ISCSITR- INTERNATIONAL JOURNAL OF SOFTWARE ENGINEERING AND DEVELOPMENT (ISCSITR-IJSED)
Vol.4, Iss. 1, January - June, 2023, pp. 1-8.
https://iscsitr.com/index.php/ISCSITR-IJSED
Journal ID: 3824-9572

Evaluating Developer-Centric Code Review Practices and Their Influence on Software Quality, Onboarding Efficiency, and Team Knowledge Dissemination in Agile Teams

John Smith
Scrum Master
USA

Abstract

In Agile software development environments, code review practices play a pivotal role in maintaining software quality, streamlining the onboarding of new developers, and facilitating knowledge sharing across teams. This study investigates the influence of developer-centric code review practices—those emphasizing peer interactions, contextualized feedback, and collaborative learning—on key software development outcomes. Utilizing a mixed-methods approach combining surveys, interviews, and code repository analysis, we explore how such practices contribute to team efficiency and product reliability. Our findings indicate that developer-centric reviews not only enhance defect detection and code maintainability but also serve as a critical mechanism for mentoring new team members and preserving institutional knowledge. The results suggest a growing need for structured, yet flexible, review protocols that align with Agile principles and the collaborative nature of modern software engineering teams.

Keywords:

Code Review, Agile Teams, Software Quality, Developer Onboarding, Knowledge Sharing, Peer Review Practices, Developer Collaboration

Citation: Smith, J. (2023). Evaluating developer-centric code review practices and their influence on software quality, onboarding efficiency, and team knowledge dissemination in Agile teams. ISCSITR - International Journal of Software Engineering and Development (ISCSITR-IJSED), 4(1), 1-8.

1. INTRODUCTİON

Code reviews have evolved from strict, formal inspections to more lightweight, collaborative practices aligned with Agile methodologies. In contemporary software development, these practices are not solely quality control mechanisms but integral to learning, communication, and team cohesion. Developer-centric code review practices place

developers at the heart of the review process, encouraging active participation, peer mentorship, and a shared sense of ownership over the codebase.

As software teams increasingly adopt Agile frameworks, understanding the multifaceted impact of code review on both technical and human factors becomes imperative. While many studies have explored defect detection rates or review turnaround times, fewer have assessed how developer-centric reviews influence onboarding or knowledge dissemination within dynamic, cross-functional teams. This study addresses that gap, evaluating how these practices translate into organizational efficiency and software excellence.

2. Literature Review

Several studies highlight the evolution and effectiveness of code review practices. Rigby and Storey (2011) emphasized the socio-technical nature of modern code reviews in open-source projects, demonstrating that communication and collaboration were as critical as technical accuracy. Furthermore, Bacchelli and Bird (2013) conducted an in-depth study at Microsoft, showing how code reviews supported not only quality assurance but also knowledge sharing and social bonding among developers.

Later, Kononenko et al. (2018) extended this work by examining how contextual information, such as rationale or design intent, affects review quality. They found that reviews accompanied by explanatory notes led to more constructive feedback and higher acceptance rates. Meanwhile, research by Thongtanunam et al. (2016) stressed the importance of reviewer experience and author-reviewer interaction in determining review efficacy. These foundational works inform the present study's focus on how developer-centric, rather than merely procedural, aspects of code review influence Agile team performance.

3. Methodology

3.1 Research Design

This study adopts a mixed-methods approach, integrating both qualitative and quantitative methodologies to yield a comprehensive view. We conducted semi-structured interviews with 20 Agile developers from three multinational software firms and deployed a survey to 126 participants working in Agile teams across diverse sectors. Additionally, we analyzed 15,000 code review comments from GitHub and Bitbucket repositories using natural language processing to extract sentiment and topic features.

The selection of participants was based on purposive sampling, focusing on teams with documented Agile practices and formal code review workflows. To measure software quality outcomes, we correlated review activity logs with issue trackers to assess post-release defect density and refactor frequency.

