Researching Crowdsourcing Software Development: Perspectives and Concerns

Klaas-Jan Stol
Lero—the Irish Software Engineering Research Centre
University of Limerick, Ireland
klaas-jan.stol@lero.ie

Brian Fitzgerald
Lero—the Irish Software Engineering Research Centre
University of Limerick, Ireland
bf@ul.ie

ABSTRACT

Crowdsourcing is an emerging form of 'outsourcing' software development. While there has been considerable research in the area of crowdsourcing in general, very little research has focused specifically on how crowdsourcing works in a software development context, and as far as we know, there have been no published studies of crowdsourcing software development from a customer perspective. Based on a review of the literature, we identified a number of key concerns related to crowdsourcing that are of particular importance in a software development context. Furthermore, we observed a number of recurring key stakeholders, or *actors*, each of whom has a unique perspective on crowdsourcing. This paper presents a research framework that consists of the various combinations of stakeholders and key concerns. The framework can be used to guide future research on the use of crowdsourcing as a 'sourcing' strategy, as well as a means to review and synthesize research findings so as to be able to compare studies on crowdsourcing in a software development context.

Categories and Subject Descriptors

K.6.3 [**Software Management**]: Software development, Software process; D.2.8 [**Software Engineering**]: Management—*Programming teams*; K.4.3 [**Organizational Impacts**]: Computer-supported collaborative work

General Terms

Management

Keywords

research framework, crowdsourcing software development

1. INTRODUCTION

Crowdsourcing is gaining significant attention in the software engineering research [4, 22, 24, 25]. Crowdsourcing has been suggested as a useful approach in GUI testing [13], performance testing [24] and even as a means to recruit participants in empirical studies of software engineering [32]. There is an increasing level of attention to social interactions and networks within software engineering research [3], and 'crowds' are an important aspect of this [6]. In this paper we are in particular concerned with using crowds as an alternative form of sourcing, contrasting it with other forms such as open-sourcing [1], inner-sourcing [30] and traditional software outsourcing. In other words, how can a crowd, or '*unknown workforce*' effectively contribute to the development of a software system?

Much research has focused on general-purpose crowdsourcing platforms such as Amazon's Mechanical Turk (AMT), for instance. However, very little research exists on crowdsourcing software development, in contrast to the topic of crowdsourcing in a more general sense. We argue that there is significant potential in software development through crowdsourcing, but that much research is needed to better understand how to optimally do this. For instance, we recently conducted an in-depth case study of a multinational company that had crowdsourced a software project, but encountered several unexpected challenges in doing so [31].

In order to guide future research on this topic we derived a research framework for crowdsourcing software development. While several authors have proposed taxonomies or research frameworks for crowdsourcing [8, 10, 14, 16, 26, 28, 33], these are not focused specifically on software development, but rather provide general classifications and characteristics of crowdsourcing in a more general context. While such general classifications can be useful when considering crowsourcing software development from a wider perspective, we argue that crowdsourcing as a form of 'sourcing' software requires a dedicated framework to systematically study this topic. This paper proceeds as follows: Section 2 presents our research framework, Section 3 outlines how the framework can be applied and Section 4 concludes this paper.

2. RESEARCH FRAMEWORK

Due to the specific intricacies of software development, crowdsourcing in a software context is different from other contexts, as represented by, for instance, AMT. Clearly, this has implications for what aspects of crowdsourcing should be studied in a software development context. To that end, based on a traditional literature review we identified a number of key concerns in crowdsourcing software development. As this set of concerns represents one dimension of our research framework, we briefly reiterate these in Section 2.1; a more extensive description can be found in ref. [31]. Furthermore, since crowdsourcing involves a number of actors, or stakeholders, it is useful to take different stakeholder perspectives so as to study crowdsourcing software development from different stakeholders' views. We identified three different perspectives, namely that of customers, workers and the platform representing an online 'market place' where customers and workers can meet and interact. These perspectives are further described in Section 2.2. Together, these two dimensions (stakeholders and concerns) define a two-dimensional grid (see Table 1, which can serve as a research framework. Such a framework can help in future research as it defines the boundaries of an area that requires further study [29].

