Crowdsourcing Software Evaluation

Nada Sherief, Nan Jiang, Mahmood Hosseini, Keith Phalp, Raian Ali
Faculty of Science and Technology, Bournemouth University, UK
{nsherief, njiang, mhosseini, kphalp, rali}@bournemouth.ac.uk

ABSTRACT
Crowdsourcing is an emerging online paradigm for problem solving which involves a large number of people often recruited on a voluntary basis and given, as a reward, some tangible or intangible incentives. It harnesses the power of the crowd for minimizing costs and, also, to solve problems which inherently require a large, decentralized and diverse crowd. In this paper, we advocate the potential of crowdsourcing for software evaluation. This is especially true in the case of complex and highly variable software systems, which work in diverse, even unpredictable, contexts. The crowd can enrich and keep the timeliness of the developers' knowledge about software evaluation via their iterative feedback. Although this seems promising, crowdsourcing evaluation introduces a new range of challenges mainly on how to organize the crowd and provide the right platforms to obtain and process their input. We focus on the activity of obtaining evaluation feedback from the crowd and conduct two focus groups to understand the various aspects of such an activity. We finally report on a set of challenges to address and realize correct and efficient crowdsourcing mechanisms for software evaluation.

Categories and Subject Descriptors
D2.1 [**Requirements/Specifications**], D.2.9 [**Management**]

General Terms
Management, Design, Experimentation, Human Factors.

Keywords
Crowdsourcing, Software Evaluation, Users Feedback.

1. INTRODUCTION
Crowdsourcing is a new form of problem solving, which is typically online and relies on a large number of people for relatively simple tasks perceivable and solvable by non-experts [1]. The tasks could be genuinely interesting for the crowd, e.g. asking the users of a popular social network to choose its new logo from a set of candidate logos, or could require certain incentives to motivate the participants, e.g. asking people to fill in a survey about the design of a newly launched product with the potential to win a draw or a certain number of free samples.

Crowdsourcing is a looser business model in comparison to outsourcing and it requires a less strict recruitment and contracting process [2]. Hence, crowdsourcing is typically used for non-critical tasks and tasks which naturally require input from the general public and where the right answer is not algorithmically computable and is based on people's acceptance and dynamics. This does not mean that the process should be open without limits. The authorization could be formal using some strict form of identity check, e.g. through a link to official emails of staff recruited to do the tasks, or social based on the reputation of members of crowd and how they did in past studies [3,4].

Crowdsourcing could be applied for simple atomic tasks and also for complex tasks, that is, those that require the aggregation of partial solutions. It could also be extended to cover those cases in which a collective intelligence, called wisdom of crowds [5], could emerge out of decentralized local knowledge and small tasks done separately by individuals and groups.

The stakeholders of software evaluation include the users who utilize the software as a means to reach their requirements. Users would need to be given a voice, perhaps continuously, on evaluating the extent to which certain software meets their expectations and what improvements they would like to see in it [6]. This should not only be done with an elite group of users at one stage, typically at the development stage, but as a longtime activity which continues to occur at runtime. This will keep the knowledge about the crowd perception of software up-to-date and inform the decision whether to adapt the software or introduce new changes for its next release.

In this paper, we discuss the potential benefits of crowdsourcing for software evaluation and settings in which this paradigm is a natural solution. We focus on a specific aspect, which is the way to obtain the crowd evaluation via feedback. To understand how to design the structure of feedback and its acquisition method, we conduct tow focus groups and identify a number of features to develop in that area. We also elaborate on a number of research challenges to address for enabling a crowd-based software evaluation. The ultimate goal of this proposal is to maximize the efficiency of software evaluation and its scalability to cater for complex and highly variable systems where the power of the crowd could become a great aid.

2. CROWDSOURCING EVALUATION
Traditional methods of software evaluation heavily rely on developers and often recruit an elite group of users selected to be representative of a wider set of potential users. Also, current methods would be limited in predicting and simulating the actual context of use especially for computing paradigms with inherent high variability and dynamicity of their context such as Mobile Apps and Cloud Computing. Thus, traditional evaluation, typically developers-led, could benefit from participatory approaches where users, individually or in groups, lead the evaluation process and provide the evaluation knowledge. This is analogous to the famous shift in higher-education from teacher-directed learning to student-centered learning [19]. This shift was mainly proposed to cater for the rapid growth of accessibility to various sources of knowledge and the diversity of interests of learners. Similarly, software evaluation can engage users not only as participants in the empirical evaluation but also in defining new quality attributes which have not been thought of by developers.

