Survey on Crowd-based Requirements Engineering

Serhat Uzunçavdar - 2017719120

1. Introduction

In the success of a system development, Requirements Engineering (RE) plays a very critical and effective role. This is simply because that this process solely focuses on the question: "What we are going to build?". In order to understand what the system-to-be should look like, the initial step of any system development should start with requirements elicitation. Once this process is done with success, it even may be the case that the whole idea is not feasible to build. RE is not an easy process and mainly depends on the people because after all, people will be using or benefiting from the developed system.

Workshops and interviews with the stakeholders are the most common activities in the RE process. Increase in the count of stakeholders will help to cover more breadth of requirements. Increase in the details of these workshops and interviews will help to cover more depth of the requirements. However, these are both time and budget consuming activities. On the other hand, too much breadth and depth can become hard to verify or realize. It is really important to keep RE activities in an optimum balance between its coverage (such as breadth and depth of requirements) and project constraints (such as time and cost). This is an important process because requirements must be complete but should not contain early decisions.

As seen, there is an ongoing tension between keeping depth in an effective level and remaining at a sufficient level of abstraction. In order to cover the breadth of requirements while remaining at a reasonable cost levels, one solution is to select smaller groups of stakeholders and this may cause non-verified or non-detailed requirements. To tackle these issues, this process can take advantage of automated approaches such as crowd-based RE. The most significant difference is that these techniques allow elicitation process to be performed remotely and reduce the necessity of the co-presence of stakeholders.

Crowd-based requirements engineering (CrowdRE) is a generic term for automated or semiautomated approaches to gather and analyze information from a crowd to derive validated user requirements.[1] In Figure-1, the general workflow of a Crowd RE process can be seen. Based on this, the main concerns of a CrowdRE process are:

- Getting the feedback in a controlled process
- Analyzing the feedback in order to produce meaningful, refine outputs

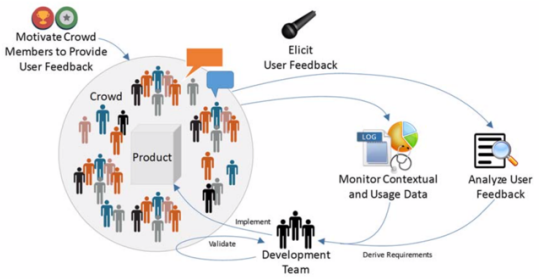

Figure 1 - The concept of CrowdRE [1]

Reasonably, a crowd is the initial component of any CrowdRE process. In terms of CrowdRE, a crowd is a group of people with a common interest in a product. The main idea behind CrowdRE is to collect usable information from this crowd and transform that feedback into meaningful and refined requirements.

Obviously generating more and more requirements is not the whole point of CrowdRE process. Once the new requirements are being generated based the crowd feedback, the system development teams analyze and prioritize these requirements and build the next-gen of the system. After that, this process should continue on its upcoming cycles and the system should be developed to the point where it can satisfy most of the stakeholders' needs and requests. We say "most of" because it is nearly impossible to satisfy everything and everyone perfectly within one well-developed system.

On this paper we will try to discuss about the basic workflow of a CrowdRE process. During this paper, first we will talk about the feedback analysis and the main challenges of this step. On Section 3 we will discuss about the monitoring process of these feedback. On Section 4 we will discuss about the legal aspects of the Crowd RE processes. Finally, we'll review and summarize the CrowdRE as the final discussion.

2. Eliciting & Analyzing the Feedback

Crowd is just one component of a CrowdRE process. We need much more than a crowd. We need some "ways" to collect and analyze the user feedbacks. In this section, we'll briefly talk about the approaches and challenges of a CrowdRE process.

Once you have the crowd, next up, there must be a way for this crowd to give feedback for that particular product such as a feedback tool or a survey or the usage data that every individual in the crowd produces. Many different platforms can be used as a feedback channel. App stores, product forums and social media platforms such as Twitter can be considered as a feedback channel.[1] These platforms are not the only channels to gather feedbacks from the users. Gamification is another approach in CrowdRE for gathering user feedbacks. In addition, some products have a built-in feedback module and/or they ask users' consents to monitor and collect their usage data. For example, in MS Office you may have seen the following dialog given in Figure 2:

