Expectations, Outcomes, And Challenges Of Modern Code Review

Presented By

Tresa Varghese Rose

Carleton UNIVERSITY

Code Review

- Code review is the manual inspection of source code by developers other than the author. Also known as peer review.
- Valuable tool for reducing the errors and increasing software quality.
- 2 types- **formal code review** (70s and 80s) and **light weight code review**

Formal Code Review

- line by line inspection of code – more time consuming.

Light weight code review

- less overhead: uses specialized tools.
- Equally effective and less time consuming.
- Some of the code review tools are
  - Crucible(Java), GitLab(Ruby on Rails)
  - ReviewBoard(Python), UpSource(Windows, Mac OSX, Linux)

In this paper we are discussing about **Modern Code Review**

- More Informal, tool based and practised regularly

Related Study

- Previous studies were made on distributed asynchronous code review and participants at different locations communicating about faults via the tool.
  - Evaluation of tools
  - Code review practices
  - Effects of factors like team size, number of sessions, cost, and type of review

- ICICLE (Intelligent Code Inspection in a C Language Environment,) is one of the tool developed for reducing time consumed and to allow asynchronous work on code review.
- An exploratory investigation study is done in Microsoft where different teams working on diverse products.

Research Questions

- What are the motivations and expectations for modern code review? Do they change from managers to developers and testers?
- What are the actual outcomes of modern code review? Do they match the expectations?
- What are the main challenges experienced when performing modern code reviews relative to the expectations and outcomes?

Research Settings

MODE Exploratory investigation based on Interviews, manual classification of comments and survey

SUBJECTS Included developers, testers and managers in different domains like SQL server to mobile phone apps and smart phone apps.

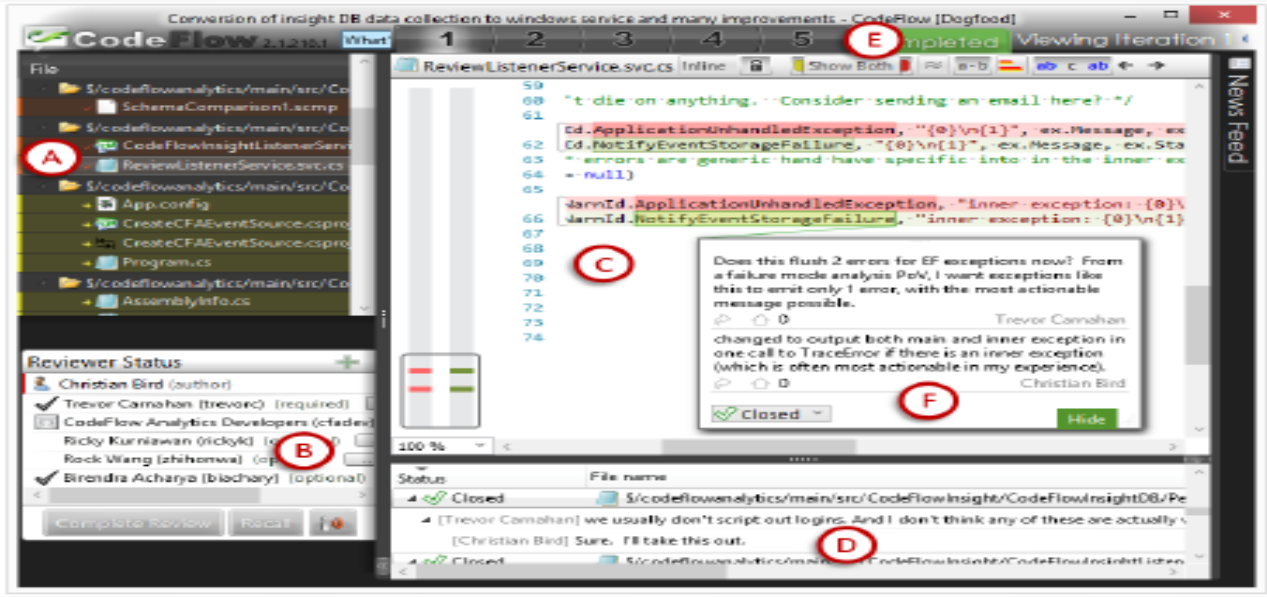
CODE REVIEW TOOL Focused on developers using the code review tool "*CodeFlow*"(since over 40,000 developers were using this tool since 2 years)

Working of CodeFlow

- CodeFlow is a tool that helps user to annotate code in viewer and participants can interact through chat mode.

Developers	Reviewers
Create a package with the changed files.	Open a CodeFlow review
Select the reviewers.	Interact with it through a single desktop window
Write a message to describe the code review	
Submit to CodeFlow Service.	
CodeFlow notifies reviewers about the incoming task via email.	

