Towards Crowdsourcing for Requirements Engineering

Mahmood Hosseini, Keith Phalp, Jacqui Taylor, and Raian Ali

Bournemouth University, UK

{mhosseini, kphalp, jtaylor, rali}@bournemouth.ac.uk

Abstract. Crowdsourcing is an emerging, typically online, distributed problem solving and production model where a problem is solved through the involvement of a large number of people. In this study, we investigate the potential of crowdsourcing in aiding Requirements Engineering. Although the whole area is still to be explored fully, we focus on the Requirements Elicitation stage. In this paper, we survey the literature on crowdsourcing in a variety of disciplines and deduce a set of features which characterize its main constructs; the crowd and the crowdsourcers. We then conduct two focus groups to explore the relationship between these features and the quality of requirements elicited via crowdsourcing. The analysis will lead to a number of hypotheses to confirm and enhance in a future research in the area. The ultimate goal is to systematically develop crowdsourcing platforms for Requirements Engineering and guarantee correctness and maximize efficiency.

Keywords: Crowdsourcing, Expert Survey, Crowdsourced Requirements

1 Introduction

Nowadays, the uncertainty of the technical and social environments, where software systems operate, has radically increased. This phenomenon has been accelerated by the ever-increasing utilization of web-based systems, e.g. Software as a Service, and mobile applications. In essence, this means that system developers and software engineers encounter a wider audience of users, which we call the general public, or the crowd. To cater for the requirements of the crowd, we perhaps need to involve that crowd; hence, the importance of crowdsourcing.

Eliciting the requirements from users is traditionally carried out as a design time activity. Recent approaches advocate that it can be also done at runtime involving the actual users, e.g., via users feedback [1]. Note that within our view of requirements elicitation we not only include the identification of new requirements but also judging the quality of existing software and the alternative configurations it supports to reach the users' requirements. Furthermore, this may involve identifying loci in the requirements model where users encounter problems and require evolution of the system, either autonomously or as a maintenance action for the next release. In addition, users could also identify requirements that are outdated and contexts of uses that affect the quality of certain ways of achieving requirements. In both ways of eliciting requirements (design time and runtime), we advocate that crowdsourcing [2] is a

promising paradigm, particularly when eliciting the requirements for systems which are highly interactive and potentially used by a wide diversity of users in a dynamic, perhaps unknown, context.

Crowdsourcing in requirements elicitation has the potential to increase the quality and comprehensiveness and even the economic feasibility of requirements elicitation. Crowdsourcing gives the software and engineering team access to a wide diversity of actual and potential users. It could be utilized throughout the life-cycle of the software. This would allow developers, potentially, to gain a wider, and more up-to-date knowledge of how users perceive the system role in meeting their requirements, and to understand how that perception changes over time. Traditional elicitation approaches, e.g., interviews and focus groups, are too expensive to deal efficiently with crowd-oriented applications. While Crowdsourcing seems a promising approach, only a very few attempts have been at utilizing this potential, e.g. [3, 4]. Crucially, as a discipline, we have not established engineering approaches to develop Crowdsourcing-based requirements engineering platforms.

The goal of this study is to explore the relationships among different styles of Crowdsourcing and the quality of crowdsourced requirements. To this end, we have surveyed the literature in Crowdsourcing and identified a set of its characterizing features. This has aided the design of two focus groups to explore the relation between these features and the quality of elicited requirements. We have identified a set of observations to confirm in a future study by consulting the opinion of researchers and practitioners in the requirements engineering community.

2 Crowdsourcing for Requirements Engineering

Requirements engineering focuses on the development of a system that best suits the needs of its stakeholders, i.e. the customers who have paid for the system, but since the customers' final goal is for their system to be utilized by end users, their immediate needs should be considered during RE [5]. As a result, it is important to know the end users and recognize their needs during requirements engineering. While requirements elicitation has been well-studied in traditional software systems which are meant for relatively stable business environments (e.g. a library, a bank) or repetitive pattern-like requirements (e.g. meeting scheduler), the new computing paradigms, e.g. the Cloud and the Mobile App, make it hard to make such an assumption. The stakeholders here are the crowd and, hence, it is a valid argument to say that we could rely on the crowd to cater for their needs properly. This means, we need to rethink of requirements elicitation to accommodate the complexity and the scale of the crowd and ensure that we get their requirements efficiently and precisely.