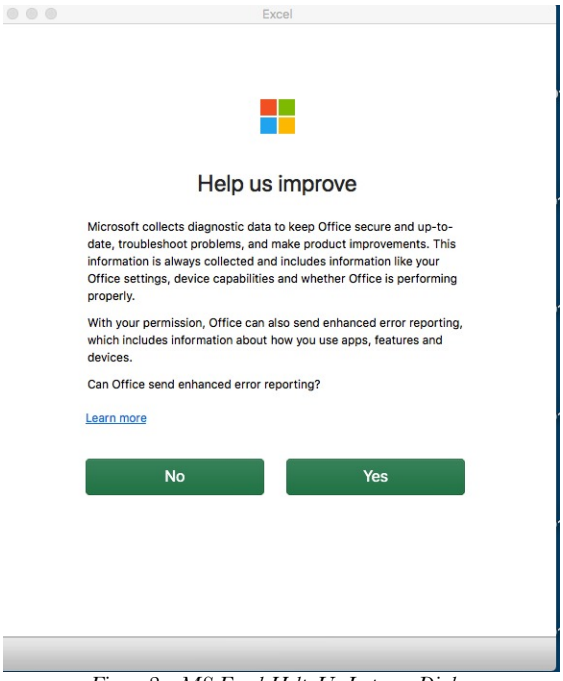

Figure 2 - MS Excel Help Us Improve Dialog

All of these feedbacks produce a message written in natural language. In order to analyze the feedback gathered from the channels mentioned above, the CrowdRE uses approaches such as linguistic analysis techniques like text-mining, sentiment analysis, irrelevant feedback filtering and structured techniques like bug reports and feature requests. [1]

Classification

A feedback might be a bug request, a description of a problem, description of a bug or an error. Alternatively, a feedback might be a feature request or improvement suggestions on content or functionality.[2] Finally, a feedback can hold different

sentences for both errors and feature requests. In addition, it is also important to classify these feedbacks under related part of the system: functionality requirements, documentation requirements, design requirements etc.

To classify the feedback, NLP techniques are being used. These techniques can describe syntactic information (e.g., part-of-speech tagging, chunking, and parsing) or semantic information (e.g., word-sense disambiguation, semantic role labeling, named entity extraction, and anaphora resolution).[3] Studies shows that no single classifier works best for all review types and data sources.[4]

Furthermore, it is also important to identify the source of the feedback; which will bring us to the second issue about the classification and it is the classification of the stakeholders. For instance, Github may have many positive reviews by the software engineers, but it might not be the case for other professionals when they're try to use Github for version control. Classifying the stakeholders will help to identify use-patterns and common needs. Thus, prioritization of the requirements will be more focused and to-the-point.

Summarizing the Reviews

When we summarize the user reviews, we see that two main approach is being used: "Feature-based summarization" and "Topic-based summarization". [4] In Feature-based summarization, the main questions we want answers to are related to the features of the software. Such as;

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

This kind of feedback helps analysts to prioritize the focused-work for the future releases. The biggest challenge here is to identify the correlation between the user opinions and the features. Users may reveal different positive or negative opinions for many features in a single review. Moreover, they may use sarcasm to state a negative review. For example, a user may state "Great. Now I lost all my data with the update" to indicate e negative experience with the feature in a semantically positive sentence. This is also because that the users tend to give more descriptive feedbacks when the experience is positive. On the other hand, users generally do not use their time to describe a negative feedback and give short, uninformative reviews such as "Worst app ever!". Therefore, analyzing the text-based feedback along with the usage information would lead analysts to better understanding the circumstances of the users.

The only difference of Topic-based summarization is that the review is being analyzed by the topic rather than the software features. These reviews hold more information on higher-level and may not indicate any feedback for specific functionality of the software. For example, the topic can be about the whole release, the price or the quality of the user support. As we can see, in this approach not all the reviews determine the prioritization of the work-list for the future releases, but the business side of the app as well.

Challenge of Motivation

How can we trust a user feedback? How can we be sure that the user is not mumbling and reflecting his/her own true opinion? Is this some boring activity for the user that he/she must participate? These are important questions if we will pour so much effort based on those feedbacks. At some level, we must ensure that users are

motivated, and that our process can rely on their feedbacks.

Motivating the crowd is a big challenge in CrowdRE process. One way to motivate users is to show them the benefit of their feedbacks. For instance, adding the most-wanted features to the product or fixing the bugs would be a preliminary motivation for users to keep giving feedback. This would be an indication of that feedbacks have been taken into consideration. Therefore, community management can be considered as an important tool for motivation.

Gamification is another incentive to be used as a motivational instrument. There are several reasons for people to participate in crowd related online work. For example, internal motivations that has been generated by tasks that allow the participant to be creative and experience autonomy, to develop skills and feel competent, to enjoy pastime or to achieve social recognition; or in some cases external motivations evoked by financial payoffs etc. [5]

3. Monitoring

As we mentioned, many different tools can be used to collect user feedbacks. The health and correctness of these channels & tools are unquestionably important. If there's some sort of sensors that are being used to collect "usage" information, these sensors must be reconfigurable at runtime and automatically replaced when failing, and the context can be better understood through distributed pluggable sensors.

