement platforms (Semann and Grotherr, 2017) and internal innovation platforms and competitions (Benbya and Leidner, 2016; Wagenknecht et al., 2017b; Hoeber et al., 2016). Internal crowdsourcing seems to be suited for solving a variety of problems which include accessing, integrating and improving internal knowledge (intelligence problems), communicating and developing new ideas (design problems) as well as making decisions and prioritization (decision problems) (Zuchowski et al., 2016).

Internal and external crowdsourcing are distinguishable through several characteristics that go beyond the composition of their respective crowds (employees as opposed to the general public). While in case of internal crowdsourcing the identities, formal relationships and abilities of members of the crowd are usually known, external crowds stay largely anonymous and the abilities of their members can only be assumed (Hetmank, 2014). Consequently, internal crowds can be assigned and are able to perform complex, knowledge-intensive tasks and external crowds are often limited to performing predominantly simple tasks (Hetmank, 2014). In case of internal crowdsourcing, monetary incentives may not be viable due to corporate policy or difficulty in securing a dedicated budget and crowdsourcers need to leverage other incentive mechanisms such as increase in reputation, alignment of personal objectives with the objectives of the crowdsourcing initiative or goodwill and personal fun (Vukovic and Bartolini, 2010). In contrast to external crowdsourcing in which crowdsourcers may struggle to handle large amounts of user-contributed data (Blohm et al., 2013) and insecure intellectual property rights (Hetmank, 2014), internal crowdsourcing faces its own set of challenges. These include a smaller number of participants, a lack of time for participation besides the existing day-to-day workload and a hesitation to participate due to fear of being judged by colleagues or superiors (Malhotra et al., 2017). Anonymous participation or optional anonymity may serve to mitigate these negative effects (Semann and Grotherr, 2017; Wagenknecht et al., 2017c). Furthermore, internal crowdsourcing initiatives face scrutiny from organizational governance and worker representation bodies due to legal and social implications. Malhotra et al. (2017) recommend granting dedicated slack time to employees to use for participation in internal crowdsourcing, possibly formalized via company directive, to mitigate these issues.

2.2 Crowdsourcing in Software Requirements Engineering

The IEEE (2017) defines the term requirement as either a “condition or capability needed by a user to solve a problem or achieve an objective” or a “condition or capability that must be met or possessed by a system […] to satisfy an agreement, standard, specification, or other formally imposed documents”. Requirements engineering can be defined as “the subset of systems engineering concerned with discovering, developing, tracing, analyzing, qualifying, communicating and managing requirements that define
the system at successive levels of abstraction” (Hull et al., 2011, p. 8). RE is a decisive success factor for software projects (Hofmann and Lehner, 2001). Traditional RE approaches employ techniques such as stakeholder interviews, joint workshops or focus groups to elicit stakeholder requirements (Wiegers and Beatty, 2013). These techniques share several shortcomings. Their need for co-presence of all participants makes them cost and time intensive. Consequently, they do not scale well to settings with a high number of stakeholders (Lim and Finkelstein, 2012). As only a selection of stakeholders can participate in these activities, geographically distributed, heterogeneous and diverse stakeholder groups can’t be representatively integrated in the RE process (Law et al., 2012). Representative integration is further hindered by dominant participants and biased opinions or stakeholder selection (Fernandes et al., 2012). These shortcomings are also reflected in the tools supporting traditional RE approaches. These are targeted towards a small number of experts, not approachable for untrained users and offer limited collaboration support throughout the requirements development sub-activities (Lohmann et al., 2009).

Crowd-based RE approaches aim to address the aforementioned challenges of traditional RE approaches and to provide additional benefits of their own. Crowd-based RE comprises “automated or semi-automated approaches to gather and analyze information from a crowd to derive validated user requirements” (Groen et al., 2017, p. 1). The crowd in case of crowd-based RE consists of current or potential users of a particular software product who interact with each other or members of the organization responsible for developing the software product (Groen et al., 2017). Software users are empowered to influence the future development of the software they use (Rashid et al., 2008). Crowd-based RE approaches are able to integrate highly diverse groups of stakeholders with different “fields of interest, knowledge and experience” (Adelbetu et al., 2012, p. 3), fostering creativity and ultimately leading to more relevant and meaningful requirements (Dalpiaz et al., 2017). As they do not require co-presence they tend to be less expensive and scale better for large stakeholder groups (Lim and Finkelstein, 2012). Both participatory and data-driven approaches to RE label themselves “crowd-based”. In case of participatory crowd-based RE, members of the crowd actively suggest and jointly develop requirements through collaboration (Snijders et al., 2015). In case of data-driven crowd-based RE, passively collected software usage data and unidirectional feedback such as app store reviews are analyzed to extract requirements (Maalej et al., 2016). With crowd-based RE, RE can evolve into a process of ample interaction between users, developers and requirements analysts (Johann and Maalej, 2015), opening up the RE process through transparency, participation, accessibility and democratized decision making (Schlagwein et al., 2017).