Such crowdsourcing-based approach has various potential benefits including the following:

- Evaluation in real context, i.e. evaluating software when users are using it in practice and out of labs.
- Validating highly-variable software, potentially with reduced cost and minimized time. The access to a large crowd enables fast and scalable evaluation.
- Maintaining the evaluation knowledge up-to-date.
- Access to a wider and diverse set of users and contexts of use unpredictable by analysts.
- Evolving the evaluation process itself, e.g. by introducing new quality attributes and requirements.
- Forming communities of interests and introducing new styles of use and, hence, preferences.

Despite the potentials of crowdsourcing, coming essentially from an easy and broad access to the crowd, the establishment of correct and efficient crowdsourcing platforms is a challenging problem mainly because of the same reasons of its potentials, i.e. the high openness and large scale [3]. Most of the existing studies in crowdsourcing in general, and those exploiting the paradigm for software evaluation, e.g. [14] and [15], advocate the use of the paradigm and use commercial platforms, such as MTurk (https://www.mturk.com/). However, the literature is still limited in providing engineering approaches and foundations to develop crowdsourcing platforms for software evaluation.

The peculiarities of software evaluation might not be accurately tackled when relying on existing general-purpose crowdsourcing platforms and, hence, in this paper we try to explore one of the challenges for crowdsourcing platforms expressly tailored to software evaluation; the obtainment of users feedback of their evaluation of software. Currently, the design and conduct of feedback acquisition are heavily reliant on developers' creativity. We still need to investigate and devise systematic approaches when designing feedback requests [20] and aid developers with proper tools. The following section conducts an empirical study as a first step to address this challenge.

3. METHOD

We took an empirical approach by conducting a multi-session focus group study, which is a popular technique of qualitative research in software engineering [21]. The main purpose of this focus group was to elicit requirements from various stakeholders to understand how crowdsourcing should be practiced in terms of feedback gathering. It was also used to explore the opportunities to use crowdsourcing mechanisms to obtain user feedbacks during software development.

3.1 Sessions

The focus group consisted of two separate sessions. A same set of questions were used in each session with different combinations and focuses (Table 1).

Table 1. Focus group session settings

Sessions	Participants	Purposes
1	Developers who gathered user feedback or got involved in feedback gathering in the past	Channels, forms, expectations
2	Regular software users who provided feedback in the past	Channels, motivations, concerns, experience

Both junior and senior software developers were invited to join the first session where the emphasis of this session was to understand how software developers normally gather user feedback, how they think a good feedback should be structured and how they collaborate and communicate with users in the development as this could inform the way we design feedback requests. The second session was conducted with regular software users who are used to providing feedback. The emphasis of this session was to explore the ways that users would like feedback requests to look like, what drives them to provide feedback and their concerns for not getting involved enough and also for being involved more than what they expect. This session was also used to investigate their motivations to take part in projects and learn their experience from that participation.

3.2 Participants

A total of 15 volunteers, 8 males and 7 females aged between 18 and 40, were invited to participate in the two focus group studies. There were 8 participants in the first session and 7 participants in the second session. These participants mainly came from Egypt and UK with various backgrounds ranging from management, student, research and IT and had different experiences in using software and providing feedback. It should be noted that most participants were already familiar with the notion of crowdsourcing and they have used it in the past, not necessarily via software, for simple tasks such as collecting the notes for lectures, using programming forums to get solutions for certain coding and debugging problems, consulting and contributing to some web forums for immigration and visa issues, etc. In addition, we made sure that all are familiar with the concept by showing demos and discussing commercial platforms such as MTurk.

3.3 Procedure

Participants of each session were recruited separately following a pre-selection process to ensure they have similar characteristics. For example, for those developers volunteered for the study, they had to have the experience of gathering user feedback or getting involved in such activities in the past. Similar pre-selection processes were also used in recruiting software users who provided feedback in the past. A moderator was recruited and used for both sessions. The moderator followed a specially designed interview guide to balance the need for natural conversation and focused discussion when conducting the focus groups.

