Recruiting Software Engineers on Prolific

Daniel Russo
Department of Computer Science
Aalborg University
Copenhagen, Denmark
daniel.russo@cs.aau.dk

ABSTRACT

Recruiting participants for software engineering research has been a primary concern of the human factors community. This is particularly true for quantitative investigations that require a minimum sample size not to be statistically underpowered. Traditional data collection techniques, such as mailing lists, are highly doubtful due to self-selection biases. The introduction of crowdsourcing platforms allows researchers to select informants with the exact requirements foreseen by the study design, gather data in a concise time frame, compensate their work with fair hourly pay, and most importantly, have a high degree of control over the entire data collection process. This experience report discusses our experience conducting sample studies using Prolific, an academic crowdsourcing platform. Topics discussed are the type of studies, selection processes, and power computation.

CCS CONCEPTS

• **Software and its engineering**;

KEYWORDS

Prolific, Crowdsourcing, Human factors, Sample studies

ACM Reference Format:
Daniel Russo. 2022. Recruiting Software Engineers on Prolific. In *Proceedings of The 1st International Workshop on Recruiting Participants for Empirical Software Engineering (RoPES 2022).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Recruiting software professionals for empirical software engineering investigation is a typical barrier for several research questions, especially for independent researchers. Often, such hurdles are bypassed by using students as a proxy of software developers, jeopardizing the generalizability of the study's conclusion [4]. Other options are to recruit developers through professional mailing lists or social media, which implies severe self-selection bias and ethical issues about developers' privacy [6]. On the other hand, scholars employed by big software organizations have direct access to a massive amount of developers (also above 3,500 [5]). However, one company employs all those informants, limiting such studies' generalization. Another typical issue is the sampling strategy. Most software engineering papers rely on convenience sampling [1], representing a significant threat to validity.

In other words, most performed research involving software engineers in our community, so far, suffers from severe limitations typically represented by the low generalizability of their conclusions.

Sample studies have an enormous potential to guide us towards a more theory-driven discipline [17, 18]. In such investigations, the relevant variables in a given population (i.e., of people or systems) or the relation between more characteristics are collected through informants. The ultimate goal is to generalize the findings. Using the taxonomy of Stol & Fitzgerald [18], while research conducted in natural settings, such as field studies, aim to understand a specific research phenomenon, contrived settings focus on causal relations primarily through experiments: neutral settings allow to investigate the world as it is. With sample studies, we can isolate the typical noise of field study and focus on the characteristics and interactions of the identified variables.

Therefore, any investigation that can (i) operationalize research variables (e.g., software quality, size, number of commits, productivity, gender)[1], (ii) have some underlying hypotheses about the relationship of the researched variables, and (iii) have a sufficiently large enough population to ensure well-powered statistical analysis is an ideal candidate for a sample study research design.

Unfortunately, sample studies have three main barriers. First, to ensure the representatives of the selected population, i.e., how to be sure that our informants are indeed software engineers? Second, find enough data points to perform a statistically well-powered analysis. Third, operationalize your variables to be sure that they reflect the investigated phenomenon. Crowdsourcing platforms, such as Prolific[2], directly address the first two barriers. Ensuring construct validity is a more complicated task, and we refer to Russo & Stol for more practical advice [15].

In the following, we will discuss how we addressed the selection process of human participants and dealt with power computations in some of our studies using Prolific [2, 9–14, 16, 19].

2 SELECTION PROCESS

Through Prolific, you have to channel your own survey. By January 2022, Prolific has more than 150,000 active users. We administered our surveys on Qualtrics[3]. Other scholars used free instruments, such as Google Forms [8]. Ralph et al. explain in their paper why the use of Google Forms was not a good idea and, in hindsight,

RoPES 2022, May 17, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

[1] In this exposition, we will not consider construct validity issues.
[2] Prolific is an academic data collection platform: www.prolific.co.
[3] www.qualtrics.com.

they would opt for Qualtrics. On Prolific, the scholar can choose whenever the survey has to be completed using a smartphone, a tablet, or a desktop computer. We allowed only participants to fill the survey using a desktop computer to ensure the participants' focus.