3 Research Design

Our study followed the design science research methodology (DSRM) (Peffers et al., 2007). Figure 1 depicts the research activities we performed for each DSRM step as well as their outcomes. To ensure the practical relevance of this research project, it was conducted cooperatively with a public-sector organization that served as a source of issues and objectives as well as a proving ground for a real-world evaluation. The case organization is responsible for providing port management services for one of Europe’s largest seaports. Its 1,800 employees perform tasks such as expansion and maintenance of marine and land infrastructure, property management, running a railway network and operating the harbor master’s office. In this case organization, we focus on research on two internally used software products: an internally developed railway management system and the standard software Microsoft SharePoint. For the first two DSRM steps problem formulation and objectives for a solution, we conducted a structured literature review, an internet search and qualitative interviews. Our structured literature review on crowd-based RE contributed to both the clarification of our research problem and serves as the theoretical foundation for our objectives for a solution: design principles for continuous internal crowd-based RE. We conducted our literature review based on guidance by Webster and Watson (2002) and vom Brocke et al. (2009). As recommended by vom Brocke et al. (2015), we performed a full-text search in citation indexing services (Google Scholar, Web of Science) and bibliographic databases (ABI/INFORM Complete, ACM Digital Library, AISel, Business Source Complete, IEEE Explore and Springer Link), filtering for peer-reviewed results where possible. After several search iterations, we identified the following search terms as being most productive (number of hits in parenthesis): “crowdsourcing AND ‘requirements engineering’” (236), “crowd-based AND ‘requirements engineering’” (108) and “social requirements engineering” (124). Other terms such as continuous, collaborative or participatory
requirements engineering yielded no significant results. Backward and forward search revealed two more relevant papers (excluding duplicates). Criteria for inclusion in our literature review were that (1) crowdsourcing is employed in the RE process and (2) the particular RE process encompasses not only requirements elicitation but also their specification, negotiation or prioritization. Including backward and forward search results and after the exclusion of duplicates, we arrived at 19 relevant articles. The last iteration of this search took place in September of 2018. We employed iterative open coding (Flick, 2014) using the qualitative content analysis software MAXQDA to analyze the full-text of these references. The resulting literature concept matrix of 16 concepts (Webster and Watson, 2002) included activities, outcomes and roles in the specific crowd-based RE process implemented in the study, motivational and prioritization mechanisms, properties of the participating crowd, properties of the platform on which crowd-based RE takes place and other concepts.

Figure 1. Research design (based on Peffers et al. (2007))

The internet search was focused on existing, real-world solutions for participative crowd-based RE. Relevant platforms exist under various labels and market themselves as customer feedback management software (e.g. UserVoice) or as idea management platforms (e.g. IdeaScale). Criteria for inclusion in our internet search were that (1) the platform is operated and moderated by an organization for a crowd of software users, (2) the platform elicits software improvement suggestions from users and (3) users can discover and interact with suggestions of other users. Consequently, mere ticketing or unidirectional feedback systems were excluded from the search. We uncovered a total of 12 relevant offerings which we subsequently analyzed for their functionality, resulting in a matrix listing general platform mechanics (e.g. commenting, filtering, voting) and specific RE features (e.g. roadmaps, portfolios or requirement merging and splitting). The findings of this internet search contributed to both the definition of our design principles and the development of our artifacts. We further conducted 8 semi-structured interviews based on recommendations by Myers and Newman (2007) with employees of the case organization to discover issues with their RE process and to elicit their requirements for continuous internal crowd-based RE. The interviews lasted 45 to 90 minutes and included three individuals with roles similar to a software product owner, three members of the internal IT consulting department and the case company’s head of RE. Interview recordings were subsequently transcribed and analyzed using qualitative content analysis (Schreier, 2014) using the software MAXQDA and resulted in a coding scheme of 22 codes pertaining to the concepts in our literature concept matrix, the specific RE process in our case organization as well as its shortcomings as potentials for crowd-based RE.