2.1 Concerns

2.1.1 Task Decomposition

Development of a significant software systems cannot be done by a single person in a crowd. In order to benefit from a potentially large crowd, the system should be split up into many small pieces that can be developed in parallel by different developers in the crowd. This raises an old question in software engineering, namely, how should the system be decomposed into smaller modules without causing problems in putting them back together once they are developed [18, 20, 21]. Common questions in software engineering within the scope of decomposition relate to assumptions, interfaces and dependencies.

2.1.2 Coordination and Communication

While task decomposition is mainly concerned with the question of how to decompose a system to be developed into manageable chunks of work, coordination is concerned with the process of managing the dependencies between these activities [23]. Coordination is important to ensure that activities are performed in a timely fashion and that together they achieve the ultimate goal of building a system. To achieve this, communication is needed between the developers and the customer.

2.1.3 Planning and Scheduling

With crowdsourcing, timely delivery of software implementations becomes much more uncertain than when it is developed in-house, or in 'normal' outsourcing scenarios where delivery is subject to a negotiated contract. One potential benefit of crowdsourcing is a quicker delivery as the work can be split up in smaller tasks which can then be executed in parallel (see Section 2.1.1). On the other hand, given that crowdsourcing competitions cannot really be expedited once the deadline is set, it is not possible to intervene to achieve faster delivery. Therefore, important questions in crowdsourcing software development is related to how a timely delivery of a software project can be guaranteed when portions are crowdsourced to an unknown workforce.

2.1.4 Quality Assurance

Some crowdsourcing advocates claim that, given a large number of submissions (from a large enough crowd), some submissions will be of high quality [5, 27], thus addressing a key concern in software engineering. Also, similar to Linus's Law, namely that given a sufficiently large group of people, there is bound to be someone who knows how to fix a certain defect, a similar line of thinking would argue that there is a wide variety of expertise available in the crowd. In other words, whatever the software development task at hand, there is bound to be someone who has sufficient domain expertise to provide a solution to a given software development task.

2.1.5 Knowledge and Intellectual Property

Software development is a knowledge-intensive task, and knowledge sharing and management plays an important part throughout the software development lifecycle [2]. A key difference between in-house development and traditional outsourcing scenarios on the one hand, and crowdsourcing on the other hand is that the latter is characterized by a possibly continuous turnover of workers [11].

2.1.6 Motivation and Remuneration

Motivation and remuneration are topics that have received significant attention in the crowdsourcing literature [9, 12, 15, 17]. Crowdsourcing tasks on platforms such as Amazon's Mechanical Turk, sometimes referred to as 'micro-tasks,' tend to be very short in duration, and only a small remuneration is offered for those, usually less than one US dollar [19]. As software development tasks are much more complex, one can no longer speak of micro-tasks as they tend to be interdependent, long in duration (days/weeks as opposed to seconds/minutes), and requiring a great deal of cognitive effort. Therefore, remuneration for such complex tasks must be significantly higher than micro-tasks. An important consideration for a crowdsourcing customer is to decide on an appropriate remuneration that will attract sufficient participants to a crowdsourcing contest. Furthermore, participants who have a lot of experience with crowdsourcing may have a significant advantage over those who are inexperienced, in that they may be more proficient with a platform, and thus may be more likely to win a crowdsourcing contest. Whether or not this puts off inexperienced participants, to the extent to 'scare them away' would be a concern for a crowdsourcing customer as this reduces participation and may affect the number of solutions offered.

2.2 Perspectives

Crowdsourcing software development generally involves three types of actors [33]: customers, who have software development work that needs to be done; workers, who participate in developing software; and platforms, or brokers, who provide an online market place where customers and workers can meet. Each of these three actors will have a different perspective on crowdsourcing software development. We briefly discuss them below.