The first selection process happens in Prolific, where participants self-selected themselves by providing several demographics. Although it is not possible to select software engineers directly, pre-screening helps to narrow down the relevant populations. In our past studies, we selected "have knowledge of software development techniques", "have computer programming skills", "use technology at work (e.g., software) at least once a day", and have an "approval rate of at least 95%". The last criterion refers to the level of reliability of Prolific platform members in Prolific past surveys. By January 2022, the Prolific users with such characteristics are more than 15,500. We do not suggest selecting per Industry since software engineers are employed in various sectors, not only software. Similarly, we do not recommend opting for balanced gender distribution since it would bias the results, considering that women represent (unfortunately) a 20%-30% minority [14].

Still, we can not be sure if these candidates are indeed software engineers. They could also have a similar profession, e.g., control engineers. Thus, proper screening is required through a pilot study. Pilot studies are concise studies where specific software engineering competence is tested. To do that, we used a one-minute time-boxed three-question multiple-choice survey. Danilova et al. developed a list of the most compelling question to ask in such pilot studies; we strongly recommend using those [3].

After the pilot study, the researcher can select only the candidates who responded correctly in the given time. Once the cohort has been finalized and selected, they are then invited to take the complete survey. Here, usual recommendations about running surveys apply (e.g., randomization). In addition, we used a number (2-3) quality attention checks to ensure that informants are indeed focused on the survey.

3 POWER ANALYSIS

How many participants are needed in a study to minimize the probability of making Type II (false negative) errors? Power assumptions should be at the very core of every sample study design. To do that, G*Power is a very valuable tool[4]. There, based on the type of analysis, you can perform an *a priori* or a sensitivity analysis. As a reviewer, you can also perform a *post-hoc* analysis to assess whenever the sample size is large enough. Russo & Stol provide a short explanation about the use of G*Power for Structural Equation Modeling analyses.

Although our primary concern is underpowered studies that lead to flawed theories, we are also worried about Type-I (false-positive) errors. This typically happens when the analysis deals with a large number of variables. In such cases, the alpha levels need to be adjusted accordingly [11]. As a suggestion, we do not recommend using a Bonferroni correction since it is overly conservative. By modifying the alpha threshold, the test statistic (e.g., p-value, t-value) also needs to be modified. For example, in Russo et al., we dealt with over 50 variables and performed a high number of tests [11]. In that case, we considered significant only relations that had a p-value smaller than 0.003.

In most cases, computing the minimum sample size is a relatively straightforward task (when dealing with nested or higher-order data, things become more complex). Adjusting the alpha level is not. Thus, our recommendation is to be guided by previous literature, especially methodologically more mature disciplines. The software engineering community also underwent a significant challenge by developing the Empirical Standards [7], which is an excellent starting point to start developing a proper sample study.

[4]https://www.psychologie.hhu.de/arbeitsgruppen/allgemeine-psychologie-und-arbeitspsychologie/gpower.

REFERENCES

[1] Sebastian Baltes and Paul Ralph. 2020. Sampling in software engineering research: A critical review and guidelines. *arXiv preprint arXiv:2002.07764* (2020).

[2] Adrian-Alexandru Cucolas and Daniel Russo. 2021. The impact of working from home on the success of scrum projects: A multi-method study. *arXiv preprint arXiv:2107.05955* (2021).

[3] Anastasia Danilova, Alena Naiakshina, Stefan Horstmann, and Matthew Smith. 2021. Do you really code? Designing and Evaluating Screening Questions for Online Surveys with Programmers. In *Proceedings of the International Conference on Software Engineering*. IEEE, 537–548.

