# THE IMPACT OF MODERN CODE REVIEWS ON PRODUCT QUALITY

**This briefing reports scientific evidence of 14 studies that investigate the impact of modern code reviews on *defect detection or repair*, *code quality*, *detection or fixes of security issues* and *software design*.**

## FINDINGS

**The impact of code reviews on defect detection or repair.**
A study on three large open source systems[1] showed that unreviewed commits have over twice as much chances of introducing bugs than reviewed commits [ISQ1]. Similarly, observations of reviews from the QT open source project show that both defect-prone and actually defective files tend to be reviewed less rigorously in terms of review intensity, participation and time than non-defective files [ISQ7].
Another study [ISQ9] has investigated how code review coverage (the proportion of reviewed code of the total code), review participation (length and speed of discussions) and reviewer expertise affect post-release defects in large open source projects[2]. The findings suggest that reviewer participation is a strong indicator for defect detection ability. While high code review coverage is important, it is even more important to monitor participation of reviewers when making release decisions and select reviewers with the adequate expertise on the specific code. However, these findings could not be confirmed in a project of proprietary software at Sony [ISQ5]. In their context, other metrics such as the proportion of In-House contributions, measure of accumulated effort to improve code changes and the rate of author self-verification contributed significantly to defect proneness [ISQ5].
Defective conditional statements are often the source of software errors. A study of the QT and Openstack projects investigated which changes to conditional statements are introduced by code reviews [ISQ3]. They found that negations in conditionals and implicit indexing in arrays are often replaced with function calls, suggesting that reviewers found that this change leads to more readable code.
*What we think: There is evidence that the effort in modern code reviews is well invested as it helps to detect defects. There is some evidence that the length and speed of review discussions, reviewer expertise and code review coverage are an indicator of review effectiveness, i.e., the probability to find defects. However, the particular employed review process affects the explanatory power of these variables. Finally, static code analysis tools could be improved to detect negations and certain array indexing patterns in conditional statements. That would reduce the reviewing effort as these issues could be detected and fixed before the code is reviewed.*

**The impact of code reviews on code quality.**
Studies were conducted to find the problems fixed by code reviews. A study classified defects types in 468 review comments of four large open source systems[3] [ISQ6]. They concluded that 75% of the defects identified during code review are evolvability type defects [ISQ6]. They also found that code review is useful in improving the internal software quality (through refactoring). Similarly, a larger study [ISQ13] that looked at 1400 changes taking place in reviewed code from two OSS also found that 75% of changes are related to evolvability and only 25% of changes are related to functionality. In addition, they find that 78-90% of the triggers for code changes are review comments. The remaining 10–22% are "undocumented". A study on three large open source systems[1] showed that reviewed commits have significantly higher readability and lower complexity. However, no conclusive evidence was reported on coupling [ISQ1].
The comparison of cost required to produce quality programs using code reviews and pair programming showed that code reviews costs 28% less compared to pair programming [ISQ14].
*What we think: There are a good number of studies that investigate what types of quality problems are fixed by the code review process in open source projects. A large percent of code changes are related to maintenance and evolutions issues and not functionality related issues. Evidence shows that code reviews are effective in identifying design problems and it also costs less compared to pair programming.*

**The impact of code reviews on detection or fixes of security issues.**
A study [ISQ8] analyzed 267,046 code review requests from 10 open source projects. The results indicate that code review leads to the identification of different vulnerability types. The study also indicates that the vulnerabilities identified through the code review process are generally fixed. Experienced developers also write vulnerable code. However, less experienced developers are more likely to write vulnerable code. Employees of organizations sponsoring the OSS project are more likely to have vulnerable code changes [ISQ8].
The experience of reviewers regarding vulnerability issues is an important factor in finding security related problems, as a study on the Chromium browser indicates [OG8].
Another large study [ISQ12] that looked into 3,126 projects in 143 languages, with 489,038 issues and 382,771 pull requests, also has similar findings. The results indicate that code review coverage reduces the number of security bugs in the investigated projects.
A study looked into the language used in code reviews to find if the linguistics characters could explain developers missing a vulnerability [RC4]. The study found that code reviews with lower inquisitiveness (fewer questions per sentence), higher positive or negative sentiment, lower cognitive load and higher assertions are more likely to miss a vulnerability.
*What we think: Although there are only four studies looking at the impact of code reviews on security issues, these studies are quite large and can be considered as strong evidence. Security critical projects should adopt code review to identify vulnerabilities early. The projects should create or adapt secure coding guidelines and code review policies. All code changes, in particular, the changes from less experienced and sponsored employees should be reviewed thoroughly. Creating a dedicated security review team with members that have most knowledge about secure coding can help to understand the security implications.*

