EXPECTATIONS, OUTCOMES, AND CHALLENGES OF MODERN CODE REVIEW

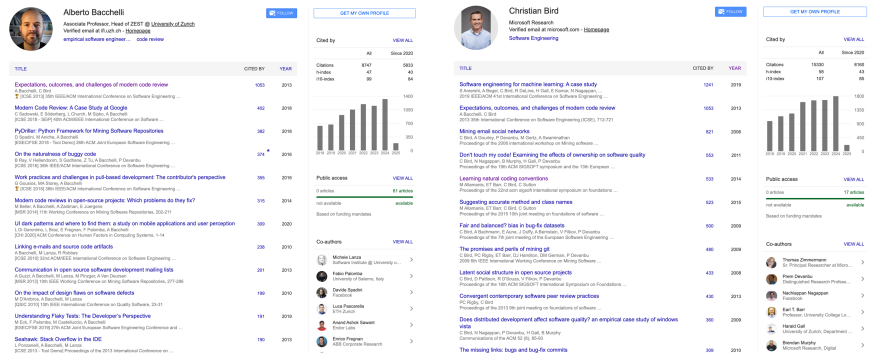
Alberto Bacchelli
REVEAL @ Faculty of Informatics
University of Lugano, Switzerland
alberto.bacchelli@usi.ch

Christian Bird
Microsoft Research
Redmond, Washington, USA
cbird@microsoft.com

Presenter: Haonan Zhang

March 7, 2025

UNIVERSITY OF WATERLOO



Traditional code review process

1. **Planning** – define scope and goals of the review and assign roles
2. **Overview meeting** – author presents and reviewers ask
3. **Preparation** – reviewers analyze the code individually using checklists
4. **Inspection meeting** – All reviewers meet in person
5. **Rework** – the authors fixes the identified problems
6. **Follow-up** – a final review ensures that all problems are addressed

Time-consuming Rigid and expensive Not suitable for rapid development

Modern code review

- **Informal** – no need for long, scheduled meetings
- **Tool-based** – use platforms like CodeFlow and GitHub
- **Flexible** – Developers can review code whenever they are available
- **Fast** – Code can be reviewed and merged in hours rather than days
- **Beyond defect-finding** – knowledge sharing and team discussions
- **Works with CI/CD** – enables fast releases with continuous integration

Code review with CodeFlow

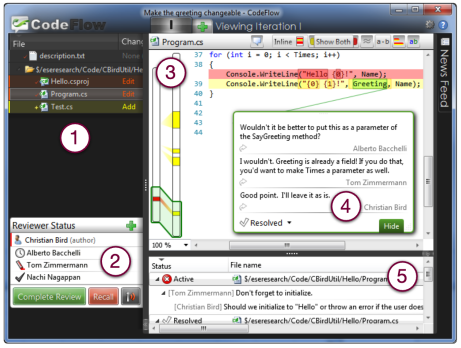
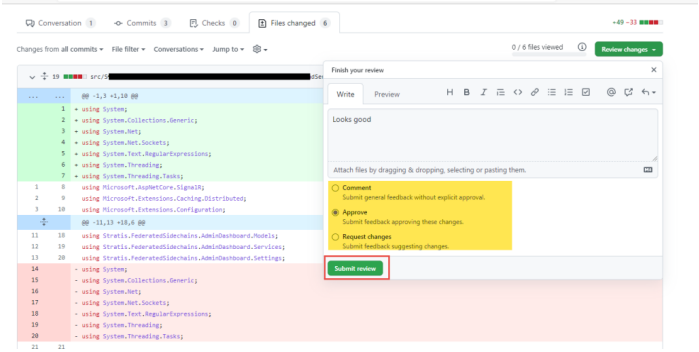


Fig. 1. CodeFlow, the main code review tool used by developers at Microsoft.

Code review with GitHub



Problem to be solved

RQ1: What are the motivations and expectations for modern code review? Do they change from managers to developers and testers?

RQ2: What are the actual outcomes of modern code review? Do they match the expectations?

RQ3: What are the main challenges experienced when performing modern code reviews relative to the expectations and outcomes?

New idea – mixed qualitative and quantitative study

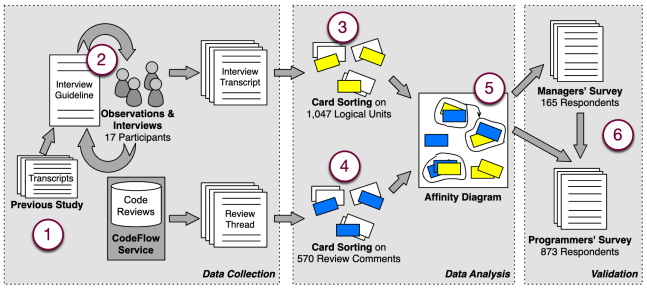


Fig. 2. The mixed approach research method applied.

