Strategies for Crowdworkers to Overcome Barriers in Competition-based Software Crowdsourcing Development

Alexandre Zanatta
University of Passo Fundo (UPF)
Passo Fundo, Brazil
zanatta@upf.br

Leticia Machado
Federal University of Pará (UFPA)
Belém, Brazil
leticia.smachado@gmail.com

Igor Steinmacher
Northern Arizona University (NAU)
Flagstaff AZ, USA
igor.steinmacher@nau.edu

Rafael Prikladnicki
Pontifical Catholic University of Rio Grande do Sul (PUCRS) Porto Alegre, Brazil
rafael.prikladnicki@pucrs.br

Cleidson R. B. de Souza
Federal University of Pará (UFPA)
Belém, Brazil
cleidson.desouza@acm.org

ABSTRACT

Crowdsourcing in software development uses a large pool of developers on-demand to outsource parts or the entire software project to a crowd. To succeed, this requires a continuous influx of developers, or simply crowdworkers. However, crowdworkers face many barriers when attempting to participate in software crowdsourcing. Often, these barriers lead to a low number and poor quality of submitted solutions. In our previous work, we identified several barriers faced by crowdworkers including finding a task according to his/her abilities, setting up the environment to perform the task, and managing one's personal time. We also proposed six strategies to overcome or minimize these barriers. In this paper, these six strategies are evaluated questioning Software Crowdsourcing (SW CS) experts. The results show that software crowdsourcing needs to: (i) provide a system that helps matching tasks requirements and crowdworker's profile; (ii) adopt containers or virtual machines to help crowdworkers set up their environment to perform the task, (iii) plan and control crowdworkers' personal time, and (iv) adopt communication channels to allow crowdworkers to clarify questions about the requirements and, as a consequence, finish the tasks.

CCS CONCEPTS

•Software and its engineering •Collab. in software development

KEYWORDS

Software crowdsourcing, barriers, strategies

ACM Reference format:

Alexandre Zanatta, Leticia Machado, Igor Steinmacher, Rafael Prikladnicki, Cleidson R. B. de Souza. Strategies for Crowdworkers to Overcome Barriers in Competition-based Software Crowdsourcing Development In *Proceedings 13th International Workshop on Cooperative and Human Aspects of Software Engineering, Seoul, South Korea. ACM, New York, NY, USA, 4 pages.*

1. Introduction

Software crowdsourcing development (SW CS) depends on a large pool of potential developers on-demand to outsource parts or the entire software project to a crowd [1]. In general, given the characteristics of SW CS projects, tasks represent the starting point of this model. A task is the work unit made available on a crowdsourcing software platform that represents the clients' needs or problems. In competitive SW CS a client requests a task and pays for its completion [2].

ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7963-2/20/05…$15.00
https://doi.org/10.1145/3387940.3392243

Hosseini *et al.* [3] comment that crowdworkers are the key actors in crowdsourcing, however, if crowdworkers dropout or become inactive, this model can collapse [1]. In a previous work, Zanatta *et al.* [4] identified eight main barriers faced by a group of onboarding crowdworkers while attempting to contribute (submit tasks) in the competition-based software crowdsourcing platform Topcoder. We also identified potential solutions to overcome such barriers based on a literature review and, on crowdworkers' suggestions [4], [5], [6]. Table 1 lists these strategies.

Table 1: Strategies to support crowdworkers

#	Strategies	LR	C
S1	Providing a system that helps matching tasks requirements and crowdworkers' profile.	X	X
S2	Decomposing complex tasks into smaller tasks (micro-tasks).	X	-
S3	Using a container or virtual machine to help set up the environment to perform the task.	X	X
S4	Planning and controlling crowdworkers' personal time.	X	-
S5	Using communication channels (chats or forums) to collaborate to understand and perform the task.	X	X
S6	Using auto-translation mechanisms (machine translation) to maximize the comprehension of the task and communication	X	-

Legend: LR – Recommendation from Literature Review
C – Recommendation from participants of case studies.

The contribution of this paper is an evaluation of these strategies that was achieved by asking software crowdsourcing experts. In summary, our research question is: "How do crowdsourcing experts perceive the suggested strategies to support crowdworkers overcoming the onboarding barriers in competitive software crowdsourcing?"