Based on the findings of our literature review, internet search and qualitative interviews, we formulate design principles for continuous internal crowd-based RE. To ensure consistency and precision of our design principles, we apply the template for formulating design principles proposed by Chandra et al. (2015). Design principles pursue the goal of informing designers on how to effectively design artifacts of a certain type (Niehaves and Orbach, 2016). Consequently, the evaluation of design principles must be concerned with an assessment of their suitability for being instantiated into concrete artifacts and the ability of these artifacts “to proffer the action described by the design principle” (Chandra et al., 2015, p. 4046). In the design and development step, we therefore apply our design principles to develop a prototypical platform and corresponding process for continuous internal crowd-based RE. We subsequently demonstrate and evaluate these prototypical artifacts in a focus group interview (Krueger and Casey, 2014) with 5 participants and three additional qualitative interviews with a total of 4 participants. For our evaluation, based on the framework for evaluation in design science research proposed by Venable et al. (2016), we draw upon the same participants as with our initial set of interviews. Both the focus group interview and the qualitative interviews represent artificial evaluation episodes (Venable et
al., 2012), consisting of a presentation of our prototypical artifacts and a hands-on session where participants assess the artifacts in real-time. Based on evaluation criteria proposed by Prat et al. (2015), we aim at assessing both the instantiated artifacts and the underlying design principles. Our interview guide was structured based on our developed design principles and design elements of our artifacts. The focus group interview as well as the individual qualitative interviews were again transcribed and subsequently coded according to elements of our developed artifacts and our design principles. This leads us to make recommendations for establishing the internal crowd-based RE approach within organizations.

4 Design Principles for Continuous Internal Crowd-based Requirements Engineering

Based on our literature review, internet search and qualitative interviews, we define eight design principles for continuous internal crowd-based RE (Table 1).

| DP1 | Provide functionality for submitting and retrieving software requirements for users to be able to communicate their own needs and understand the needs of other users. |
| DP2 | Provide collaboration tools for users and product owners to be able to jointly specify software requirements. |
| DP3 | Provide functionality for tracking the status of a requirement throughout its lifecycle for users to be able to transparently trace the progress of submitted requirements. |
| DP4 | Provide decision support tools for product owners to be able to identify requirements with maximum impact on user satisfaction given that implementation resources are limited. |
| DP5 | Provide a prioritization system in order for users to rank individual requirements. |
| DP6 | Provide functionality to communicate the implementation effort associated with a requirement in order for user expectations to be managed. |
| DP7 | Provide quality assurance mechanisms to reduce product owner effort. |
| DP8 | Provide a structured process that spans the entire software product lifecycle in order to steer user and product owner activity and to bring about actionable outcomes. |

Table 1. Design principles for continuous internal crowd-based requirements engineering

Traditional RE approaches lack the ability to involve large numbers of users, especially if they are geographically distributed (Groen et al., 2017). Further, the involvement of stakeholders in traditional RE approaches is often selective and may exclude key stakeholders such as current software users (Snijders et al., 2015). This results in a lack of representativeness in the elicited requirements. An approach for internal crowd-based RE should alleviate these drawbacks and offer a standardized channel for all users of a software to participate in the RE process without requiring their co-presence. Each user should therefore be able to communicate his or her needs by submitting software requirements (DP1). This corresponds to the RE activity of requirements elicitation, i.e. the identification of individual stakeholders’ requirements (Wiegers and Beatty, 2013). Once requirements are elicited from stakeholders, they need to be specified, i.e. defined in written form. While in traditional RE approaches, specification would be the responsibility of requirements analysts and a select number of stakeholders (Wiegers and Beatty, 2013), crowd-based RE enables all software users to discuss, refine and clarify requirements collaboratively. Software users express their particular needs and work on their realization with a community of other users (Renzel et al., 2013). Software users are afforded “social spaces” (Law et al., 2012, p. 206) to meet and jointly specify requirements. Efficient collaboration among stakeholders is also shown to lead to higher requirement quality and accuracy (Dalpiaz et al., 2017; Unkelos-Shpigel and Hadar, 2015). Therefore, an approach for continuous internal crowd-based RE should provide users and product owners with the tools necessary to undertake this collaborative specification (DP2). In this context, the term product owner corresponds to the instance inside an organization responsible for determining a software product’s future development. Although traditional RE approaches are concerned with transparency relating to requirement traceability and documentation, this information is ordinarily only available to IT or RE professionals and not to software users (Rashid et al., 2008). Consequently, software users lack the ability to track the status of requirements post submission. Transparent handling of user submissions may, however, positively affect motivation of software users during requirements elicitation (Rashid et al., 2008) and may also support the management of user expectations. An approach...
for continuous internal crowd-based RE should therefore offer software users the ability to transparently trace their submitted requirements (DP3).