Motivations for code review

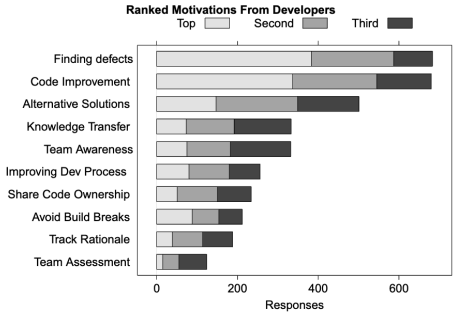
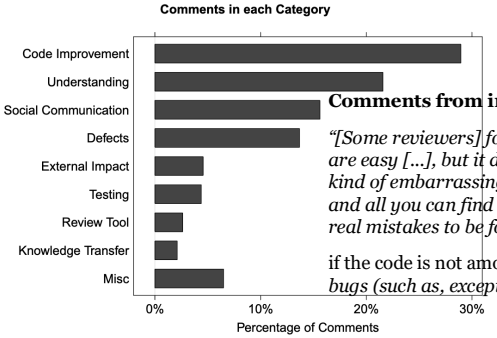


Fig. 3. Developers' motivations for code review.

Outcomes of code review



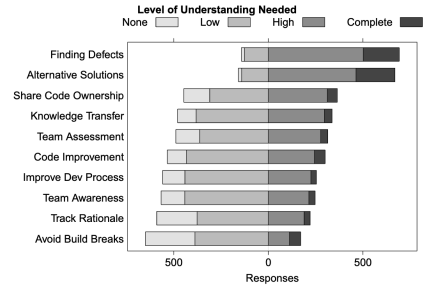
Comments from interviews:

"[Some reviewers] focus on formatting mistakes because they are easy [...], but it doesn't really help. [...] In some ways it's kind of embarrassing if someone asks you to do a code review and all you can find are formatting mistakes when there are real mistakes to be found"

if the code is not among their codebase, they look at *"obvious bugs (such as, exception handling)."*

Fig. 4. Proportion of comments by card sort category.

Why there are gaps between expectations and outcomes?



- Code Review is understanding
- Many interviewees eventually acknowledged that understanding is their main challenge when doing code reviews
- In the code review comments analyzed, the second most frequent category concerns understanding

Fig. 5. Developers responses in surveys of the amount of code understanding for code review outcomes.

Challenges of modern code review

- Understanding the code is hard
  - Most time is spent on trying to grasp context rather than finding defects.
  - Descriptions in review tools aren't always helpful.
- Lack of tool support for code comprehension
  - Most tools only highlight diffs but lack features for understanding rationale.
- Superficial reviews when unfamiliar with code
  - Reviewers unfamiliar with the code tend to focus on minor issues rather than deep flaws.
- Lack of synchronous communication
  - Review comments are often misunderstood due to lack of real-time discussions.

Positive points

- Comprehensive qualitative and quantitative study
  - 1047 logical units, 570 comments, 873 developers, and 165 managers
- Prove that finding defects is not the only purpose in modern code review
  - Knowledge transfer, share code ownership
- Show different perspectives about code review (manger vs. developer)
  - 17% developers put alternative solutions as first motivation
  - 4% managers only mentioned it
- Highlight the challenges of code review
  - Practical implications to practitioners and researchers

Negative points

- Findings may not generalize to other companies/tools
  - Smaller companies, open-source projects, GitHub, or startups?
- Potential observer bias in interviews and observations
  - Subconsciously influenced to follow best practices leading to inflated positive results
- Subjectivity might be introduced when items fall into multiple categories
  - Defect-finding vs. code improvement
- Missing practices happen outside CodeFlow
  - Motivations are not limited to this, but outcomes are
  - Many valuable code review about finding defect can happen outside CodeFlow
  - Would you submit code change for review when the reviewer is sitting right beside you?

Negative points

- Lack of verification of the outcomes
  - Comments can not reflect the actual implementations
  - People can say they will improve the code but do nothing as it is not very important
  - When people say they will fix the defect then it's very likely they will…
- Lack of long-term analysis of the outcomes of code review
  - Some outcomes may take time to exhibit (e.g., knowledge transfer)
  - Some review outcomes shift over time
  - Code improvement → developers learn and apply this principle → knowledge transfer

Overall rating

5/5

Future Work

- **Help write a good code review to facilitate code review process**
  - **What is a good code review? What impacts the code review process?**

Future Work

Code Review Quality: How Developers See It

Oleksii Kononenko
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
okononen@uwaterloo.ca

Olga Baysal
School of Computer Science
Carleton University
Ottawa, ON, Canada
olga.baysal@carleton.ca

Michael W. Godfrey
School of Computer Science
University of Waterloo
Waterloo, ON, Canada
migod@uwaterloo.ca

RQ1: While most of developers write patches as well as review them, a dedicated group of developers is responsible for reviewing code changes. The majority of reviewers conduct code review in Bugzilla despite having access to a custom built code review tool, and use various communication channels for discussing code modifications.