2. Research settings

For data collection, we designed a web-based questionnaire to be distributed to SW CS experts. This survey was validated using the Collingridge method [7]. First, we evaluated our questions using a group familiar with SW CS and we reviewed using an expert on questionnaire construction. In the end, our survey contained approximately 25 open-ended and closed-ended questions divided into 2 sections. The first section contained questions about our strategies on how to minimize the barriers, while the second section asked information about the participants.

Second, we ran a pilot test on a subset of 3 SW CS experts to improve the questionnaire. Third, we collected and cleaned the

data. The questionnaire was sent to 91 experts in SW CS. We considered expert practitioners and researchers in crowdsourcing who were members of the program committee of a specialized event called "Workshop on Crowdsourcing in Software Engineering (CSI-SE)." We also invited the authors who had papers published in CSI-SE between 2014 and 2018. After three months, we received answers from 14[1] respondents, a 15.4% response rate. This study was performed between January and April 2018. In all phases, all participants voluntarily agreed to join the study. The complete survey is available at https://goo.gl/forms/Ys2HomFkT56oItW03. We analyzed the data guided by the procedures and techniques of qualitative research.

3. Results

To preserve experts' identity, we assigned an ID to the participants. Most of the software crowdsourcing experts (85%) had between two and five years of experience in software crowdsourcing.

When asked to provide their perceptions about Strategy 1 (S1 on Table 1), four participants reported that inappropriate task-worker matching might harm the quality of the submissions. Their answers are presented below:

> P1: "*Tools or systems are welcomed to reduce the overhead of selecting tasks."*
> P6: *"Finding appropriate tasks certainly seems important"*
> P10: "*Matching task with expertise is an important part of a crowdsourcing platform."*
> P13: "*Some support for crowdworkers in finding suitable tasks is very useful. But I only selected "3" at this question because I think such task selection/ recommendation should be part of crowdsourcing systems be default rather."*

P14 believes that the crowdsourcing platform should facilitate crowdworkers finding a task to start with, i.e., this should be something already embedded in these platforms instead of a "new" feature. Another participant made a parallel between this strategy and the fact that students rate learning as an important motivation for joining in Open Source Software (OSS).

> P14: "*In Open Source projects, newcomers who search for tasks (instead of approaching the project with a task at hand already) are often students. Students also rate learning as an important motivation for joining. Thus, the recommendation sounds very reasonable."*

When asked to provide their perception about Strategy 2 (S2) four participants agreed that decomposing a task was a good strategy but they commented that this can be very difficult to do.

> P1: "*That's what we do in a software engineering capstone course where I participated. We encourage students to take fine-grained tasks."*
> P6: "*I generally would agree, but it can be really difficult to break down tasks. Depends on the context I guess."*
> P10: "*Decomposing too much would not be good for quality control. A task should take half a day to one week to complete ideally with a relatively independent chunk of work."*
> P11: "*I agree that it is up to the requesters to ensure clarity of the task requirements."*

In conclusion, SW CS experts comment that the platform facilitated them decomposing a task to start with. However, at the same time 2 participants (P6 and P10) mention that this decomposition process might be challenging.

When we asked the participants to provide their perception about Strategy 3 (S3), P1 commented:

> *"I agree with the use of virtual machines or docker containers to overcome the barrier 'It's hard to configure the necessary environment to perform the task.'"*

Some participants mentioned that crowdworkers should use IDEs under the concept of cloud computing to execute the tasks, i.e., these technologies help the preparation of the environment to perform the task. Note that this implies in additional work for the tasks requester who is the one who has to prepare this environment to be shared with the crowd members.

When asked to provide their perception about Strategy 4 (S4) P1 reported that there are some tasks in which it is difficult to evaluate and manage the time required to learn and execute them:

> P1: *"There are tasks in which it is very difficult to be very specific when defining hours or timelines."*

Still, P7 reported that it is necessary to define how much effort or time is required to execute the task, but it depends on the size of the task and the skill of the crowdworkers.