As a product owner’s resources for improving a software and developing new features are limited, it is impossible to implement all requirements a software product’s users may have. As a crowd of software users may be able to propose large numbers of valuable ideas, selecting the right ones for implementation represents a significant challenge (Merz, 2018). Presented with different options, the product owner must decide which requirements may yield the highest increase in user satisfaction if implemented (Groen et al., 2017). An approach for continuous internal crowd-based RE should offer decision support tools that enable product owners to make the best possible implementation decision (DP4). A product owner may consider criteria such as the business value if implemented or business penalty if not implemented, implementation cost and associated risks (Dalpiaz et al., 2017) as well as strategy fit and dependencies between requirements (Daneva et al., 2013) when making an implementation decision. Crowd-based RE aims at augmenting this implementation decision with a direct, representative, user-based prioritization (Groen et al., 2017). An approach to continuous internal crowd-based RE should therefore implement user-based prioritization (DP5). This corresponds to the RE activity of negotiation, i.e. the definition of implementation priorities within a set of requirements (Wiegers and Beatty, 2013). To support users in making their prioritization, it may be constructive to help them understand the implementation effort associated with the requirements they propose. An untrained user may not be able to correctly assess the implementation effort of his or her proposed requirement. However, grasping the implementation effort of one’s own requirements and those of other users may contribute to the management of user expectations. Without awareness about an organization’s limited implementation resources, a user-based prioritization could raise expectations that the company might not be able or willing to meet (Snijders et al., 2014). However, if a user recognizes that many high-effort requirements have been proposed, he or she may better understand that it is not possible for the product owner to implement all of them. An approach for continuous internal crowd-based RE therefore needs to communicate a requirement’s implementation effort to software users (DP6). Furthermore, software users with no training in RE specification cannot be expected to submit high-quality and fully specified requirements on an open platform without any assistance. While requirement quality may improve through collaborative specification and with support of product owners, at least initially there is a danger of low-quality requirements being submitted. As users of a particular software are likely to have similar needs, large numbers of duplicate requirements can be expected if no preventive measures are taken. Crowd-based RE approaches are efficient in discovering and preventing redundant requirements, contributing to quality assurance (Ninaus et al., 2014). We therefore define the implementation of quality assurance mechanisms with the goal of reducing product owner effort as a design principle (DP7).

Groen et al. (2017) identify a lack of continuity in current RE approaches and call for the continuous collection of feedback from a group of heterogeneous stakeholders. As the work practices and work context of members of an organization are subject to constant change, the software they use to perform their work has to keep up by continuously evolving and adapting (Law et al., 2012). If an organization fails to continuously improve a software product after its implementation, the realization of its envisioned benefits is at risk (Semmann and Böhmann, 2015). We therefore define a process that extends into the post-project phase of a software product’s lifecycle as a design principle for continuous internal crowd-based RE (DP8). This process should provide some form of structure to the approach in form of a breakdown of steps that have to be performed, time limits for each step and pursued intermediary and end results. Without this structure, the process may lack momentum which could ultimately result in a lack of participation and actionable results.

5CrowdCore Process and Platform

We apply the design principles to develop a process and platform for Crowd-based Continuous Internal Requirements Engineering (CrowdCore). The overall objective of these artifacts is to ensure long-term realization of a software product’s intended benefits post-implementation. It does so via elicitation and collaborative specification of requirements directly from software users and by support software product owners in selecting those requirements which promise maximum impact on user satisfaction. The CrowdCore process (DP8) consists of four distinct, deadline-bound phases that are grounded in open
innovation and crowdsourcing practice (Wagenknecht et al., 2017a): ideation, consolidation, voting and decision (Figure 2). Two roles are active in this process: users of a software product and a product owner, the role responsible for determining the future development of that software product. We model this role according to the product owner in agile software development whose key responsibilities include maximizing the value delivered by the development team to the organization, management and prioritization of the product backlog and communicating with all stakeholder groups (Cohn, 2005). Throughout all four phases, interaction takes place between software users, the product owner, and requirements submitted by software users. Letting users interact freely on an open platform necessitates some form of oversight and guidance (Lohmann et al., 2009) which in case of CrowdCore falls under the responsibilities of the product owner. In his capacity as a facilitator, the product owner motivates members of the crowd to interact and participate by providing incentives such as praise and encouragement (Leimeister et al., 2006), by managing expectations and by advising software users in the specification of requirements. Feedbacking crowdsourcing participants on their submissions is an important quality control mechanism (Tavanapour and Bittner, 2018). The ideation phase represents the entry point to the CrowdCore process cycle, in which software users can submit their requirements (DP1). Further, it affords software users time to discover and interact with the submissions of other users with the goal of collaborative requirements specification (DP2).

The consolidation phase allows a product owner to determine which requirements he or she wants to let proceed into the voting phase. Requirements which at this point in the CrowdCore cycle do not comply with quality criteria such as feasibility, correctness, uniqueness, verifiability, clarity or consistency (Hull et al., 2011), should be closed with appropriate and transparent feedback in form of a status update (DP3). A product owner may also want to merge overlapping requirements into one or split large requirements into multiple ones. During the voting phase, users prioritize the remaining requirements (DP5). For our employed prioritization mechanism, we draw on enterprise crowdfunding mechanisms (Feldmann et al., 2014). At the beginning of the voting phase, each user receives a budget of votes on a per-software basis that he can distribute to requirements. This vote budget can be adjusted by a software’s product owner. Users can spend any number of votes on a single requirement but voting negatively, i.e. reducing the net amount of votes a requirement possesses, is not possible. However, votes that have been spent can be taken back as long as the voting phase is active. Voting for one’s own requirements is not possible and voters remain anonymous. We choose this voting system for a number of reasons: first, it allows users to grade their degree of support by spending more than one vote on a single requirement. Being able to customize the vote budget in each CrowdCore process cycle allows product owners to assign a vote budget to users of a software that is smaller than the number of proposed requirements. Such a scarce vote budget suggests to software users that implementation resources are not unlimited and forces them to make a prioritization decision.