3.4 Analysis

Each session lasted two hours. All conversations were audio recorded and transcribed with consent from participants. They were aggregated and analyzed by using thematic analysis method following the recommendation of six stages of analysis [22].

4. RESULTS

Four thematic areas were formed and 15 themes were identified from the analysis. In clockwise direction, the four thematic areas are: subject, structure, engagement and involvement. The final thematic map is shown in Figure 1 where the number after each theme indicates the number of participants who emphasized it as a relevant aspect to consider. To stay within the page limits of this paper, we omit the codes related to the themes. A thorough analysis of the focus groups and a confirmation of the results with the participants and also a larger sample of users via a quantitative method, such as questionnaire, will be done in our future work.

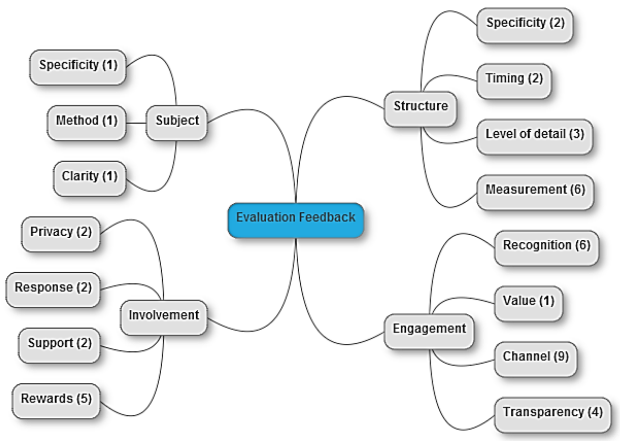

Figure 1. The thematic map

4.1 Subject
Subject refers to the context in which users would like to respond to the feedback request. This includes subject specificity, clarity and feedback method. In detail, participants would like to use a method they prefer to provide feedback such as "*Snapshots, Text, or Audio"* and they would like to give more detailed feedback explanation when they "*reach a clear problem specification*". This means that crowdsourcing software evaluation needs to translate software-related terms to terminology and interfaces users could understand so they feel confident in giving meaningful evaluation feedback.

4.2 Structure
Structure refers to the merits of a feedback which are favorable to be seen, mainly, by software developers. Participants confirmed some common senses such as "*real-time feedback*", "*giving detailed feedbacks*" and "*give feedback to specific problems*". It is also interesting to see many participants thought a feedback would be more useful if it could be discrete in certain ways, e.g. whether it contains "*a group of predefined keywords*", "*structured in a specific way*" and "*feature oriented*". Feedback structure introduces the challenge of balancing between simplicity and expressiveness of crowdsourcing evaluation from users who do not necessarily have a technical background but they are still able to give specific and measured feedback when the question is designed in a way that fits their mind-set and interests of using the system.

4.3 Engagement
It refers to the engagement of users with an evaluation feedback acquisition process via crowdsourcing. In other words, it reflects a strong desire to be part of the value created in this way. A few characteristics of engaged users have been noted. Participants noted some key characteristics of engaged users with the process. First, they would like to be recognized and valued in a way in the participation. For example, they would like to receive "*a friendly confirmation for participation*" and "*more personalized options for feedback*". Second, they thought channel and transparency were important to the process. For example, "*it would increase the users' trust and willingness to give feedback if they know the cycle of how their feedback will be used*", "*the interactions should be very simple*" and "*it encourages users to give feedback if they can meet with analysts to discuss problems in some ways*".

4.4 Involvement
Involvement refers to a variety of "environmental" aspects that encourage users to the process and can directly impact the decisions and activities in using/evaluating the software. Privacy issues were raised by participants as they "*would like to stay anonymous*" and they thought "*it is important if the user can control who is able to see his feedback*". The level of support from the feedback system and the software response based on feedback were also considered important. For example, "*the software's speed of response to my feedback affects my willingness to give feedback*" and "*there can be videos to explain to the users what they can do (in order to provide feedback)*". Furthermore, participants were particularly interested in the rewards mechanism for involvement. For example, "*trying new features or versions for free if the user gave good feedback*".