> P7: "*…at least some effort estimation should be provided. If there is a set of smaller tasks, an overall timeline as well as timelines for smaller pieces could be provided."*

Furthermore, P5 mentioned that once the task has been divided into small activities, the next step is to determine the activity duration: how long it will take to accomplish from beginning to finish. The crowdworker can perform a basic analysis to estimate the duration of an activity:

> P5: *"I believe that for highly specialized tasks such as software development, the crowd is required to have a certain degree of expertise, which means they're in a better position than the requester (mostly) to define how much effort / time is required, pretty much like software engineers give estimates to their tasks to management in a work environment."*

In summary, crowdworkers do need previous technical knowledge to be able to estimate the time required to finish a task. Meanwhile, crowdsourcing platforms need to provide an initial estimated time for tasks. This will allow crowdworkers to better manage their time and successfully deliver a task considering that participants use their free time to solve the tasks.

When asked to provide their perception about Strategy 5 (S5) the experts suggested the crowdsourcing platforms must provide (or indicate) support for collaboration mechanisms through communication channels among crowd members.

> P1: *"Kanban boards or slack channels would be a good example [of communication channels]."*
> P10: *"Platform should support some async/sync collaboration channels."*
> P11: *"This shouldn't be necessary...there are tools like Slack that can be used both synchronously and asynchronously."*

[1] There is no consensus in the literature about the size of the sample in qualitative analysis it depends on the research project and the theoretical and conceptual saturation observed by the researcher. *"Thus, to end this Introduction as we began, the answer to 'How many qualitative interviews is enough' is 'it depends'"*. Flick, Uwe in Baker, S. E. e Edwards, R How many qualitative interviews is enough. (2012) http://eprints.ncrm.ac.uk/2273/4,

P12: *"But the recommendation should suggest what channels and what frequency."*

Two other participants (P7, P8) commented that it depends on the type of task; the crowdworker should use different channels to communicate synchronously or asynchronously.

P7: *"This might also depend on the type of task, if there are isolated tasks which might not need any collaboration, others might exist where discussion is needed (e.g., team meetings, or knowledge exchange)."*
P8: *"Depends on the type of Crowdsourced platform and level of the issue raising, I agree that there is a need for different communication channel."*

Only two participants related that they do not agree with the need for a platform with more communication features. Their answers are presented below:

P5: *"Wouldn't it defeat the purpose of crowdsourcing? How would you manage communication of hundreds or thousands of workers? I believe a pure microtask crowdsourced project should keep a minimum of communication and rely on basic voting/score systems for certain decisions. Obviously, this is a personal opinion and it would require further research to evaluate the viability of this approach."*
P13: *"I assume it is meant that in *competitive crowdsourcing*, workers within a group should collaborate (but they do not really need to collaborate with workers outside the group). Additionally, if a project requires much collaboration between workers, then probably the division in micro-tasks was not optimally performed (as it should have resulted in micro-tasks that can be solved independently of each other)."*

In summary, most of the experts think it is fundamental that the SW CS platforms support async/sync collaboration tools to guide and help crowdworkers. However, this is not without challenges. For instance, P12 believes that is important to suggest the frequency and the communication channels to be used. Meanwhile, P7, P8, and P9 argue that this is possible depending on the tasks and how they were decomposed (see Strategy S2). Finally, two participants do not even agree with this strategy.

When asked to provide their perception about Strategy 6 (S6) one participant commented:

P1: *"I think it is a good solution (Google translation integration)."*

Three participants mentioned that using machine translation can be used such as "a starting point", but that is not a replacement for true understanding of the task.

P9: "*I cannot answer this question generally. English is not my native language, yet, I feel confident enough to understand tasks described to me in English, so I wouldn't use auto-translation. Others might need some form of translation. Auto-translation is one option, but certainly not optimal, since it's prone to make mistakes, especially with respect to jargon. In any case, auto-translation is an assistance at best, not a solution to the language barrier."*
P11: *"Machine translation can be a starting point, but is not a substitute for true understanding of task and program requirements."*
P13: *"At first this is a good idea, but auto-translation might misinform the worker, especially because translation learned from general language (web) corpora will not work very well on specialized texts such as those within a software project. Therefore, I am not that convinced about this recommendation. At the same time, I would not know what else to propose."*

In addition, P6, P8, and P10 argued that automatic translation could introduce mistakes and crowdworkers must understand the original language of the task.