Crowdsourcing is an area of study which has recently gained attention in various scientific fields, such as business, management, environmental sciences, social studies, computing and so on. Crowdsourcing could also aid software development, particularly at the requirements engineering stage, since the crowd could be the potential users of software which is designed to meet their requirements. There are a number of

studies which attempted to utilize the power of the crowd and end-users to solve requirements engineering problems. These include:

Requirements-Driven Social Adaptation: Uncertainty about the role of a software in meeting its requirements in a dynamic context makes the validation of a system a hard and lenghty task. In [1] and [6], Ali et al. propose the notions of Social Adaptation and Social Sensing for a lifelong validation of the difference alternatives of a variable software. The approach is based on acquiring and analyzing the actual users perception on the role of the system in achieving their requirements and its quality. They propose to utilize that to make adaptation decisions.

Feedback-based requirements engineering: users feedback on software could help developers to better understand the requirements of the next release of the system. This feedback could be explicit, e.g. via forums, or implicit, e.g. through monitoring their patterns of use of the software. Pagano and Maalej [7] propose the effects of user feedback on software and requirements engineering teams. They signify the importance of user feedback content on the number of downloads a mobile phone application gets.

Stakeholders' discovery: in complex and dynamic systems, it is hard to identify the set of stakeholders and their roles and expertise and also their requirements. Crowdsourcing here would help identifying a comprehensive set of stakholders from an initial set of stakeholders specified by the analysts. Lim et al. [8] propose that the identification of stakholders relevant to the system is not strightforward and propose a participatory approach to stakeholders identification. The work considers the set of stakeholders as a social network. The analysts could know only few members who will then recommend more members to the analysts and so on.

Requirements Identification: in software paradigms like Cloud Computing and Mobile Apps, the users set is highly diverse and unpredictable. This means relying on an elite group of users to understand what functionality and quality attributes to meet in the software is limited and also costly. We could harness the power of the crowd to understand their requirements as part of the requirements elicitation stage. CrowdREquire [3] is an example of initiatives where the concept of crowdsourcing was advocated for requirements elicitation.

Empirical Validation: Validation and users testing for implemented systems share the same difficulties as mentioned above for requirements elicitation and for requirements-driven adaptation in the sense that it is costy and lengthy and it often leads to results which are valid only temporarily, especially in a dynamic environment. This is true as users might not maintain the same opinion when time passes due to the emergence of competitive solutions and the use of software in contexts which were not thought of at the engineering stage. There has been also some research on the use of crowdsourcing for empirical studies in software engineering [9]. While this was not meant for a particular activity in softwae engineering, the empirical nature of requirements elicitation and validation would suggest investing on developing crowdsourcing platofrms for empirical studies so that we harness the pwor of the rowd to cater for its scale and complexity.

3 How to Crowdsource Requirements Elicitation: Initial Results

In this section, we investigate the potential impact of the different settings of crowdsourcing on the quality of crowdsourced requirements. To this end, we followed a three-steps methodology and report on the results of the first two phases.

In the first step, we studied the literature on crowdsourcing in a variety of disciplines and we identified the features of crowdsourcing according to the different constituents of crowdsourcing, i.e. the crowdsourcer (requirements engineers in the case of requirements engineering) and the crowd (potential software stakeholders including end-users in the case of requirements engineering). This list is illustrated in Table 1.

Table 1. List of crowd and crowdsourcer features

The Crowd	The Crowdsourcer
1. Diversity 1.1. Spatial Diversity 1.2. Gender Diversity 1.3. Age Diversity 1.4. Expertise Diversity **2. Unknown-ness** 2.1. Not Known to Crowdsourcer 2.2. Not Known to Each Other **3. Largeness** 3.1. Number Fulfils the Task 3.2. Number Not Abundant **4. Undefined-ness** **5. Suitability** 5.1. Competence 5.2. Collaboration 5.3. Volunteering 5.4. Motivation 5.4.1. Mental Satisfaction 5.4.2. Self-Esteem 5.4.3. Personal Skill Development 5.4.4. Knowledge Sharing 5.4.5. Love of Community	**1. Incentives Provision** 1.1. Financial Incentives 1.2. Social Incentives 1.3. Entertainment Incentives **2. Opt-out Provision** **3. Ethicality Provision** 3.1. Open Call 3.2. Feedback to Crowd 3.3 No Harm to Crowd **4. Privacy Provision**

