Week 8: Paper 2: Expectations, Outcomes, and Challenges of Modern Code Review

Haonan Zhang
h294zhang@uwaterloo.ca

1 Problem being solved

In this paper, the authors try to answer 3 research questions: 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?

2 New idea

The authors employ a mixed-method approach, combining qualitative (interviews, observations, and card sorting) and quantitative (survey-based validation) techniques. They began by using the results from a previous study about the usage of CodeFlows. Although most questions were not relevant to the present research, one question, "What do you hope to accomplish when you submit a code review?" provided an initial set of motivations (finding defects, maintaining awareness, improving code quality, assessing design). These served as a starting point (and informal interview guide) for the next step.

Then a series of one-to-one meetings (40–60 minutes) was conducted. During which the researchers observed participants performing live code reviews and then interviewed them. The observation is done with minimal interference, and when the developers start to exhibit similar patterns to those they observed before, they reduce the observation time to give more time for the interview and collect more useful data. The semi-structured interviews try to explore a wider range of review motivations and practices. The interviews were transcribed and broken into coherent units for further analysis.

Open card sorting was then used to group these units (ideas) into higher-level categories. Because there were no predefined categories, the team iterated on the grouping until consistent themes emerged. A similar card sort was performed on code review comments taken from the large CodeFlow dataset. By sampling 200 review threads and printing each comment on a card (570 total), the researchers could identify key themes in how teams discussed and gave feedback during code reviews.

The categories generated by the two rounds of card sorting (interviews and review comments) were then organized into an affinity diagram. This technique places all the categories onto a wall and clusters them into logical groups through discussion. It provides a high-level picture of the major themes that emerged.

To verify and generalize these findings, two surveys were conducted. The first targeted managers (165 responses) to capture high-level perspectives on review practices in their teams. The second was sent to 2,000 developers, returning 873 responses. This quantitative validation served to confirm or challenge the earlier themes and categories derived from the qualitative steps.

3 Positive points

1. Comprehensive qualitative and quantitative study.
2. Prove that finding defects is not the only purpose in modern code review.
3. Show different perspectives about code review (manger vs. developer).
4. Highlight the challenges of code review.

4 Negative points

1. Findings may not generalize to other companies/tools.
2. Potential observer bias in interviews and observations.
3. Subjectivity might be introduced when items fall into multiple categories.
4. Missing practices happen outside CodeFlow.
5. Lack of verification of the outcomes and long-term analysis of the outcomes of code review.

5 Future work

Help write a good code review to facilitate the code review process. Long-term impact of code review on the quality of software. Help find proper reviewers for code review. Help understand the code change or automate code review.

6 Rating

5/5.

7 Discussion points

1. How do different organizational cultures affect code review practice?
2. How to keep a balance between qualities and rapid iteration?
3. How to give constructive reviews while avoiding discouraging the authors?
4. How do you think about code review automation?

8 Class Discussion

How can we give constructive reviews while avoiding discouraging authors?

1. Discouragement comes more from the feeling that code review isn't helping, or it isn't being taken seriously.
2. Even worse if it leads to bad code being pushed and breaking things.
3. Actually good constructive feedback may not actually discourage at all.
4. Knowledge transfer is important especially for new hires to illustrate how the company does things and highlight nuances that may not be clear from a high level.
5. New hires need to be onboarded to understand company-based tools and how they're designed
6. Face to face meetings can pressure individuals to perform better or act more quickly

How do you feel about code review automation?

1. Code tracking tools have made leaps and bounds in terms of allowing for better communication of code review concepts.
2. Even as early as the 90s, code review was very low level.
3. However, it is still important that people communicate with each other about code directly.
4. Code review is expensive at companies with fewer employees.
5. Hybrid approaches are good for helping this.
6. he hardest part of code review is understanding what is going on in the code.
7. Any automated aspect should still be reviewed by the relevant party.