5. RESEARCH CHALLENGES
As we mentioned earlier, the openness and fast access to the crowd are the main reasons of the high potential of crowdsourcing for software evaluation, especially for large scale software designed to work in dynamic contexts and by wide range of users. At the same time, these same features make its correct implementation challenging. Amongst the various challenges, we here discuss those informed by the focus group results and related to obtaining users evaluation feedback:

- Translation of evaluation criteria and users' judgment to terms and language which are perceivable by users and require minimized facilitation of moderators.
- Criteria to decide the right crowd in terms of engagement and expertise in the requirements and software functionalities to evaluate.
- Ensuring privacy when the evaluation is established as a collaborative activity, e.g. in a forum-like setting, especially when the evaluators play different roles in an organization and their feedback could reveal their work style and personal preferences.
- Aggregation of highly diverse feedback coming from large scale crowd with different interests and expertise.
- Balancing between user-friendliness of interaction methods and precision of collected evaluation feedback.
- Balancing between reward mechanisms, which should incentivize the crowd, and quality of provided feedback.
- Capture of the context of use in which the software was used and evaluated to increase feedback expressiveness.

To summarize, the thematic map shown in Figure 1 suggests that the correct design of the acquisition methods of evaluation feedback should consider the inter-dependencies among the human/crowd factors, the requirements and functionality being evaluated, the evaluation attributes and properties, and the interfaces and structure for expressing the crowd evaluation feedback.

6. RELATED WORK
There are several established approaches where the role of users is central, such as: User centred design [7], User Experience [8], Agile methodology [9], and Usability Testing [10]. All these techniques involve users in the software development life cycle, including the prototyping and evaluation. These techniques can certainly aid the design of crowdsourced online evaluation, but they are expensive and time consuming when used for highly variable software designed to be used by a large crowd in contexts unpredictable at design time.

Recently, more work has been directed towards inventing more systematic methods for representing and obtaining user feedback and making best use of it at runtime during the actual use of software. In [11] and [12] the authors propose a process for continuous and context-aware user input that can be used further in community sharing and inform the developers on how to fix problems and debug the system. In [13], the authors have conducted an empirical study on the users' involvement for the purpose of software evaluation and evolution and validate a set of hypotheses. In [6, 18], the crowd feedback was also advocated for shaping software adaptation as users are powerful to capture and communicate certain information that cannot be monitored by automated means and also cannot be fully specified by designers at design time, yet are necessary to plan and enact adaptation.

In general, when designing an empirical study in software engineering, engaging the right type of participants and appropriate number is always a challenge. Researchers are often required to perform trade-offs to be able to perform the study [14]. The authors in [15] suggest the use of crowdsourcing to address such a challenge. They use Amazon's Mechanical Turk (MTurk) as a tool that allows them to easily manage crowdsourced studies, perform prerequisite qualification tests for filtering participants, ensure privacy, manage payments, and collect results. The authors in [16], used crowdsourcing and MTurk platform in evaluating the usability of a school website. The advantages are claimed to be more participants' involvement, low cost, high speed, and various users' backgrounds, while the disadvantages include lower quality feedback, less interactions, more spammers, less focused user groups. Another study [17] statistically showed that there are no much differences between lab evaluations and crowdsourcing.

A general observation of the current literature is that it treats crowdsourcing as a whole concept without addressing its peculiarities and its different configurations and how to engineer and customize it to fit the type of software evaluation task, the software features being evaluated and the users recruited. Aspects like the interaction style and the model of obtained feedback are generally overlooked. Our work is a first attempt to address that range of challenges.

7. CONCLUSION

This paper proposed a systematic development of a crowdsourcing-based solution to software evaluation. While the concept of crowdsourcing is shown to be promising considering the increasing complexity and diversity of contexts for current systems, there is still a lack of foundations on how to engineer it and ensure correctness and maximize quality. This paper focused on the activity of interacting with users and getting their feedback on software quality as one important step for a holistic approach for crowdsourced software evaluation.

8. ACKNOWLEDGMENTS

The research was supported by an FP7 Marie Curie CIG grant (the SOCIAD Project) and by Bournemouth University – Fusion Investment Fund (the BBB, BUUU and VolaComp projects) and the Graduate School PGR Development Fund.

9. REFERENCES

[1] Howe, J. (2006). The rise of crowdsourcing. Wired magazine, 14(6), 1-4.