In the second step, we held two focus groups, consisting of a mixed group of users and software engineers. We introduced to them the concept of crowdsourcing and its different applications, focusing on software development. The first focus group consisted of seven people; three software developers and four users, four female and three male participants. The second consisted of seven people; four software developers and three users, with one female and six male participants. The 14 participants in

the two focus groups were from 10 countries and with an age range of 23 to 50. Then we asked them pre-prepared questions about the effect of different settings of crowdsourcing on the quality of requirements obtained from the crowd. At the end of each question, the participants were asked to write their comments and answer a correlated question in a pre-prepared questionnaire,. This helped us to get qualitative and quantitative data. We then analyzed the focus group results, both by listening to the recordings and by studying the filled-in questionnaires and the written comments. The results which were obtained from the focus groups are listed in Table 2.

Table 2. List of crowdsourcing features and their quality attributes

Crowdsourcing Features	**Quality Attributes**	**Crowdsourcing Features**	**Quality Attributes**
Largeness	More accuracy, Less biased information, Saturation, More difficulty in coordination	Diversity	Less consensus, More relevance, More creativity, More inconsistency
Anonymous Participation	More honesty, Less credibility	Competence	More correctness, More creativity, More relevance, More willing to participate
Collaboration	More overhead in management, More inconsistency, Possible Dominance of some users, More understandability	Motivation (Intrinsic)	More correctness, More relevance, More completeness
Volunteering and Open Call	More correctness, More chance of encountering malicious participants	Incentives (Extrinsic)	More motivation to be actively engaged, More participants, Misleading the crowd from true involvement
Opt-out Opportunity	More motivation to be actively involved, More participants, More possibility of carrying on with the task	Feedback	More motivation to engage, Disturbing participants, More influence on next stages, Feeling that participant's idea is important, More willingness to participate in future

In our third step, we aim to collect and analyze the opinion of the experts (researchers and practitioners) in requirements engineering about our findings including those obtained from the focus groups. The final result will be a benchmark for the relationship between different crowdsourcing settings and the various quality attributes of elicited requirements. The expert survey will be conducted as part of the activities of the 20th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2014). The survey could be found in the following URL: http://www.esurv.org?u=CSfRE. Also, a PDF version of the online survey is accessible from the following URL: http://goo.gl/oIVdbt.

4 Conclusion

This paper reports on initial results of investigating the use of crowdsourcing for requirements engineering, with focus on elicitation activity. We reported on the results of the literature review and two conducted focus groups as a preliminary step to design an expert survey. The goal is to devise systematic engineering approaches to develop crowdsourcing for requirements engineering.

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

References

1. R. Ali, C. Solis, M. Salehie, I. Omoronyia, B. Nuseibeh and W. Maalej. 2011. Social sensing: when users become monitors. In *ESEC/FSE11*.
2. D. Brabham. 2008. Crowdsourcing as a model for problem solving: an introduction and cases. *Convergence: The International Journal of Research into New Media Technologies 14* (1): 75-90.
3. A. Adepetu, A. Altaf Khaja, Y. Al–Abd, A. Al–Zaabi, and D. Svetinovic. 2012. CrowdREquire: a requirements engineering crowdsourcing platform. *The 2012 AAAI Spring Symposium Series*.
4. S. L. Lim, D. Quercia and A. Finkelstein. 2010. StakeSource: harnessing the power of crowdsourcing and social networks in stakeholder analysis. In *ICSE'10*: 239-242.
5. S. Kujala, M. Kauppinen, L. Lehtola, and T. Kojo. 2005. The role of user involvement in requirements quality and project success. In *the 13th IEEE International Conference on Requirements Engineering, 2005 Proceedings*: 75-84.
6. R. Ali, C. Solis, I. Omoronyia, M. Salehie, and B. Nuseibeh. 2012. Social adaptation: when software gives users a voice. In *ENASE'12*.
7. D. Pagano and W. Maalej. 2013. User feedback in the appstore: An empirical study. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*: 125-134.
8. S. L. Lim, D. Quercia, and A. Finkelstein. 2010. StakeNet: using social networks to analyse the stakeholders of large-scale software projects. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*: 295-304.
9. K. T. Stolee and S. Elbaum. 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*: 35-38.