*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

**The impact of code reviews on software design.**
A study [ISQ4] investigated how code review practices impact design quality in large open source projects[2]. They found that code review practices can help to reduce the incidence of anti-patterns in software systems [ISQ4].
A high code review coverage (the proportion of reviewed code of the total code) can reduce the occurrence of design anti-patterns such as Blob, Data class, Data clumps, Feature envy and Code Duplication in a software system. The lack of participation (length and speed of discussions) during code reviews has a negative impact on the occurrence of Tradition Breaker, Code duplication, Data class, Feature Envy, God Class and Schizophrenic class.
Similarly, a study [ISQ11] specifically looked for the occurrences of review comments related to five code smells (Data Clumps, Duplicate Code, Feature Envy , Large Class and Long Parameter List) in openstack and wikimedia. They conclude that the code review process did identify the five code smells.
*What we think: Overall there is some evidence that shows that good review coverage and participation are effective in identifying design problems.*

[1] Android, LibreOffice and Scilab
[2] QT, VTK, ITK
[3] Eclipse, mylyn, android and openstack

**References**

| ID | Title | Link |
|---|---|---|
| ISQ1 | Four Eyes Are Better Than Two: On The Impact Of Code Reviews On Software Quality | http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.709.2980&rep=rep1&type=pdf |
| ISQ3 | How Are If-Conditional Statements Fixed Through Peer Code Review? | https://www.jstage.jst.go.jp/article/transinf/E101.D/11/E101.D_2018EDP7004/_pdf |
| ISQ4 | Do Code Review Practices Impact Design Quality? A Case Study Of The Qt, Vtk, And Itk Projects | https://www.researchgate.net/profile/Rodrigo_Morales20/publication/281735036_Do_code_review_practices_impact_design_quality_A_case_study_of_the_Qt_VTK_and_ITK_projects/links/582944ae08ae950ace7035d9/Do-code-review-practices-impact-design-quality-A-case-study-of-the-Qt-VTK-and-ITK-projects.pdf |
| ISQ5 | A Study Of The Quality-Impacting Practices Of Modern Code Review At Sony Mobile | https://dl.acm.org/doi/pdf/10.1145/2889160.2889243 |
| ISQ6 | Feedback Topics In Modern Code Review: Automatic Identification And Impact On Changes | https://pdfs.semanticscholar.org/9ec2/61f63bb3123aea8949ea6697d6e0e42f6fcf.pdf |
| ISQ7 | Investigating Code Review Practices In Defective Files: An Empirical Study Of The Qt System | https://www.researchgate.net/profile/Patanamon_Thongtanunam/publication/306479551_Investigating_Code_Review_Practices_in_Defective_Files_An_Empirical_Study_of_the_Qt_System/links/57be6e7208ae2f5eb32e0293/Investigating-Code-Review-Practices-in-Defective-Files-An-Empirical-Study-of-the-Qt-System.pdf |
| ISQ8 | Identifying The Characteristics Of Vulnerable Code Changes: An Empirical Study | https://dl.acm.org/doi/pdf/10.1145/2635868.2635880 |
| ISQ9 | An Empirical Study Of The Impact Of Modern Code Review Practices On Software Quality | https://link.springer.com/content/pdf/10.1007/s10664-015-9381-9.pdf |
| ISQ11 | Empirical Evaluation Of Code Smells In Open Source Projects: Preliminary Results | https://dl.acm.org/doi/pdf/10.1145/2975945.2975947 |
| ISQ12 | A Large-Scale Study Of Modern Code Review And Security In Open Source Projects | https://dl.acm.org/doi/pdf/10.1145/3127005.3127014 |
| ISQ13 | Modern Code Reviews in Open-source Projects: Which Problems Do They Fix? | https://dl.acm.org/doi/pdf/10.1145/2597073.2597082 |
| ISQ14 | Investigating The Impact Of Peer Code Review And Pair Programming On Test-Driven Development | https://ieeexplore.ieee.org/abstract/document/6950664/ |
| RC4 | Natural Language Insights From Code Reviews That Missed A Vulnerability: A Large Scale Study Of Chromium | https://nuthanmunaiah.github.io/static/assets/natural-language-insights.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. |