[2] Estellés-Arolas, E., & González-Ladrón-de-Guevara, F. (2012). Towards an integrated crowdsourcing definition. Journal of Information science, 38(2), 189-200.

[3] Naroditskiy, V., Jennings, N. R., Van Hentenryck, P., & Cebrian, M. (2013). Crowdsourcing Dilemma. arXiv preprint arXiv:1304.3548.

[4] Stringhini, G., Kruegel, C., & Vigna, G. (2010). Detecting spammers on social networks. In Proceedings of the 26th Annual Computer Security Applications Conference (pp. 1-9). ACM.

[5] Surowiecki, J. (2005). The wisdom of crowds. Random House LLC.

[6] Ali, R., Solis, C., Omoronyia, I., Salehie, M., Nuseibeh, B. (2012). Social Adaptation: When Software Gives Users a Voice. In the proceedings of the 7th Inerantional Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012).

[7] Vredenburg, K., Mao, J.Y, Smith, P., Carey, T. 2002. A survey of user-centered design practice. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'02), ACM, Minneapolis, Minnesota, USA. p. 471-478.

[8] Law, E.L.-C. and P.v. Schaik. 2010. Editorial: Modelling user experience - An agenda for research and practice. Interact. Comput. 22(5): p. 313-322.

[9] Dybå, T. and T. Dingsøyr. 2008. Empirical studies of agile software development: A systematic review. Information and Software Technology. 50(9–10): p. 833-859.

[10] Adikari, S. and C. McDonald. 2006. User and Usability Modeling for HCI/HMI: A Research Design. In Proceedings of the International Conference on Information and Automation. ICIA'06. Shandong, 151 - 154.

[11] Maalej, W., Happel, H-J., Rashid, A. 2009. When users become collaborators: towards continuous and context-aware user input. In Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications (OOPSLA '09). ACM, Orlando, Florida, USA, 981-990.

[12] Knauss, A. 2012. On the usage of context for requirements elicitation: End-user involvement in IT ecosystems. In Proceedings of the 2012 IEEE 20th International Requirements Engineering Conference (RE '12). IEEE Computer Society, Washington, DC, USA, 345-348.

[13] Pagano, D., Brügge, B. 2013. User involvement in software evolution practice: a case study. In Proceedings of the 2013 International Conference on Software Engineering (ICSE '13). IEEE Press, Piscataway, NJ, USA, 953-962.

[14] Kittur, A., Chi, E. H., and Suh, B. 2008. Crowdsourcing user studies with Mechanical Turk. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'08). ACM, Florence, Italy, 453-456.

[15] Stolee, K.T. and Elbaum, S. 2010. Exploring the use of crowdsourcing to support empirical studies in software engineering. In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'10). ACM, Bolzano-Bozen, Italy, 1-4.

[16] Liu, D., Bias, R. G., Lease, M., and Kuipers, R. 2012. Crowdsourcing for usability testing. In Proceedings of the American Society for Information Science and Technology, 49(1): p. 1-10.

[17] Komarov, S., Reinecke, K., and Gajos, K.Z. 2013. Crowdsourcing performance evaluations of user interfaces. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13). ACM, Paris, France, 207-216.

[18] Ali, R., Solis, C., Salehie, M., Omoronyia, I., Nuseibeh, B., Maalej, W. 2011. Social sensing: when users become monitors. In Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering (ESEC/ FSE'11), ACM, Szeged, Hungary, 476-479.

[19] D. Brandes and P. Ginnis. A guide to student-centred learning. Nelson Thornes, 1996.

[20] Almaliki, M., Faniyi, F., Bahsoon, R., Phalp, K., Ali, R. 2014. Requirements-driven Social Adaptation: Expert Survey. The 20th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2014).

[21] Kontio, J., Lehtola, L., and Bragge, J. 2004. Using the focus group method in software engineering: obtaining practitioner and user experiences. In Proceedings of the International Symposium on Empirical Software Engineering (Redondo Beach, CA, USA, August 19 – 20, 2004). ISESE '04. IEEE, New York, NY, 271-280.

[22] Braun, V. and Clarke, V. 2006. Using thematic analysis in psychology. Qualitative Research in Psychology. 3, 2 (Jul. 2006), 77 – 101