2.2.1 Customers

Customers, or requesters, are organizations or individuals who have software tasks that need to be done. Customers can have different motivations for crowdsourcing; for instance, an organization may temporarily want to increase their workforce. By 'outsourcing' some of the work to the 'crowd,' an organization can become quite flexible in scaling up and down their production capacity.

2.2.2 Workers

Workers are the individuals who perform the work–they develop the chunks of software that are 'outsourced' by a customer. Little is known about who these workers and there are many open questions that would be of great interest. Some initial work has been done on workers' motivation, but we need a better understanding of why participants engage in this 'insecure' form of employment. Furthermore, while 'normal' development jobs typically require certain levels of formal education, little is known about the background of crowdsourcing workers. Other considerations include the reliability of workers, both in terms of customers being able to deliver products in a timely fashion, as well as careful consideration of product specific knowledge and IP.

2.2.3 Platforms

Platforms provide an online 'marketplace' for workers and customers to meet. The largest platform for crowdsourcing software development is TopCoder, with a developer community of more than 600,000 as of January 2014. However, there have also been indications only a small fraction of its registered users are actually participating in software development. Platforms may have different participation models. TopCoder for instance uses a competition-based model–in TopCoder's model, the winner gets paid a certain

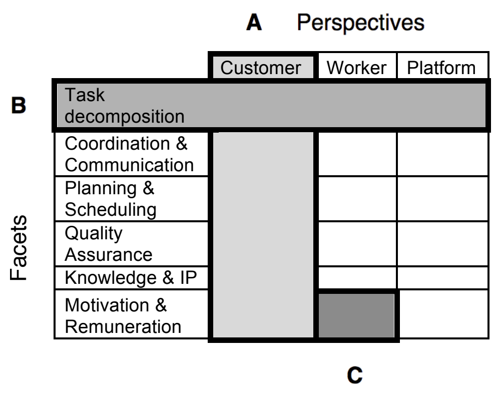

Figure 1: (A) a multi-facet/concern, single-stakeholder-perspective study, (B) a single-facet/concern, multi-stakeholder-perspective study, (C) a single-facet/concern, single-stakeholder-perspective study.

prize, and the runner-up receives 50% of this prize. Furthermore, TopCoder has other mechanisms to incentivize developers to participate. Inexperienced developers may belief they have little chance of winning when competing with experienced participants in a contest. To address this, TopCoder also has a 'Digital Run,' which is a way to earn credits even when a participant does not win (see [31] for more details).

3. USING THE FRAMEWORK

The proposed research framework offers a comprehensive view on crowdsourcing as a form of outsourcing of software development. Each of these six key concerns presents a 'facet' of software development that requires further research to better understand how crowdsourcing can be effectively applied to software development. Furthermore, the three perspectives that have been recognized in the wider crowdsourcing literature [33] offer a useful multi-view lens on those concerns. Combined together, they offer a research 'grid' that can be used to define the scope of future research in this area; Table 1 presents a number of example research questions that we believe are worthy of further study. In what follows below, we present a few examples of how the research framework can be used, but other study designs are possible as well.

One study design for investigating crowdsourcing is a single-perspective, multi-faceted study as we have recently reported in [31]. The design of this study is labelled 'A' in the research framework shown in Figure 1. Our case study [31] describes a number of challenges that one organization encountered.

Another study design can be constructed by applying a multi-stakeholder perspective, single-concern/facet approach; this design is labelled 'B' in Figure 1. Using this study design, the focus is on only one of the concerns/facets identified in Section 2.1, but from different stakeholder perspectives, namely from the customer's, the worker's, and the platform's perspective.

A third study design may focus on one particular facet/concern; for instance, the study design labelled 'C' suggests that motivation is studied from a worker's perspective. Indeed, this is a topic that has already attracted some interest [7, 34].

By systematically reusing this research framework, each 'cell' will become increasingly "populated" by different studies that apply to that cell, by addressing one or more facets, studied by one or more stakeholder perspectives. This in turn will facilitate a more straightforward comparison and synthesis of these studies in secondary studies, such as systematic literature reviews, which have become a widely adopted means to synthesize studies. One challenge in conducting such secondary studies is that comparing studies that were independently designed and grounded in a certain research context is quite difficult.