P6: *"I think there's a risk here if the translation introduces mistakes … have to be careful with this one.*
P8: Readability is a very important factor that machine cannot help in translation."
P10: *"Machine translation of tasks will introduce a lot of incorrect wording. Ideally, worker should understand the original language of the task."*

English is a universal technical language for SCD projects and crowdworkers must be able to speak the same language for large communities in order to have effective communication and learning. Therefore, crowdworkers need to have enough competence in the English language to express or understand a non-native language to understand the tasks. Meanwhile, another expert suggested having a platform in a different language:

P5: *"Why not just localizing the microtask platform? Besides, as the worker is required to have certain skills to perform the task, IMO certain languages should be required as well."*

4. Discussion

In contrast to distributed development which typically involves developers from the same organization in a more collaborative way, SW CS generally operates on a structure of competitions, projects are transitory or short-lived, unstable and undefined virtual workers. The SW CS nature has a big impact on different parts of a software developer's work including managing time, decomposing and documenting tasks, dealing with cultural and language differences, coordinating and communicating with a diverse and virtual team.

Regarding SW CS nature we divide our strategies into two categories: Technical Strategies (TS) represent overall recommendations relating to using nonspecific technologies and, Personal Strategies (PS) refers to using strategies that will meet crowdworkers individual needs within SW CS context.

4.1. Technical Strategies

The SW CS model taps global talents to work on software development, but it also increases the complexity when one needs to decide which development tasks are more suitable to be crowdsourced as well as setting and orchestrating undefined virtual workers.

Providing a system that helps matching tasks requirements and crowdworkers' profiles can be used to find a task according to crowdworkers' expertise. Dustdar *et al.* [8] present a tool that helps project managers to delegate the tasks to the crowd according to their profile, which can contribute to the understanding of the task by the crowdworker.

Fershtman and Gneezy [9] report that decomposing a task into smaller tasks (micro-tasks) must find a balance between providing sufficient specification and maintaining the details necessary for its understanding. Horton and Chilton [10] state that a reduced global vision of the projects is an intricated source of misunderstandings in *traditional* software development. SW CS should take into this account since it is regarded as more complex

than traditional software development due to its unique characteristics: limited communication, extreme distributed development, high diversity and heterogeneous infrastructure [1].

Projects that can be broken into small modules with clear requisites and limited interdependencies are keener to have success. However, Machado *et al.* [11] and Vaz *et al.* [12] reported the crowd's misconceptions about the task's documentation leading to a low number of submitted solutions during competitive SW CS projects. According to Stol *et al.* [1] to break the project into small modules, one needs to have "clear" requisites, therefore to limit the dependency of tasks can define the success of the project.

Using a container or virtual machine to help set up the development environment for a task can help crowdworkers to engage in the tasks, since preparing the environment to implement the task can demand a high effort by the crowd [5]. This means the crowdsourcing platform should indicate a collaborative environment or repository to maximize crowdworkers' performance. Bari *et al.* [13] mentioned that crowdsourcing platforms should support setting up the environment during the task by the crowd. It should be noted that preparing a computational structure, with specific software and hardware among other aspects, is also a problem for the execution of OSS projects [14].

4.2. Personal Strategies

We speculate that online workers in SW CS suffer from the same limitations about planning and controlling personal time than in traditional software projects. In addition, both groups need to communicate and collaborate to perform the tasks.

Crowdworkers' personal time management can be fundamental to execute the task and a crucial issue to coordinate tasks vs. effort includes processes to understand, exchange information, prepare the environment, execute, and finish the task. Time management refers to the worker's ability to include processes to manage the timely completion of the task available in the platform. Park and Jensen [15] mentioned that users spend a significant amount of time learning about a project before effectively taking part in it. Machado *et al.* [11] reported a study where the participants stated that there was little collaboration among the members before, during and after the execution of the task. Besides that, the communication channels (usually online forums) can extend task documentation to provide technical help to crowdworkers during the competitions. This is necessary because crowdworkers need efficient collaboration mechanisms to create a shared vision about the goal, restrictions and acceptance criteria from projects.

Panichella *et al.* [16] comment that communication between the crowd and the clients is fundamental for SW CS mainly to answer the crowd's questions.