RQ2: Developers believe that factors such as the experience of developers, the choice of a reviewer, size of a patch, its quality and rationale affect the time needed for review; while bug severity, code quality and its rationale, presence and quality of tests, and developer personality impact review decisions.

RQ3: Developer perception of code review quality is shaped by their experience and defined as a function of clear and thorough feedback provided in a timely manner by a peer with a supreme knowledge of the code base, strong personal and inter-personal qualities.

Future Work

- Help write a good code review to facilitate code review process
  - What is a good code review? What impacts the quality of code review?
- **Long-term impact of code review on the quality of software**
  - **Does code review really lead to less post-release defects?**

Future Work

An empirical study of the impact of modern code review practices on software quality

Shane McIntosh · Yasutaka Kamei · Bram Adams · Ahmed E. Hassan

(RQ1) **Is there a relationship between code review coverage and post-release defects?**
We find that review coverage is negatively associated with the incidence of post-release defects in three of the four studied releases. However, it only provides a significant amount of explanatory power to two of the four studied releases, suggesting that review coverage alone does not guarantee a low incidence rate of post-release defects.

(RQ2) **Is there a relationship between code review participation and post-release defects?**
We find that the incidence of post-release defects is also associated with developer participation in code review. Review discussion metrics play a statistically significant role in the explanatory power of all of the studied systems.

(RQ3) **Is there a relationship between code reviewer expertise and post-release defects?**
Our models indicate that components with many changes that do not involve a subject matter expert in the authoring or reviewing process tend to be prone to post-release defects.

Future Work

- Help write a good code review to facilitate code review process
  - What is a good code review? What impacts the quality of code review?
- Long-term impact of code review on the quality of software
  - Does code review really lead to less post-release defects?
- **Help find proper reviewers for code review**
  - **Tools based on different metrics to recommend reviewers?**

Future Work

Automatically Recommending Peer Reviewers in Modern Code Review

Motahareh Bahrami Zanjani, *Student Member, IEEE*,
Huzefa Kagdi, *Member, IEEE*, and Christian Bird, *Member, IEEE*

Step 1: Extract source code under review: Given a code change under review for which reviewers are desired, it extracts each source code file.

Step 2: Formulate reviewer expertise: For each source code file in Step 1, it forms a reviewer expertise model based on how many, who performed, and when reviews were performed on it in the past. That is, we need to know the contribution of each past reviewer over the total number of reviews on it from the code-review history.

Step 3: Score and recommend reviewers: Finally, the cumulative contributions of the reviewer in Step 2 for all the source code files in Step 1 are scored to arrive at a ranked list of candidate reviewers. A user defined parameter m is used to recommend the top m candidates from this list. The choice of m can be guided by the organizational or project practices or historical information on the typical number of reviewers.

Future Work

- Help write a good code review to facilitate code review process
  - What is a good code review? What impacts the quality of code review?
- Long-term impact of code review on the quality of software
  - Does code review really lead to less post-release defects?
- Help find proper reviewers for code review
  - Tools based on different metrics to recommend reviewers?
- **Help understand code change or event automate code review**
  - **AI-based tools to generate summary and elaborate more on code comment?**

Future Work

Fine-Tuning Large Language Models to Improve Accuracy and Comprehensibility of Automated Code Review

YONGDA YU and GUOPING RONG, Nanjing University, Nanjing, China
HAIFENG SHEN, Southern Cross University, Gold Coast, Australia
HE ZHANG and DONG SHAO, Nanjing University, Nanjing, China
MIN WANG, ZHAO WEI, YONG XU, and JUHONG WANG, Tencent Technology (Beijing) Co. Ltd, Beijing, China

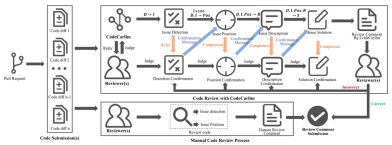


Fig. 5. The code review process with and without *Carllm*.

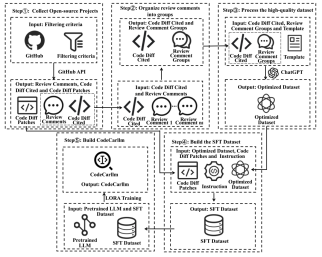


Fig. 1. The process of constructing *Carllm*.

Discussion points

- How do different organizational cultures affect code review practice?
  - Quality focused and speed focused
- How to keep a balance between qualities and rapid iteration?
  - DevOps-heavy organizations, startups
- How to give constructive reviews while avoiding discouraging the authors?
  - Reviewers may get down after reviewing too many bad code changes
- How do you think about code review automation?
  - Totally automated, hybrid or no automation
  - Advantages and disadvantages
  - How to improve?