On the other hand, there might be contradictory data when interpreting the feedbacks (for example, only some users might struggle with a feature) which is difficult, but a comprehensive understanding can be obtained by aggregating the data with user feedback from other sources. If there are multiple platform where a product is released on (for example, same app can be used on IOS and Android devices) aggregated data must be monitored and refined carefully.

Moreover, user contributions are neither necessarily based on rationale nor on logical reasoning and are rather subjective. [2] Monitoring the pulse of users, communicating with them and explaining requirements decision to them are important. A stronger participation of the users themselves in addition to using their data is necessary. Similarly, as we mentioned, chunks of crowd data will most likely lead to contradictory contributions. Sometimes users themselves are concerned when accessing the crowd data. Analyzing the data may cause misuse in the sense of extracting personal information of users. On the other hand, it is important to ensure the authenticity of the participation.

4. Legal & Ethical Concerns

User consent is a hot topic of today's digital world. Since CrowdRE activities take huge advantage of public users, the legal & ethical aspects of these users' information must be handled delicately. Let's review these concerns briefly.

- **Abuse of personal information**: Hackers may find an access on the collection of the personal information and use it for some illegal purposes.
- **Privacy**: Privacy can be defined as that the individual ability to control personally information about oneself which consider as the most important thing in digital age.[6] The methods, operations and algorithms used to gather and analyze the crowd data must be hidden as a black-box model in order to protect the privacy of the user.

Another concern about the user's privacy can be considered in the context of digital advertisement. As we all know, digital marketing agencies use these kinds of information in order to create personas and target audiences to show more focused and personalized ads. Considering that a CrowdRE process may easily produce which kind of audience has what kind of needs, we can assume that a bulk information like this can be exploit by giant companies.

5. Conclusion

On this survey, we've tried to understand the concept, process, benefits and concerns of crowd-based requirements engineering. 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. [1]

CrowdRE processes has a wide potential in many domains where software products have many stakeholders whom can produce obtainable usage data and user feedback. 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 (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 evolving "smart" domains such as smart cities, smart health etc. the targeted group of stakeholders is quite large. [1]

Current RE practices are considered "decision-centric" processes because requirements elicitation process mainly based on intuitive knowledge, domain experience of the stakeholders or some logical schemas. In an ideal world, a carefully-crafted combination of computational intelligence and human expertise would be great improvement for the RE process. Changing the mindset in software engineering teams and accepting users as equal stakeholders with potentially good ideas and suggestions is an important cultural challenge.[2]

In CrowdRE, the center of the requirements process shifts from carefully selected experienced stakeholders to group of unknown users. Unquestionably this is good because those users are the reasons for building such software in the first place. However, "distilling" those random reviews into qualified feedbacks is not an easy task. As we saw on this survey, technology can come to our aid for collecting, categorizing and analyzing these feedbacks. But that is not enough; we also need to be more planned and systematic to conduct the better CrowdRE process.

6. Retrospective

As the author of this paper and as a software programmer I've witnessed many project's failure due to incomplete requirements and/or not having any requirement process at all. When we look at the giant software companies it is easy to see that these companies pay huge attention to their requirement processes. In that manner I strongly believe that in any type of software project, requirements should be the initial phase. However, in a "start-up" centric tech-world that we live in, I find crowd-based requirements engineering is very hard to implement and monitor. Many of these start-up companies are concerned about their investors and building the product. In addition, the skill-set of semantic feedback analysis is not a common

strong-muscle for the software -minded entrepreneurs. Therefore, the necessary tools for crowd-based requirements engineering my cost a lot of money. Finally, as we've discussed on this paper, one approach for the CrowdRE is to launch an initial product to the market and gather reviews and feedbacks from the users. If the software is built for the end-users, this might be a good choice. But in this scenario, there would be two different requirements process: in the beginning of the product and during the product's life on the market. It is a great way to see the strengths and weaknesses of the product and/or the future of the product. On the other hand, if the app is not built for end-users, CrowdRE may not be the way to go. For example, if you're building an app for a social media where any user can register and use personally, feedbacks must be taken into account. However, if I were to build a software let's say for two separate system's HTTP communication, the traditional requirements engineering would suffice.

References:

1. E. C. Groen et al., "The Crowd in Requirements Engineering: The Landscape and Challenges," in IEEE Software, vol. 34, no. 2

2. W. Maalej, M. Nayebi, T. Johann and G. Ruhe, "Toward Data-Driven Requirements Engineering," in IEEE Software, vol. 33, no. 1

3. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, "Natural Language Processing (Almost) from Scratch" 12(Aug):2493−2537, 2011.

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

5. Morschheuser, Benedikt & Hamari, Juho & Koivisto, Jonna. (2016). Gamification in Crowdsourcing: A Review. 10.1109/HICSS.2016.543.