Ten Years Of Asking Questions On Code Review

Alberto Bacchelli
Associate Professor
Department of Informatics
University of Zurich
ZEST: Zurich Empirical Software engineering Team

**Research**
- Software quality
  - Peer code review
  - Software testing
  - Software security
- Fundamentals of Data Science for Software Engineering
  - Predictive Analytics
  - Data-driven Tools

**Education**
- Software Design & Construction
- Software Testing
- Software Analytics
  (aka Data Science for S.E.)

http://zest.ifi.uzh.ch
zest@ifi.uzh.ch
ZEST: Zurich Empirical Software engineering Team

Research

- Software quality
  - Peer code review
  - Software testing
  - Software security
- Fundamentals of Data Science for Software Engineering
  - Predictive Analytics
  - Data-driven Tools

Google

ABB

Software Improvement Group

mozilla

Microsoft

AdNovum

HUawei
Empirical software engineering involves the scientific use of quantitative and qualitative data to understand and improve software product, software development process and software management.

Empirical software engineering starts with a good question:
• Does pair programming work?
• Is static typing really good?
• What are the advantages of properly following continuous integration?
• How does using GitHub influence open-source projects?

Empirical software engineering leads to actionable results:
• The creation of new tools
• The improvement of existing tools
• The improvement of existing development and engineering processes
• More questions :)

Empirical Software Engineering?
Software Engineering as a Socio-Technical Space

human / social aspects

socio-technical aspects

technical aspects
Software Engineering as a Socio-Technical Space

Productivity Paradox — Margaret-Anne Storey ICSE 2019
Joint Space — Code Review

code review

human / social

techical

socio-technical
Joint Space — Code Review

Software system timeline

Version i

Author

Code review

Reviewers

Version i+1
Why research on code review?
Let's look for “code review best practices” on Google...

11 proven practices for more effective, efficient peer code review - IBM
https://www.ibm.com\Learn\Rational
25 Jan 2011 - Aim for an inspection rate of fewer than 300–500 LOC per hour. Establish quantifiable goals for code review, and capture metrics so you can improve your processes. Verify that the defects are actually fixed. Foster a good code review culture in which finding defects is viewed positively.
Beware of the Big Brother ...

Code Review Best Practices - Kevin London's blog
kevinlondon.com/2015/05/05/code-review-best-practices.html
5 May 2015 - I think it's a good idea to crystallize some of the things I look for when I'm doing code reviews and talk about the best way I've found to approach ...

Best practices for effective code reviews - WillowTree Apps
https://willowtreeapps.com/ideas/best-practices-for-effective-code-reviews
27 Oct 2016 - Today, I'd like to share our process and some best practices we follow when conducting code reviews. The process. The code review process ...

https://nyu-cds.github.io/effective-code-reviews/02-best-practices/
What are some best practices for code reviews? Objectives. Learn about effective practices for code reviews. Learn what makes reviews work better and what ...

7 best practices for doing code reviews - The Asana Blog
https://blog.asana.com/2016/12/7-ways-to-uplevel-your-code-review-skills/
20 Dec 2016 - The Asana engineering team shares code review best practices that will help you become a better reviewer. Learn how Asana reviews code.

Code Review in Agile Teams - part II - Atlassian Blog
https://www.atlassian.com/blog/archives/code_review_in_agile_teams_part_ii
8 Mar 2010 - Ready to try adopting code review within your team or across your .... reveal a few best practices around code review that evolved at Atlassian.

Best Practices: Code Reviews - MSDN - Microsoft

...
Two examples

11 PROVEN PRACTICES FOR MORE EFFECTIVE, EFFICIENT CODE REVIEW
- Review fewer than 200–400 lines of code at a time
- Aim for an inspection rate of fewer than 300–500 LOC per hour
- Take enough time for a proper, slow review, but not more than 60–90 minutes
- Be sure that authors annotate source code before the review begins
- Establish quantifiable goals […] and capture metrics [to] improve your processes
- Use checklists, because they substantially improve results
- Verify that the defects are actually fixed
- Foster a good code review culture in which finding defects is viewed positively
- Beware of the Big Brother effect
- Review at least part of the code, even if you can't do all of it, [for] The Ego Effect
- Adopt lightweight, tool-assisted code reviews

7 WAYS TO IMPROVE YOUR CODE REVIEW SKILLS
- Prioritize the goals of code reviews with your team
- Run the app and try playing with the feature
- Visualize method call hierarchies
- Do code reviews as soon as you see the request
- Imagine how you would make this change before you read it
- Read the change in a realistic development environment
- Always give approval, unless you can prove that there is a bug
11 PROVEN PRACTICES FOR MORE EFFECTIVE, EFFICIENT CODE REVIEW
- Review fewer than 200–400 lines of code at a time
- Aim for an inspection rate of fewer than 300–500 LOC per hour
- Take enough time for a proper, slow review, but not more than 60–90 minutes
- Be sure that authors annotate source code before the review begins
- Establish quantifiable goals […] and capture metrics [to] improve your processes
- Use checklists, because they substantially improve results
- Verify that the defects are actually fixed
- Foster a good code review culture in which finding defects is viewed positively
- Beware of the Big Brother effect
- Review at least part of the code, even if you can’t do all of it, [for] The Ego Effect
- Adopt lightweight, tool-assisted code reviews