Table 1: how developers and reviewers use codeflow



- A → the list of files changed in the current submission, plus a 'description.txt' file
- B → the list of reviewers and their status .
- C → CodeFlow's main view .
- D → Both developers and reviewers can add comments inline.
- E → Current review status.
- F → Summary of all comments

Research Method

- An analysis of previous studies are done.
- 100 randomly selected candidates from different teams who have done 50 – 250 code reviews since codeFlow release have been contacted.
- Developers were asked to inform them before doing next review so that they can be observed.
- A semi-structured One on one interviews were conducted for 40 – 60 mins on the 17 respondents which include jr and sr developers, testers and s/w architect.
- The interview guideline of which the context was refined after each interview.

Motivations and Observations

- Finding defects
- Code improvement
- Alternative solutions
- Knowledge transfer
- Team Awareness and Transparency
- Share code Ownership

Observations	First motivation	Second Motivation	Third Motivation
Finding defects	383(44%)	204 (23%)	96 (11%)
Code Improvement	337 (39%)	208 (24%)	135 (15%)
Alternative Solutions	147 (17%)	202 (23%)	152 (17%)
Knowledge Transfer	73 (8%)	119 (14%)	141 (16%)
Team Awareness	75 (9%)	108 (12%)	149 (17%)
Share code Ownership	51 (6%)	100 (11%)	91 (10%)

Table 2: Observations based on developers

Reasons	Percentage
Finding defects	44%
Code Improvement	31%
Alternative Solutions	2%

Table 3: Observations based on managers

- Knowledge Transfer is Mentioned code review as an education means and not as first reason.
- Team Awareness & Transparency: Managers often mentioned the concept of team awareness as a motivation for code review.
- Team must be kept aware of the directions taken by the code, and nobody should be allowed to “secretly” make changes that might break the code or alter functionalities.
- But academic research seems to have given little attention to it.

- Manually inspected and classified the contents from the 570 comments in discussions within code review
- Observations were recorded, categorized and analysed.
- The concept emerged are validated using 2 surveys on 165 managers and 873 programmers.
- First survey had six optional questions and the answers are analyzed before sending second survey.
- The 18 questioned second survey is sent to randomly selected 2000 developers who has done atleast 1 code review per week during a time period of 6 months.
- The response rates were high(28% and 44%).

Outcomes Vs Motivation

- Outcomes were analyzed with the study on 570 comments.

Outcome(# of comments)	Comments Category
Code Improvements(165)	For using better code
	Removing not necessary or unused code
	Improving code readability
Defects Finding(78)	Wrong Expression
	High Level Issues
	On Security
	Wrong Exception Handling
Knowledge Transfer(12)	For learning how to tackle some issues.

Why expectations do not meet reality

- Subject of study(570 comments are not enough)
- Most comments on defects were low level issues.- as per interviewees. While programmers and managers expect conceptual or design level errors.
- As a result priority of fixing defect is less than code improvements.
- Some complained that quality of review is low as reviewers are looking for easy errors.
- if the code is not among their codebase, they look at "obvious bugs (such as, exception handling).
- Finally, managers mentioned "catching early obvious bugs" or "finding obvious inefficiencies or errors" as reasons for doing code review

Challenges

- **Understanding** the code change.
  - Descriptions on code change are not sufficient.
  - Most of the time is consumed in understanding.
- Clarifying the **questions** and doubts.
- Scheduling and **time** issues.
  - Easier if they are working on their own code.
  - More time needed to review others code.

- **Measures taken** to deal with challenges
  - **Reading** the descriptions of code change, try to run the changed code.
  - Sending **mails** for high level details
  - Personal **meetings** for clarifications from author.

Recommendations

For Practitioners :

- **Quality Assurance** : there is a mismatch between expectations and outcomes. Not able to identify micro level or deep defects.

- **Understanding** : if reviewers understand the code well, reviewing will be much faster. Team should increase the breadth of understanding of developers . change authors should include code owners and other with more understanding. If author provides context and direction, reviewers can work faster and better.

- **Communication**: should provide mechanisms for in-person or at least synchronous communications.

For Researchers:

- Automate code review tasks: Most comments were related to code improvement or micro level concerns. Automation frees reviewers to look for deeper defects.
- Program comprehension in Practice : challenges that developers face when reviewing has a direct relationship to the quality of review comments.
- Socio technical factors : studies can be done on how awareness and learning increase as a part of code review.

Limitations

- Study was limited to one company - wont contribute to academic community or scientific development.
- To determine how often code review improves team awareness and transfers knowledge is not easy.
- Cannot compare frequency of meetings with other outcomes while we just know they mostly for understanding the code change.

Discussion

- What more challenges are faced during a code review?
- How relevant is this study in the software industry ?
- What are the limitations of the paper ?

Conclusion

- Contributions of the paper:
  - Characterizing the motivations of developers and managers for code review and compare with actual outcomes.
  - Relating the outcomes to understanding needs and discuss how developers achieve such needs.
- Actual outcomes of code review do not always match with the motivation of finding errors.
- Most of the issues detected are low-level issues.
- Code review provide benefits such as improving code style, increased learning, knowledge transfer, team awareness and improved solutions to the forums.


Thank
you