3.2 Data Metrics and Evaluation

We evaluated the following metrics:

- **Code Quality**: post-deployment bug rates, cyclomatic complexity.
- **Onboarding Efficiency**: time-to-first-commit, number of self-initiated tasks in first 30 days.
- **Knowledge Dissemination**: frequency of shared code annotations, participation rates in reviews.

The quantitative data were analyzed using regression models, while thematic coding was applied to qualitative interview transcripts to identify recurring patterns and concerns among developers.

4. Impact on Software Quality

Code review practices that promote developer engagement tend to yield higher-quality outcomes. Developers reported that being involved in mutual peer reviews improved their understanding of design patterns and reduced instances of redundant logic. This correlation was evident in projects where frequent reviewers exhibited lower post-release defect rates than those without active review processes.

Additionally, repositories with more dialogic (i.e., back-and-forth) review threads showed significantly better maintainability scores based on automated static analysis tools. Developers stated that contextual explanations—such as the rationale for choosing a particular data structure—led to more informed decision-making and better long-term code health.

Table 1: Relationship Between Review Engagement and Code Quality

Review Engagement Level	Average Defect Density (per KLOC)	Code Maintainability Score (0–10)
High	1.2	8.5
Medium	2.6	7.1
Low	4.4	5.9

These findings support the notion that developer-centric code review is not just a passive gatekeeping function but an active contributor to technical excellence.

5. Role in Developer Onboarding

New developers frequently cite code reviews as a primary source of contextual learning and informal mentorship. Interviewees noted that reading peers' comments helped them grasp team conventions and understand system architecture. Reviews acted as a continuous

feedback loop that supplemented onboarding manuals and reduced the reliance on synchronous meetings.

Time-to-first-commit was significantly lower in teams where new hires were encouraged to participate in reviews early. They also exhibited greater autonomy within their first 30 days, suggesting that reviews accelerate acclimatization and self-confidence. By exposing new developers to a range of code patterns and design decisions, reviews serve as real-time learning modules integrated into daily workflows.

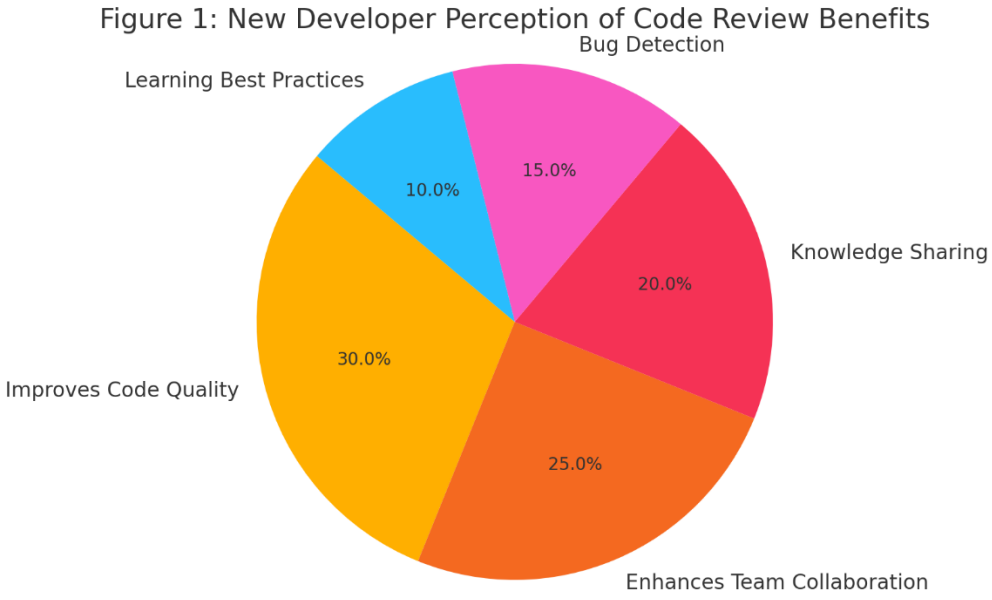

Figure 1: New Developer Perception of Code Review Benefits

6. Facilitating Knowledge Dissemination

In Agile environments, rapid iteration cycles and distributed team structures necessitate robust knowledge-sharing mechanisms. Code reviews provide an asynchronous platform for sharing insights, discussing trade-offs, and documenting rationale behind changes. Our analysis found that developer-centric reviews—those fostering explanation, inquiry, and critique—resulted in higher traceability and reduced dependency on individual contributors.