4. CONCLUSION

While crowdsourcing has been used a technique to perform some essential activities within software development (e.g., testing), very few studies exist of using crowdsourcing as a *sourcing* strategy, whereby the actual *development* of software is crowdsourced and integrated into a final product. In order to guide future research on this topic we have developed a two-dimensional research framework. One dimension comprises six key concerns that we identified from the literature on crowdsourcing, whereas the other dimension comprises three different perspectives from the three main types of stakeholders that can be observed in typical crowdsourcing scenarios. We presented a number of potential research questions that we believe are worthy of further study (see Table 1) and also showed how different study designs can be constructed by selecting one or more concerns and stakeholder perspectives. Besides designing and positioning studies of crowdsourcing software development, the framework can also be used to categorize and synthesize research findings, for instance in future systematic literature reviews on this topic.

5. ACKNOWLEDGMENTS

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero–the Irish Software Engineering Research Centre (www.lero.ie).

6. REFERENCES

[1] P. Ågerfalk and B. Fitzgerald. Outsourcing to an unknown worforce: Exploring opensourcing as a global sourcing strategy. *MIS Quarterly*, 32(2), 2008.

[2] A. Aurum, R. Jeffery, C. Wohlin, and M. Handzic. *Managing Software Engineering Knowledge*. Springer, 2003.

[3] A. Begel, J. Bosch, and M. A. Storey. Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder. *IEEE Softw.*, 30(1), 2013.

[4] A. Begel, J. D. Herbsleb, and M.-A. Storey. The future of collaborative software development. In *CSCw*, 2012.

[5] E. Bonabeau. Decisions 2.0: The power of collective intelligence. *MIT Sloan Manage Rev*, 50(2):45–52, 2009.

[6] A. Bozzon, M. Brambilla, S. Ceri, M. Silvestri, and G. Vesci. Choosing the right crowd: Expert finding in social networks. In *Proc. EDBT/ICDT*, 2013.

[7] D. C. Brabham. Moving the crowd at istockphoto: The composition of the crowd and motivations for participation in crowdsourcing application. *First Monday*, 13(6), 2008.

[8] D. C. Brabham. *Crowdsourcing*. MIT Press, 2013.

[9] D. Chandler and A. Kapelner. Breaking monotony with meaning: Motivation in crowdsourcing markets. *Journal of Economic Behavior & Organization*, 90:123–133, 2013.

[10] L. B. Chilton, G. Little, D. Edge, D. S. Weld, and J. A. Landay. Cascade: Crowdsourcing taxonomy creation. In *CHI*. ACM, 2013.

[11] L. Dabbish, R. Farzan, R. Kraut, and T. Postmes. Fresh faces in the crowd: Turnover, identity, and commitment in online groups. In *CSCW*, 2012.

Table 1: Research Framework to Study Crowdsourcing Software Development and Example Questions.

Concern	Customer	Worker	Platform
Task decomposition	How to effectively decompose a system for development by the crowd?	How to effectively contribute to development of a large system without being a formal member of a team?	How can platforms assist customers in decomposing their software projects into manageable chunks of work?
Coordination & Communication	How to coordinate a group of unknown developers without a direct line of communication?	Can workers collaborate and communicate with other developers in the crowd? How to effectively communicate with a customer?	What mechanisms can platforms put in place to facilitate coordination and communication between workers and customers?
Planning & Scheduling	How can a customer ensure a timely delivery of the product given that solutions are coming from an unknown workforce and thus a schedule cannot be enforced?	How can workers manage to spend their time effectively in tending to different customers?	How can platforms help customers in 'predicting' and ensuring timely delivery of a project based on previous data?
Quality Assurance	How to ensure that the quality of deliverables from the crowd is sufficient?	What are best practices to effectively satisfy customers?	What mechanisms should be in place to ensure that the crowd submits high quality solutions?
Knowledge & IP	How to balance knowledge sharing and protection of IP? How to transfer knowledge to "short-term" workers who represent an unknown workforce?	What knowledge is needed to contribute in a meaningful way?	What mechanisms should be in place for effective knowledge sharing between customers and workers?
Motivation & Remuneration	How to ensure that developers are motivated to participate and submit solutions?	Why do developers participate? Why participate when there is little chance of winning?	What mechanisms can platforms offer to incentivize potential contributors to participate?