In the decision phase, a final set of prioritized requirements is selected by the product owner. This set of requirements should present those requirements which promise a maximum impact on user satisfaction (DP4). While the consolidation phase was focused on filtering out requirements which do not meet quality criteria, the decision phase is focused on selecting requirements based on the value they provide for software users. The prioritization made by software users during the voting phase plays an important role in this decisions, but a product owner will also want to consider other prioritization criteria such as the strategic fit and business value of a requirement as well as dependencies and risks associated with a requirement (Daneva et al., 2013) in his or her decision. Each requirement receives individual feedback concerning the reasoning behind the product owner decision (DP3). The last phase of the CrowdCore process results in a set of prioritized requirements. These requirements can then be introduced into the software development process of the software product in question. As a result of the flexibly adaptable CrowdCore process cycle duration, the CrowdCore process can be adapted to match both agile software development approaches such as SCRUM which are able to manage a high cadence of requirements input and more traditional approaches with infrequent large-scale releases. Subsequently, the CrowdCore process starts anew by transitioning into the ideation phase. Requirements which did not get selected for implementation can still be accessed on the CrowdCore platform and may be reintroduced during the next CrowdCore process cycle. Phase-changes are initiated at fixed deadlines defined by the product owner. The ideation phase is afforded the longest amount of time to ensure all software users get the opportunity to voice their opinion and to allow sufficient time for interaction with and
collaborative specification of requirements. The consolidation and decision phases are short to keep product owners from dragging out decisions and to keep the entire process cycle from stalling. Based on an analysis of the software development and release process in our case organization, we assume four twelve-week CrowdCore process cycles per year with the ideation phase taking 8 weeks, voting 2 weeks and consolidation and decision one week each. This duration should only be considered a default which can and should be adapted to the specific context in which the CrowdCore process is applied.

The CrowdCore platform is implemented as a single-page web application and was developed using the web development framework Django which allows for rapid software prototyping. The platform initially presents users with an overview of all requirements that have been submitted during the ideation phase and their associated meta-information such as title, author, date of creation and tags. In our exemplary screenshot (Figure 2), the CrowdCore platform is used to develop requirements for the software product Microsoft SharePoint 2016. Requirements can be submitted to the platform by entering an initial specification and assigning tags (DP1). A prefilled submission template provides guidance to users submitting a requirement. Users on the CrowdCore platform are labeled with their real name and possess a profile page which displays their individual activities on the platform, statistics on how many votes they have received for their proposed ideas and their competencies. For some users, being able to demonstrate their competency in a certain field can be an important motivator and predictor of quality of submitted ideas (Bretschneider et al., 2012).

Figure 2. CrowdCore platform in use for the software product Microsoft SharePoint (voting phase, sample data) and CrowdCore process

The submission entails a self-assessment test (Figure 3) consisting of five questions that compel users to reflect on their requirement before submission with the goal of ensuring requirement quality and reducing product owner effort (DP7). Similar to the criteria used by the product owner for assessment in the consolidation phase, the assessment uses quality criteria such as feasibility or verifiability. The questions are formulated as to be comprehensible to software users without RE knowledge and are answered on a simple three-point Likert-type rating scale (Lehmann and Hulbert, 1972) of ‘Agree’, ‘Unsure’ and ‘Disagree’ to allow for a neutral answer. Each question answered as ‘Agree’ increases the test score by one while each question answered as ‘Don’t agree’ decreases the test score by one. Questions answered ‘Unsure’ have no effect on the test score. If the test score is higher than or equal to three, a message of encouragement is shown to the user, inviting him to submit his requirement. A score below zero advises the user to refrain from submitting his requirement and seek support while a score from zero to two produces a warning message, asking the user to carefully review his requirement before submitting it. With just five questions and instant feedback, the optional test is kept simple and as lightweight as possible. Requirements can be discovered on the CrowdCore platform in a number of ways (DP1). Aside from browsing the paged list of submitted requirements, users can perform a full-text search on requirement specifications and metadata. Further, filtering and sorting via tags, author and implementation status are supported. Users can also subscribe to requirements and receive notifications for any new votes, discussion entries or status changes and share requirements via email. Each requirement’s detail view consists of the requirements specification, metadata and a discussion system which is the primary tool for collaboration on the platform (DP2). The discussion system allows software users to weigh pros and cons of an individual requirement, express support or disapproval and to contribute additional specification information. Individual discussions are structured into threads which keeps complexity at an acceptable level even in case of multiple parallel conversations concerning the same
requirement. Product owners can provide a public effort estimate for a requirement to give software users a rough assessment of the effort associated with its implementation. This effort estimation uses a simple three-point scale of low, medium and high and a complementary textual description to communicate implementation effort (DP6).

