Review Participation in Modern Code Review: An Empirical Study of the Android, Qt, and OpenStack Projects (Journal-First Abstract)

Patanamon Thongtanunam, Shane McIntosh, Ahmed E. Hassan, Hajimu Iida
The University of Adelaide, McGill University, Queen’s University, Nara Institute of Science and Technology.
patanamon.thongtanunam@adelaide.edu.au, shane.mcintosh@mcgill.ca, ahmed@cs.queensu.ca, iida@itc.naist.jp

Abstract—This paper empirically investigates the factors influence review participation in the MCR process. Through a case study of the Android, Qt, and OpenStack open source projects, we find that the amount of review participation in the past is a significant indicator of patches that will suffer from poor review participation. Moreover, the description length of a patch and the purpose of introducing new features also share a relationship with the likelihood of receiving poor review participation.

This paper is an extended abstract of a paper published in the Empirical Software Engineering journal. The original paper is communicated by Jeffrey C. Carver.

I. OVERVIEW

Software code review is a well-established software quality practice of having team members critique changes to a software system. Recently, current software code review practices tend to converge on Modern Code Review (MCR), i.e., a lightweight variant of the software inspection process of the past, that has been widely adopted in open source and proprietary projects. Similar to other collaborative processes, the value that is derived from MCR is dependent on the participation of team members. Recent work has shown that review participation has become a key aspect of MCR practices in terms of the efficiency, effectiveness, and software quality. However, the lightweight MCR process often lacks mechanisms for ensuring a base level of review participation, which the software inspection process of the past achieved through formalized inspection meetings and review checklists.

Despite the importance of review participation, little is known about the factors that influence review participation in the MCR process. An understanding of these factors helps team to better manage the code review process.

In this study [1], we set out to investigate the characteristics of patches that: (1) do not attract reviewers, (2) are not discussed, and (3) receive slow initial feedback. We measure patch characteristics using 20 patch and MCR process metrics grouped along five dimensions, i.e., patch properties, review participation history, past involvement of an author, past involvement of reviewers, and review environment dimensions. To investigate the relationship between the patch characteristics and the likelihood that a patch will suffer from a lack of review participation, we use contemporary regression modelling techniques to estimate the relationship between explanatory variables and the response, while relaxing the requirement of a linear relationship which enables a more accurate and robust fit of the data.

Using data collected from the Android, Qt, and OpenStack open source projects, we empirically study 196,712 code reviews. The results of our study show that our models can identify patches that will suffer from poor review participation with an AUC ranging 0.61-0.76, indicating that our models can estimate a likelihood that a patch will suffer from poor review participation using patch and MCR process metrics. Moreover, we find that the past tendency metrics (e.g., the number of reviewers of prior patches) share a strong relationship with the likelihood that a patch will suffer from poor review participation. In addition, the description length and the number of days since the last modification of the patched files are a strong indicator of a patch that is likely to not be selected by reviewers. The past involvement of reviewers shares an increasing relationship with the likelihood that a patch will be discussed, while the past involvement of an author shares an inverse relationship with the likelihood. Furthermore, a patch is likely to receive slow initial feedback if its purpose is to introduces new features.

Our results highlight the need for patch submission policies that monitor these characteristics. The patches with these characteristics should be given more attention in order to increase review participation. We believe that our results and empirical observations help to support the management of the MCR process and adherence to our recommendations will help development teams improve review participation, which will likely lead to a more responsive MCR processes.

References

Please cite the original paper which is published in the Empirical Software Engineering (EMSE) journal:

Available Online:
http://dx.doi.org/10.1007/s10664-016-9452-6