MODERN CODE REVIEWS AND ORGANIZATIONAL FACTORS

This briefing reports scientific evidence of 12 studies that investigate organizational factors and their relationship to modern code reviews.

FINDINGS

The difference between core and irregular contributors and reviewers: process aspects.
In 2011, Mozilla migrated to a rapid release cycle, with releases every six weeks instead of, on average, every 10 months. A study has investigated how this change impacted the code reviews process, in particular regarding the review of contributions from core and casual developers [OG1]. In the rapid release cycle, patches were generally reviewed more quickly. Contributions from core developers were rejected faster, while contributions from casual developers were accepted faster than before. Finally, patches by casual contributors were more likely abandoned compared to core contributors. It is therefore important to create a positive experience for first-time contributors to encourage further participation in the project.
Similar observations were made in a study on eight open source projects [OG7]: core developers receive quicker feedback on review requests, and complete the review process faster.
Another study investigated social networks of code contributors and reviews on five open source projects (AngularJS, Docker, Rails, Symfony, and JQuery) [OG4]. Through social network analysis, they identified core and peripheral contributors. They also observed that core contributors receive the fastest review feedback.
Similar observations were made in another study on open source systems (Musicbrainz server on Asterisk) [OG13], where top code contributors were also the top reviewers.
What we think: There is some clear evidence that core developers, at least in open source projects, are treated differently than casual contributors. It makes sense that regular contributors build up a good reputation over time, but also know the particular contribution etiquette well and are therefore fast-tracked in reviews. However, this could potentially lead to bias and lower quality reviews and code. It would be interesting to study the effect of reputation on the fault detection effectiveness in code reviews.
The difference between core and irregular contributors and reviewers: acceptance of contributions.
In a study on eight open source projects (Chromium OS, ITK/VTK, LibreOffice, OmapZoom, Openstack, OVirt, Qt, and Type3), it was found that core contributors are more likely to have their changes accepted to the code base than irregular contributors [OG7].
A potential explanation for this observation was found in another study that investigated 22 open source projects and 1.4 million lines of submitted code [OG10]. The study showed that rejected code is significantly different (due to different code styles) to the project code than accepted code. Furthermore, code that has a different code style is subject to a more thorough review. More experienced contributors submit code that is more conformant to the project's code style.
What we think: There is some evidence that indicates that the experience of contributors affects the acceptance of code in reviews. One of the identified reasons in conformance to coding styles. Ensuring the conformity of code before it is reviewed could therefore reduce the reviewing effort. Furthermore, it would be interesting to study if there exist other objective code quality aspects that could be used to determine the likelihood of contributions acceptance.
The difference between core and irregular contributors and reviewers: agreement between reviewers.
A study on the Qt and Openstack open source projects investigated the consequences of disagreement between reviewers who review the same patch [OG9]. The study found that more experienced reviewers are more likely to have a higher level of agreement than less experienced reviewers. Unsurprisingly, reviews in which reviewers with generally low agreement participate take longer time and have more discussions.
What we think: The authors of OG9 suggest to choose reviewers that usually have a high agreement, in case a review must be done in a brief time. We think this is dangerous advice, since the degree of agreement is not necessarily an indicator of the quality of reviews (even though experience seems to correlate with agreement). Since experience is measured by past review participation, after some time, reviewers might become too trustworthy of each other and become less critical. High agreement among reviewers could also be a warning sign for reviewer bias.
The difference between core and irregular contributors and reviewers: career paths.
A study in the Openstack project investigating the career paths of contributors (from non-reviewer, i.e. developer, to reviewer, to core reviewer) found that (a) there is little movement between the population of developers and reviewers, (b) the turnover of core reviewers is high and occurs rapidly, (c) companies are interested in having core reviewers in their full-time staff, and (d) being a core reviewer seems to be helpful in achieving a full-time employment in a project [OG5].
What we think: We know very little about the progression of reviewers in open source projects or companies. As the career path has been reported in other software engineering studies as a strong motivator, defining a reviewer progression path could be interesting for companies to actively steer and develop.
The effect of the number of involved reviewers on code reviews.
A study on the Mozilla project found that developer participation is a good indicator of review quality, i.e., the more the developers are involved in the discussion of bugs and their resolution, the less likely the reviewers are to miss potential problems in the code [OG12]. The same holds not true for reviewer comments: surprisingly, the studied data indicates that the more reviewers participate with comments on reviews, the more likely they miss bugs in the code they review. A possible explanation is that controversial or complex code changes simply lead also to more discussions.
A study on the Chromium browser also made a counter-intuitive observation: files vulnerable to security issues tended to be reviewed by more people [OG8]. One explanation that was given to this observation is that reviewers get confused about what their role in the review is if there are many reviewers involved (diffusion of responsibility). Similar results were found in a study of a commercial application: the more reviewers are active, the less efficient the review and the lower the comment density (comments per lines of code) [PR12]. In a study including both open source (Apache, Subversion, Linux, FreeBSD, KDE, Gnome, Android, Chrome OS) and commercial (at Lucent, AMD and Microsoft) projects, it was observed that it is general practice to involve two reviewers in a review [PR2].
What we think: There is some evidence indicating that more reviewers not necessarily means higher review quality. Practice observations in large open source and commercial projects indicate that two is a good number. However, the type of change (complexity, size, system impact) should be taken into consideration too. If many reviewers are required, they should have clear responsibilities in the review process.
Information needs of reviewers in code reviews.
A study on the Openstack, Android and Qt projects identified the following information need categories: alternative solutions and improvements, correct understanding, rationale, code context, necessity, specialized expertise, splitability of a change [OG15]. The authors of the study find that some of the information needs can be satisfied by current tools and research results, but some aspects seem not to be solved yet and need further investigation.
What we think: This kind of systematic research is important as it points to hindrances that prevent effective code reviews in practice. Practitioners can use the list to select tools that satisfy these information needs and researchers can investigate means to automate the analysis and processing of the needed information.