![Figure 3. CrowdCore platform user self-assessment and impact effort matrix](image)

The CrowdCore platform uses a status system consisting of the five statuses ‘Open’, ‘Backlog’, ‘In progress’, ‘Implemented’ and ‘Closed’ to transparently communicate the implementation status of a requirement to software users (DP3). Statuses are set by product owners and must be accompanied by a status update message describing the reasoning behind the status change. They are visible on the overview screen and each requirement’s individual detail view. Users can also access a simple software roadmap that displays requirements with the status ‘Open’, ‘In progress’ and ‘Implemented’ for a software product. We do not choose a more precise time scale for the roadmap (e.g. Release 2 or Q2 2018) as product owners may be reluctant to make such binding commitments. Fuzzy categories such as ‘In progress’ do not require a discouraging degree of commitment from product owners. Further, the software roadmap displays the public effort estimation for each requirement. Besides being a planning and decision-making tool, a product roadmap can also serve as an outward-facing communication channel, intended to let software users know which features to expect in the future and when, contributing to expectation management (DP6). At the same time, the roadmap ensures commitment from product owners, as each implementation decision is transparently communicated to all software users. The CrowdCore platform offers decision support in the form of dynamically generated impact and effort matrices that can be accessed by the product owner (DP4). An impact and effort matrix maps potential projects or products by the effort required to implement them and their expected impact (Gray et al., 2010). In case of CrowdCore, these matrices display the votes a requirement has received during the voting phase as well as the effort necessary for its implementation (Figure 3). This effort estimation used for generating impact and effort matrices is not visible to software users and is based on an assessment by the product owner on the planning poker scale (Cohn, 2005). Based on this assessment, the product owner can identify those requirements which promise the highest impact by selecting those with a high number of votes in relation to their estimated effort (DP4). Evaluating requirements using the impact and effort matrices can assist in prioritization and serve as an input for roadmap planning (DP3).

6 Evaluation of CrowdCore Process and Platform

In this section, we present a selection of the most meaningful insights regarding the effectiveness of our developed artifacts and underlying design principles gained during our evaluation. As a general conclusion from the evaluation with experts from our case organization, participants seem to consider our approach a suitable tool for representatively involving and empowering software users in the RE process. Our case company’s head of RE summarized: “Generally, it is well-suited to integrate users. And to not only give them the feeling but also to visualize that their opinion matters”. Evaluation participants stated that status updates and corresponding notifications as well as the simple product roadmap represent a suitable tool for informing software users of the status of their submissions (DP3). A product owner described: “A lack of transparency is a central topic in IT. We often hear: ‘Why don’t we get any
feedback, why aren’t we notified of anything?” With this we would have a good way of providing transparency”. Product owners appreciated the decision support (DP4) provided by impact and effort matrices, one stating that “for the product owner it is not only a decision support, but it opens up an information source he would otherwise not have”. Further, product owners approved of the vote budget distributed to software users (DP5) on a per-software basis which allows them to roughly scale the vote budget to their available implementation resources with the abstraction layer of votes shielding them from having to precisely disclose their monetary budget. Evaluation participants did, however, question if internal crowd-based RE can be applied for all types of software products: “The software does by itself have to be somehow customizable in a way that when a requirement is defined, or a proposal made, that these things can be realistically implemented”. If an organization collects a large number of requirements for a software from its users but is unable to implement these requirements in a timely manner, there might be severe backlash from software users, ultimately resulting in a decrease in crowd participation. Financing the implementation of requirements developed via crowd-based RE was subject to discussion during evaluation. After a software is successfully implemented in the case organization, there is usually no dedicated budget available for the implementation of new features. Even if a customer, such as a department inside the organization, would be willing to pay for new features, capacity for developing such features inside the IT department is limited.