[12] D. DiPalantino and M. Vojnovic. Crowdsourcing and all-pay auctions. In *10th Conf. Electronic Commerce*, 2009.

[13] E. Dolstra, R. Vliegendhart, and J. Pouwelse. Crowdsourcing gui tests. In *6th International Conference on Software Testing, Verification and Validation*, 2013.

[14] L. B. Erickson, I. Petrick, and E. M. Trauth. Organizational uses of the crowd: Developing a framework for the study of crowdsourcing. In *SIGMIS-CPR*, 2012.

[15] S. Faridani, B. Hartmann, and P. G. Ipeirotis. What's the right price? pricing tasks for finishing on time. In *AAAI Workshop on Human Computation*, 2011.

[16] L. Hetmank. Components and functions of crowdsourcing systems–a systematic literature review. In *11th Int'l Conf. Wirtchaftsinformatik*, 2013.

[17] J. J. Horton and L. B. Chilton. The labor economics of paid crowdsourcing. In *Conference on Electronic Commerce*, 2010.

[18] P. G. Ipeirotis and P. K. Paritosh. Managing crowdsourced human computation. In *WWW*, 2011.

[19] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *ACM SIGKDD Workshop on Human Computation*, pages 64–67, 2010.

[20] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut. Crowdforge: Crowdsourcing complex work. In *Proc. ACM Symposium on User Interface Software and Technology*. ACM, 2011.

[21] A. Kulkarni, M. Can, and B. Hartmann. Collaboratively crowdsourcing workflows with turkomatic. In *CSCW*, 2012.

[22] T. D. LaToza, W. B. Towne, A. van der Hoek, and J. D. Herbsleb. Crowd development. In *Proc. CHASE*, 2013.

[23] T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Comput Surv*, 26(1), 1994.

[24] R. Musson, J. Richards, D. Fisher, C. Bird, B. Bussone, and S. Ganguly. Leveraging the crowd: how 48,000 users helped improve lync performance. *IEEE Softw.*, 30(4), 2013.

[25] F. Pastore, L. Mariani, and F. G. Crowdoracles: Can the crowd solve the oracle problem? In *6th Int'l Conf. Software Testing, Verification and Validation*, 2013.

[26] A. C. Rouse. A preliminary taxonomy of crowdsourcing. In *Australasian Conf. Information Systems*, 2010.

[27] E. Schenk and C. Guittard. Crowdsourcing: What can be outsourced to the crowd, and why?, 2009. HAL Working Papers.

[28] E. Schenk and C. Guittard. Towards a characterization of crowdsourcing practices. *J Innovation Economics*, 1(7), 2011.

[29] A. Schwarz, M. Mehta, N. Johnson, and W. Chin. Understanding frameworks and reviews: A commentary to assist us in moving our field forward by analyzing our past. *Database Adv Inform Syst*, 38(3), 2007.

[30] K. Stol, P. Avgeriou, M. Babar, Y. Lucas, and B. Fitzgerald. Key factors for adopting inner source. *ACM Trans Softw Engineer Methodol*, Forthcoming, 2014.

[31] K. Stol and B. Fitzgerald. Two's company, three's a crowd: A case study of crowdsourcing software development. In *36th International Conference on Software Engineering*, 2014.

[32] K. T. Stolee and S. Elbaum. Exploring the use of crowdsourcing to support empirical studies in software engineering. In *Proc. ESEM*, 2010.

[33] Y. Zhao and Q. Zhu. Evaluation on crowdsourcing research: Current status and future direction. *Inf Syst Front*, April, 2012.

[34] Y. Zhao and Q. Zhu. Exploring the motivation of participants in crowdsourcing contest. In *ICIS*, 2012.