Who is this briefing for?

Software engineering practitioners who want to make decisions about code reviews based on scientific evidence.

Where the findings come from?

All findings of this briefing were extracted from the research studies identified through a literature review.

What is included in this briefing?

Summaries of research papers in the form of takeaways for the practitioners.

To access other evidence briefings on code reviews:
https://rethought.se/modern-code-reviews/

For additional information about code review research contact -
deepika.badampudi@bth.se
michael.unterkalmsteiner@bth.se
ricardo.britto@bth.se

References

ID	Title	Link
PR2	Convergent Contemporary Software Peer Review Practices	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.641.1046&rep=rep1&type=pdf
PR12	Investigating The Effectiveness Of Peer Code Review In Distributed Software Development	Please contact one of the authors of this evidence briefing to receive a copy of this paper.
OG1	The Secret Life Of Patches: A Firefox Case Study	http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.259.9506&rep=rep1&type=pdf
OG4	Who Can Help To Review This Piece Of Code?	https://hal.inria.fr/hal-01614583/document
OG5	Reviewing Career Paths Of The Openstack Developers	http://www.win.tue.nl/~aserebre/ICSME2017Perry.pdf
OG7	Impact Of Developer Reputation On Code Review Outcomes In Oss Projects: An Empirical Investigation	http://www.amiangshu.com/papers/Bosu-ESEM-2014.pdf
OG8	An Empirical Investigation Of Socio-Technical Code Review Metrics And Security Vulnerabilities	Please contact one of the authors of this evidence briefing to receive a copy of this paper.
OG9	The Impact Of A Low Level Of Agreement Among Reviewers In A Code Review Process	https://hal.inria.fr/hal-01369055/file/426535_1_En_8_Chapter.pdf
OG10	Will They Like This? Evaluating Code Contributions With Language Models	https://www.sback.it/publications/msr2015.pdf
OG12	Investigating Code Review Quality: Do People And Participation Matter?	http://svn-plg.uwaterloo.ca/~migod/papers/2015/icsme15-OleksiiOlgaLatifa.pdf
OG13	Peer Code Review In Open Source Communities Using Review Board	http://www.amiangshu.com/papers/plat03-bosu.pdf
OG15	Information Needs In Contemporary Code Review	https://www.lucapascarella.com/articles/2018/Pascarella_CSCW_2018.pdf