A product owner for a software product accessible by all members of the case organization expressed that he strongly believed the capabilities of the crowd of software users would be an important predictor for the success of internal crowd-based RE: “You need a certain suitability and competence of the crowd”. He further stated that while it is to be expected that untrained software users will not produce high quality requirements on their own, this would not constitute a “deal-breaker”. An organization should rather “empower people and give them an understanding of requirements” to achieve crowd-based RE’s intended result. The case organization’s head of RE described internal crowd-based RE as a “double-edged sword”. He expects that the establishment of internal crowd-based RE would necessitate an initial investment in terms of technology, personnel, training and internal communication as well as a warm-up phase until first actionable outcomes are produced. Yet, a representative source of information on user needs, the potential to discover synergies between requirements and the elimination of duplicates are desirable benefits from a RE perspective. In his opinion, the success of internal crowd-based RE is strongly dependent on software users trusting product owners to commit to implementing the user-prioritized requirements. Consequently, he deems it essential to define a well-structured process with defined outcomes and deadlines (DP8) that is complied with by software users and product owners alike. The evaluation demonstrates that our CrowdCore artifacts seem to be able to effectively and representatively integrate software users in the requirements engineering process via elicitation and collaborative specification of software requirements. Product owners are empowered to select the most promising requirements for implementation. While these results also point towards potential improvements of a software product’s benefit realization post-implementation, a naturalistic long-term evaluation is be necessary to confirm such an effect.

7 Discussion

7.1 Recommendations for conducting continuous internal crowd-based RE

Based on the feedback collected during our focus group interview and individual interviews, we formulate recommendations for conducting crowd-based RE in an intra-organizational setting. Continuous internal crowd-based RE is not a suitable approach for collecting requirements for every type of software product. As a precondition for implementing such an approach, it is therefore necessary to assess a software’s suitability. As one of the goals of crowd-based RE is to involve large numbers of geographically distributed users in the RE process (Lim and Finkelstein, 2012), software products with small and geographically concentrated user bases are less suited. Further, in case of standard software developed by a third party, granting users the opportunity to communicate their needs will inevitably result in disappointment as the organization has no way of following up on requirements. Software that is suitable consequently includes software that is developed by the organization itself (or by a third party on behalf of the organization) as well as extendable or customizable software. Similarly, organizational demand
and readiness for a crowd-based RE approach are needed. Our case organization showed openness towards embracing our proposed crowd-based RE approach because it fits in with an ongoing overall paradigm shift affecting the entire organization. As one evaluation participant explained: “We are in the middle of this cultural change. The [case organization] came from ‘no one is allowed to say anything’ to ‘everyone can say something’. And right now we are working on integrating this participation or this dialogue process.” Our evaluation further demonstrated that software users may struggle to differentiate between software requirements and question and answer (Q&A) type topics related to software errors or usage instructions. As the publication and discussion of Q&A type topics would divert user attention from requirement submission and specification and cause significant product owner effort, organizations need to strictly enforce content delineation between requirements and other types of submissions through preventive communication and moderation. Organizations need to provide adequate resources to execute crowd-based RE and implement its outcomes. Personnel resources for facilitation by the product owner are necessary to execute crowd-based RE. Depending on the specific organizational context, software users need to be granted dedicated time during which they can participate in crowd-RE, possibly formalized via a company directive (Malhotra et al., 2017). Implementing requirements elicited through crowd-based RE necessitates securing a dedicated budget. In case of the hitherto dominating use-case of crowd-based RE, software companies collecting feedback from external customers, budgets for developing new features could always be assumed to be readily available. However, especially in case of internally used software, development budgets are often constrained or mainly allotted for maintenance and security fixes. As previous research demonstrates, it is particularly difficult to secure budgets for software improvement post-implementation (Semmann and Böhmann, 2015; Markus, 2004; Wagner and Newell, 2007). Transferring crowd-based RE into intra-organizational settings therefore requires organizations to secure budgets for implementing outcomes of crowd-based RE.

A frictionless integration of crowd-based RE into an organizational context requires organizations to establish organizational interfaces. If implemented, crowd-based RE is likely to take over some of the responsibilities formerly held by a different unit or role inside an organization. To avoid confusion and conflicting responsibilities, interfaces in form of intra-organizational agreements need to be defined, an observation in line with previous research (Grotherr et al., 2018a). In our case company, the service desk represented – for lack of other options – the primary channel for users to submit software requirements which were then forwarded to the IT or requirements management department. Establishing crowd-based RE would shift that responsibility away from the service desk. One evaluation participant explained: “If you really want to introduce this into [organizational] culture, I would define clear rules concerning expectations towards [the service desk]. And then I would explicitly take receiving improvement suggestions out of the scope of work [of the service desk] and point towards the platform instead”. Consequently, the service desk needs to be instructed on how to handle incoming requirements in the future, likely by agreeing that the service desk should refer users to the crowd-based RE platform. Furthermore, if tightly integrated with internal IT via mechanisms such as single-sign-on, internal crowdsourcing initiatives benefit from lower entry barriers (Rohrbeck et al., 2015).

