Jan Jürjens, Kurt Schneider (Hrsg.): Software Engineering 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 117

# Comparing Pre Commit Reviews and Post Commit Reviews Using Process Simulation

Tobias Baum[1] Fabian Kortum[2] Kurt Schneider[3] Arthur Brack[4] Jens Schauder[5]

**Abstract:** Previous studies found that two variations of change-based code review are used in industry: Pre commit review and post commit review. Which one is better in a given situation is not obvious. So we asked: Are there practically relevant performance differences between pre and post commit reviews? How are these differences influenced by contextual factors? To assess these questions, we designed and validated a parametric discrete event simulation model of certain agile development processes. Our analysis indicates that the best choice does depend on the context, but also that there are many situations with no practically relevant difference between both choices. We identified the main influencing factors and underlying effects and condensed our findings into heuristic rules.

**Keywords:** Pre- and Post Commit Review, Agile Software Development, Discrete Event Simulation

In recent years, "change-based" or "modern" [RB13] code review has become the dominant style of code review, especially in open source and agile development. It can be performed before the changes are integrated into the main development branch/trunk ("pre commit review") or afterwards ("post commit review") [Ba16b]. Pre commit review has recently gained popularity [RB13], especially in the form of "pull requests". This sparks discussions about the preferable way to perform reviews. So we asked: *Are there practically relevant performance differences between pre- and post commit reviews? How are these differences influenced by contextual factors?* To answer these questions and derive simple heuristics that can help teams to decide whether a variant is adequate in a concrete situation, we used a simulation model. This model allowed us to study a massive number of different development situations. Methodologically, we started with qualitative empirical observations, induced a simulation model and used this model to deduce quantitative data. We then used this data to validate the model, but also as the starting point of another cycle, inducing heuristics and checking them against the simulation results. Our research question and the design of large parts of the discrete event model is based on recent studies about review processes [RB13, Ba16b]. It was checked in detail by two experienced industrial practitioners. To holistically compare the review process variants, we evaluated them with regard to the three classic target dimensions of software projects: quality, cost/efficiency and time to market. The ensure the adequacy of our measures, we performed a survey among 15 industrial software developers. While creating the model we observed several best-practice guidelines, among others those by Ali, Petersen and

[1] Leibniz Universität Hannover, FG Software Engineering, Hannover, tobias.baum@inf.uni-hannover.de
[2] Leibniz Universität Hannover, FG Software Engineering, Hannover, fabian.kortum@inf.uni-hannover.de
[3] Leibniz Universität Hannover, FG Software Engineering, Hannover, kurt.schneider@inf.uni-hannover.de
[4] SET GmbH, Hannover, arthur.brack@set.de
[5] T-Systems on site services GmbH, Wolfsburg, jens@schauderhaft.de

Wohlin [APW14]. The model is implemented as a discrete event simulation model using the DESMO-J simulation framework [PK05]. After some iterations for verification and refinement, we started to generate data with the simulation model. We used exploratory data analysis and local sensitivity analysis to identify the main effects and to extract heuristics for the choice between pre and post commit review.

The largest difference between pre and post commit reviews exists in terms of cycle time: When there are dependencies between tasks so that one task has to be committed before another can be started, this leads to longer cycle times for pre commit reviews. This difference in cycle time is also the only practically relevant difference for the majority of cases. Other effects that lead to differences with regard to quality and efficiency are connected to task-switch overhead and to the possibility that other developers can find issues as soon as they reach the trunk.

Based on our results, we can summarize our heuristics for practitioners as: If you are currently doing code reviews and feel you have no problem with cycle time or developers being held back by issues that would be found in reviews, don't bother switching from post to pre or vice versa. If you don't have an existing process yet, use pre commit reviews if your team is rather large and cycle time doesn't matter much to you or you can arrange review so that no dependent task waits for the review to be finished. Use post commit reviews if your team is rather small or you have dependencies between tasks that would otherwise increase cycle time. For pre commit reviews in the form of pull requests, also keep in mind the benefits that were out of the scope of our study: Pull requests allow easier contribution to (open source) projects by outsiders [GPD14], and they enforce more process discipline and are an easy way to keep unreviewed changes from being delivered to the customer [Ba16b]. Further information on this study can be found in [Ba16a].

## References

[APW14] Ali, Nauman Bin; Petersen, Kai; Wohlin, Claes: A systematic literature review on the industrial use of software process simulation. Journal of Systems and Software, 97:65–85, 2014.

[Ba16a] Baum, Tobias; Kortum, Fabian; Schneider, Kurt; Brack, Arthur; Schauder, Jens: Comparing Pre Commit Reviews and Post Commit Reviews Using Process Simulation. In: Software and System Process (ICSSP), 2016 International Conference on. 2016.

[Ba16b] Baum, Tobias; Liskin, Olga; Niklas, Kai; Schneider, Kurt: A Faceted Classification Scheme for Change-Based Industrial Code Review Processes. In: Software Quality, Reliability and Security (QRS), 2016 IEEE International Conference on. IEEE, 2016.

[GPD14] Gousios, Georgios; Pinzger, Martin; Deursen, Arie van: An exploratory study of the pull-based software development model. In: Proceedings of the 36th International Conference on Software Engineering. ACM, pp. 345–355, 2014.

[PK05] Page, Bernd; Kreuzer, Wolfgang: The Java Simulation Handbook – Simulating Discrete Event Systems with UML and Java. Shaker, 2005.

[RB13] Rigby, Peter C; Bird, Christian: Convergent contemporary software peer review practices. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. ACM, pp. 202–212, 2013.