English skills for non-native crowdworkers are a communication problem and an important barrier to tasks performed on competitive crowdsourcing. Steinmacher *et al.* [14] shows that expressing oneself in or understanding a non-native language affects negatively the collaboration among crowdworkers while performing their tasks. As indicated in our results, a possible solution should be using auto-translation mechanisms to facilitate the comprehension of the task and the communication with other crowdworkers [4].

5. Conclusions

In this paper, we presented the evaluation of six strategies aimed to support crowdworkers in overcoming the onboarding barriers that they face in software crowdsourcing platforms. We have limitations and topics that remain for future work. One limitation is that our study focused only on Topcoder platform. In future work, we plan to build a tool that might help matching tasks requirements crowdworker profile.

ACKNOWLEDGMENTS

The authors thanks software engineering experts who participated in the studies, and thanks UPF and PUCRS for supporting this work. This project is partially funded by FAPERGS (project 17/2551-0001/205-4), CNPq (Grants 430642/2016-4, 420801/2016-2 and 311256/2018-0); and FAPESP (Grant #2015/24527-3).

REFERENCES

[1] 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.

[2] Y. Yang, M. R. Karim, R. Saremi and G. Ruhe, "Who Should Take This Task?: Dynamic Decision Support for Crowd Workers," in *10th International Symposium on Empirical Software Engineering and Measurement*, 2016.

[3] M. Hosseini, K. Phalp, J. Taylor and A. Raian, "The four pillars of crowdsourcing: A reference model," in *8th International Conference on Research Challenges in Information Science*, 2014.

[4] A. L. Zanatta, I. Steinmacher, L. Machado, C. R. d. Souza and R. Prikladnicki, "Barriers Faced by Newcomers in Software Crowdsourcing Projects," *IEEE Software*, vol. 34, no. 2, pp. 37-43, Mar 2017.

[5] A. L. Zanatta, L. S. Machado and I. Steinmacher, "Competence, Collaboration and Time Management: Barriers and Recommendations for crowdworkers," in *5th International Workshop on Crowd Sourcing in Software Engineering*, Gotemburgo, 2018.

[6] L. S. Machado, A. L. Zanatta, S. Marczak and R. Prikladnicki, " The Good, the Bad and the Ugly: An Onboard Journey in Software Crowdsourcing Competitive Model," in *4th International Workshop on Crowd Sourcing in Software Engineering*, Buenos Aires, 2017.

[7] D. S. Collingridge and E. E. Gantt, "The quality of qualitative research," *American journal of medical quality*, vol. 23, no. 5, pp. 389-395, 2008.

[8] S. Dustdar and M. Gaedke, "The social routing principle," *Internet Computing*, vol. 15, no. 4, pp. 80-83, Jul 2011.

[9] C. Fershtman and U. Gneezy, "The tradeoff between performance and quitting in high power tournaments," *Journal of the European Economic Association*, vol. 9, no. 2, pp. 318-336, Abr 2011.

[10] J. J. Horton and L. B. Chilton, "The labor economics of paid crowdsourcing," in *11th ACM Conference on Electronic Commerce*, 2010.

[11] L. Machado, J. Kroll, R. Prikladnicki, C. R. de Souza and E. Carmel, "Software crowdsourcing challenges in the Brazilian IT Industry," in *18th International Conference on Enterprise Information Systems*, Rome, 2016.

[12] L. Vaz, I. Steinmacher and S. Marczak, "An Empirical Study on Task Documentation in Software Crowdsourcing on TopCoder," *ACM/IEEE 14th International Conference on Global Software Engineering*, pp. 48-57, May 2019.

[13] E. Bari, M. Johnston, W. Wu and W. T. Tsai, "Software Crowdsourcing Practices and Research Directions," in *Service-Oriented System Engineering* , 2016.

[14] I. Steinmacher, M. A. G. Silva, M. A. Gerosa and D. F. Redmiles, "A systematic literature review on the barriers faced by newcomers to open source software projects," *Information and Software Technology*, vol. 59, no. C, pp. 67-85, Mar 2015.

[15] Y. Park and C. Jensen, "Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers," in *5th IEEE International Workshop Visualizing Software for Understanding and Analysis*, 2009.

[16] S. Panichella, G. Bavota, M. D. Penta, G. Canfora and G. Antoniol, "How developers' collaborations identified from different sources tell us about code changes," in *IEEE international Conference on Software Maintenance and Evolution*, 2014.