7.2 Implications

Based on our application of internal crowdsourcing for RE in an intra-organizational setting, we were able to develop important design knowledge. Internal crowdsourcing has the ability to address a number of problem types, including intelligence, design and decision problems (Zuchowski et al., 2016). While some applications of internal crowdsourcing leverage the ability to solve one of these problem types, applying internal crowdsourcing to RE necessitated solving all three problem types. CrowdCore’s ideation phase is aimed at solving both a design and an intelligence problem. Eliciting new requirements can be considered a design problem as users present novel ideas in an enterprise-wide brainstorming while the collaborative specification of these proposed requirements represents an intelligence problem, accessing and integrating a pool of internal knowledge. The voting phase represents a decision problem as in this phase users select the requirements of their preference for implementation through voting. Therefore, our research demonstrates that while crowdsourcing mechanics can be scaled up to serve as the prime design element of an IT artifact, e.g. crowdfunding in the case of Kickstarter, it is also possible to flexibly adapt and implement these individual mechanics as building blocks for artifacts with a diverging overall objective. In our case, CrowdCore’s voting system represents a building block
influenced by internal crowdfunding mechanisms (Feldmann et al., 2014; Muller et al., 2013). Each software user is assigned a vote budget per software product and can subsequently spend these votes on requirements to show their support in the voting phase of the CrowdCore process. However, the CrowdCore process puts a stronger emphasis on collaborative development of an idea before it is put up for funding as compared to the crowdfunding process, as proposed for example by Beaulieu et al. (2015). In fact, the ideation phase of the CrowdCore process in which software users collaboratively specify requirements, is granted the highest amount of time during a process cycle out of all phases. As opposed to CrowdCore, crowdfunding also features a publicly visible funding goal that once it is reached implies that all necessary resources for implementing an idea have been collected. Furthermore, CrowdCore does not offer tangible rewards to software users who vote for a requirement. However, both approaches value transparency, in case of crowdfunding via regular updates during and after a crowdfunding campaign and in case of CrowdCore via status updates and a public roadmap. In public crowdfunding campaigns, this need for transparency likely arises from the fact that individuals spend their own money to fund ideas. In case of internal crowdfunding where each employee receives “virtual” spending money, this effect may be less pronounced (Muller et al., 2013).

As information systems are multilevel in nature (Bélanger et al., 2014; Grotherr et al., 2018b), our proposed recommendations highlight the importance of an ensemble view of technology, one of the information systems artifact being embedded in a constantly evolving social and environmental context (Orlikowski and Iacono, 2001). While an information systems artifact may possess each capability its designer envisioned, it is of equal importance to consider training, support services, organizational arrangements and policies as well as incentives to enable its continued effective management and use (Orlikowski and Iacono, 2001). For instance, while the commercial platforms discovered during our internet search can be a source of inspiration for the IT-enabled platform, these offerings do not bring along a process with predefined roles, activities or outcomes. Without such a process, these platforms represent tools which fulfill an envisioned purpose but do not consider the organizational needs they will be embedded in. Not assuming this holistic perspective and focusing on the design of the artifact itself and not its surroundings may result in artifacts that are hard to integrate in a social context.

8 Conclusion

In this study, we followed a DSR approach to develop design principles for continuous internal crowd-based RE. We ground these design principles in existing literature on internal crowdsourcing as well as crowd-based RE and draw on qualitative data collected from a public-sector case organization. We present eight design principles and apply them by developing a process and platform for continuous internal crowd-based RE. Issues discovered in our evaluation of the developed artifacts enable us to make recommendations for conducting internal crowd-based RE. This paper contributes to the field of internal crowdsourcing by developing design principles for one of its application areas: improving an organization’s approach to RE. We further contribute to knowledge on crowd-based RE by proposing design principles that focus on intra-organizational settings and extend into the post-project phase of the software product lifecycle. We offer valuable insights to practitioners seeking to leverage crowdsourcing to continuously improve the software used inside their organization, especially in the post-project phase. Our recommendations address important preconditions and implementation considerations for continuous internal crowd-based RE. In their entirety, our proposed design principles and developed artifacts demonstrate how internal crowdsourcing enables a new form of open requirements engineering; one establishing transparency concerning effort, support among peers and status of a requirement, equal access to the RE process for all software users, participatory specification of requirements and their democratic prioritization. Our study is faced with several limitations. Although our research is rigorously grounded in the extant literature on internal crowdsourcing and crowd-based RE, our empirical data is limited to qualitative interviews within a single case organization. And while we did evaluate our design principles and the effectiveness of our artifacts via a focus group and qualitative interviews, a naturalistic long-term evaluation is necessary in order to substantiate the effectiveness of our work.

ACKNOWLEDGEMENTS This research was funded in part by the German Ministry for Education and Research (funding reference 02K14A170) as part of the research project EXTEND. Further information can be obtained via http://www.projekt-extend.de.
References