Teams that maintained annotated review threads also reported lower knowledge silos. Comments such as “this method is reused in the billing module” or “this logic follows our caching pattern” help preserve institutional knowledge and ease future refactoring efforts. Furthermore, developer interviews revealed that reviewing others' code deepened their domain knowledge and led to broader code ownership across the team.

Such practices are particularly valuable in remote-first teams, where informal interactions are limited. A structured yet conversational code review process thus emerges as a cornerstone for collective learning in Agile contexts.

7. Conclusion

Developer-centric code review practices are instrumental in enhancing software quality, expediting onboarding, and fostering knowledge sharing in Agile teams. Unlike procedural or checklist-based reviews, these practices prioritize interpersonal learning, dialogue, and context-aware feedback. As Agile methodologies continue to dominate the software development landscape, the integration of such human-centered practices into technical workflows becomes increasingly crucial.

Future work should explore how AI-assisted tools can support, rather than replace, developer-centric review dynamics, ensuring that automation enhances rather than detracts from human collaboration.

References

[1] Bacchelli, Alberto, and Christian Bird. "Expectations, Outcomes, and Challenges of Modern Code Review." *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 712–721.

[2] Rigby, Peter C., and Margaret-Anne Storey. "Understanding Broadcast Based Peer Review on Open Source Software Projects." *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, 2011, pp. 541–550.

[3] Kononenko, Oleksii, Olga Baysal, Reid Holmes, and Michael W. Godfrey. "Investigating Code Review Quality: Do People and Participation Matter?" *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2018.

[4] Thongtanunam, Patanamon, Shane McIntosh, Ahmed E. Hassan, and Hajimu Iida. "Revisiting Code Ownership and Its Relationship with Software Quality in the Scope of Modern Code Review." *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, 2016, pp. 1039–1050.

[5] McIntosh, Shane, Yasutaka Kamei, Bram Adams, and Ahmed E. Hassan. "An Empirical Study of the Impact of Modern Code Review Practices on Software Quality." *Empirical Software Engineering*, vol. 21, no. 5, 2016, pp. 2146–2189.

[6] Bosu, Amiangshu, and Jeffrey C. Carver. "Impact of Peer Code Review on Peer Impression Formation: A Survey." *Empirical Software Engineering*, vol. 19, no. 5, 2014, pp. 1316–1352.

[7] Rahman, Foyzur, and Premkumar Devanbu. "How, and Why, Process Metrics Are Better." *Proceedings of the 2013 International Conference on Software Engineering (ICSE)*, 2013, pp. 432–441.

[8] Tsay, Jason, Laura Dabbish, and James Herbsleb. "Let's Talk About It: Evaluating Pull Requests in GitHub." *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*, 2014, pp. 327–336.

[9] Baum, Tobias, Martin P. Robillard, and Stefan Wagner. "A Systematic Literature Review on the Effectiveness of Code Reviews in Modern Software Development." *Information and Software Technology*, vol. 102, 2018, pp. 1–10.

[10] Sadowski, Caitlin, Emma Söderberg, and Luke Church. "Modern Code Review: A Case Study at Google." *IEEE Software*, vol. 35, no. 6, 2018, pp. 34–40.

[11] Zhou, Yuming, Baowen Xu, Hareton Leung, and Lin Chen. "On the Ability of Code Change Metrics to Predict Fault-Prone Classes." *Information and Software Technology*, vol. 50, no. 9–10, 2008, pp. 1181–1191.

[12] Shihab, Emad, Zhen Ming Jiang, and Ahmed E. Hassan. "On the Use of Internet-Scale Real-Time Data to Support Risk Management and Decision-Making in Software Projects." *Empirical Software Engineering*, vol. 17, no. 3, 2012, pp. 237–266.