[4] Robert Feldt, Thomas Zimmermann, Gunnar R Bergersen, Davide Falessi, Andreas Jedlitschka, Natalia Juristo, Jürgen Münch, Markku Oivo, Per Runeson, Martin Shepperd, et al. 2018. Four commentaries on the use of students and professionals in empirical software engineering experiments. *Empirical Software Engineering* 23, 6 (2018), 3801–3820.

[5] Denae Ford, Margaret-Anne Storey, Thomas Zimmermann, Christian Bird, Sonia Jaffe, Chandra Maddila, Jenna L Butler, Brian Houck, and Nachiappan Nagappan. 2021. A tale of two cities: Software developers working from home during the covid-19 pandemic. *ACM Transactions on Software Engineering and Methodology* 31, 2 (2021), 1–37.

[6] Nicolas E Gold and Jens Krinke. 2020. Ethical Mining: A Case Study on MSR Mining Challenges. In *Proceedings of the 17th International Conference on Mining Software Repositories*. 265–276.

[7] Paul Ralph et al. 2020. Empirical standards for software engineering research. *arXiv preprint arXiv:2010.03525* (2020).

[8] Paul Ralph et al. 2020. Pandemic Programming: How COVID-19 affects software developers and how their organizations can help. *Empirical Software Engineering* (2020). https://doi.org/doi.org/10.1007/s10664-020-09875-y

[9] Daniel Russo. 2021. The Agile Success Model: A Mixed-Methods Study of a Large-Scale Agile Transformation. *ACM Trans. Softw. Eng. Methodol.* 30, 4 (2021).

[10] Daniel Russo, Paul HP Hanel, Seraphina Altnickel, and Niels van Berkel. 2021. The daily life of software engineers during the covid-19 pandemic. In *Proceedings of the International Conference on Software Engineering*. IEEE, 364–373.

[11] Daniel Russo, Paul HP Hanel, Seraphina Altnickel, and Niels van Berkel. 2021. Developers Task Satisfaction and Performance during the COVID-19 Pandemic. *arXiv preprint arXiv:2107.07944* (2021).

[12] Daniel Russo, Paul HP Hanel, and Niels van Berkel. 2021. Understanding Developers Well-Being and Productivity: A Longitudinal Analysis of the COVID-19 Pandemic. *arXiv preprint arXiv:2111.10349* (2021).

[13] Daniel Russo, Paul H. P. Hanel, Seraphina Altnickel, and Niels van Berkel. 2021. Predictors of Well-being and Productivity among Software Professionals during the COVID-19 Pandemic–A Longitudinal Study. *Empirical Software Engineering* 26, 62 (2021), 1–64. https://doi.org/10.1007/s10664-021-09945-9

[14] Daniel Russo and Klaas-Jan Stol. 2020. Gender Differences in Personality Traits of Software Engineers. *IEEE Transactions on Software Engineering* In Press (2020), 16.

[15] Daniel Russo and Klaas-Jan Stol. 2021. PLS-SEM for Software Engineering Research: An Introduction and Survey. *Comput. Surveys* 54, 4 (2021), 1–38.

[16] Daniel Russo, Klaas-Jan Stol, and Andres R. Masegosa. 2022. From Anecdote to Evidence: The Relationship Between Personality and Need for Cognition of Developers. *Empirical Software Engineering* In Press (2022).

[17] Klaas-Jan Stol and Brian Fitzgerald. 2015. Theory-oriented software engineering. *Science of Computer Programming* 101 (2015), 79–98.

[18] Klaas-Jan Stol and Brian Fitzgerald. 2018. The ABC of software engineering research. *ACM Transactions on Software Engineering and Methodology* 27, 3 (2018), 11.

[19] Niels van Berkel, Jorge Goncalves, Daniel Russo, Simo Hosio, and Mikael B Skov. 2021. Effect of Information Presentation on Fairness Perceptions of Machine Learning Predictors. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–13.