7 WAYS TO IMPROVE YOUR CODE REVIEW SKILLS
- Prioritize the goals of code reviews with your team
- Run the app and try playing with the feature
- Visualize method call hierarchies
- Do code reviews as soon as you see the request
- Imagine how you would make this change before you read it
- Read the change in a realistic development environment
- Always give approval, unless you can prove that there is a bug
Ten Years Of Asking Uncomfortable Questions On Code Review

Alberto Bacchelli
Associate Professor
Department of Informatics
University of Zurich
Asking Uncomfortable Questions On Code Review

But.. why are we doing code review at all?
Why are managers and developers employing code review?
Modern code review @ Microsoft
Modern code review @ Microsoft
Used across all Microsoft product teams by more than 70,000 developers, so far.
observations

18-20 interviews

survey to 165 managers
List of motivations for doing code review

- Alternative Solutions
- Avoid Build Breaks
- Code Improvement
- Team Assessment
- Share Code Ownership
- Team Awareness
- Knowledge Transfer
- Improve Dev. Process
- Track Rationale
- Finding Defects
observations

18-20 interviews

survey to 165 managers

survey to 873 developers
Why do Microsoft developers do code reviews?

- Finding defects
- Code improvements
- Alternative solutions
- Knowledge transfer
- Team awareness
- Improving dev process
- Share code ownership

1st reason
2nd reason
3rd reason
Why do Microsoft developers do code reviews?

“Finding defects is the main reason for doing code review.”

72 managers and 384 developers @ Microsoft

- Finding defects
- Knowledge transfer
- Team awareness
- Improving dev process
- Share code ownership

1st reason
2nd reason
3rd reason
What is the outcome of code review at Microsoft?
What is the outcome of code review at Microsoft?
Wouldn’t it be better to put this as a parameter of the SayGreeting method?

- Alberto Bacchelli

I wouldn’t. Greeting is already a field! If you do that, you’d want to make Times a parameter as well.

- Tom Zimmermann

Good point. I’ll leave it as is.

- Christian Bird
classification of 570 review comments

18-20 interviews

survey to 873 developers

survey to 165 managers

observations
Results of review comments’ analysis

- Code improvement: 30%
- Understanding: 20%
- Social communication: 15%
- Defects: 15%
- External impact: 10%
- Testing: 5%
- Review tool: 5%
- Knowledge transfer: 5%
- Misc: 0%
Results of review comments’ analysis

- code improvement: 30%
- understanding: 20%
- social communication: 15%
- defects: 10%
- external impact: 5%
- testing: 5%
- review tool: 5%
- knowledge transfer: 5%
- misc: 5%

% of comments
Results of review comments’ analysis

- Code improvement: 30%
- Understanding: 20%
- Social communication: 10%
- Defects: 10%
- External impact: 5%
- Testing: 5%
- Review tool: 5%
- Knowledge transfer: 5%
- Misc: 5%

% of comments
Results of review comments’ analysis

“what if they are all used?”

“is it possible that this statement never match?”

“should this end date be current date?”

“does it work if you put 0 here?”

“any doubt about the precedence here?”

“should be &&?”
Results of review comments’ analysis

- code improvement
- understanding
- social communication
- defects
- external impact
- testing
- review tool
- knowledge transfer
- misc

% of comments
If you want to use any Software Engineering process PROPERLY, you need to evaluate it and reflect on it. Even if you are Microsoft.
But.. why are we doing code review at all?

Our code review tools are great! Aren't they?
Modern code review tools: Microsoft CodeFlow
Modern code review tools: Gerrit

Change Id: l26cd0da83acc9df2162b6ec6174fc25712f68de2

Owner: jp abgrall
Project: kernel/common
Branch: android-3.4
Topic:
Uploaded: Feb 25, 2013 10:05 PM
Updated: Feb 25, 2013 10:12 PM
Status: Merged

Reviewer: Code-Review
- Verified
  - jp abgrall
  - Nick Kralevich

Commit Message
sock_diag: Fix out-of-bounds access to sock_diag_handlers[].

Userland can send a netlink message requesting SOCK_DIAG_BY_FAMILY
with a family greater or equal than AF_MAX -- the array size of
sock_diag_handlers[]. The current code does not test for this
condition therefore is vulnerable to an out-of-bound access opening
doors for a privilege escalation.

Signed-off-by: Mathias Krause <minipli <at> googlemail.com>

Old Version History: Base

Patch Set 1
Author: Mathias Krause <minipli <at> googlemail.com> Feb 23, 2013 12:13 PM
Committer: JP Abgrall <jp@google.com> Feb 25, 2013 9:57 PM
Parent(s):
  1a6c74e0d58f6ca028f36c3d794f8ecf8543bbf
  gpu: ion: Add support for sharing buffers with dma buf kernel handles
  checkout | checkout | pull | cherry-pick | patch
Sending a pull request #248

Open cameronmcefee wants to merge 1 commit into octocat:master from cameronmcefee:master

5 Conversation 1 1 Commits 1 1 Files Changed 1

cameronmcefee commented 2 years ago

I made some changes. Please review.

octocat commented 2 years ago

Awesome, thanks!

cameronmcefee commented 2 years ago

Why yes, of course.

cameronmcefee closed the pull request 2 years ago
Modern code review tools: Atlassian Crucible

Code review tools are all very similar and basically only help with the logistics of the review, not with the review task itself!
Modern code review tools: Atlassian Crucible

```
public static TrackedBranchesSearchCriteria withReview(String reviewPermaId) {
    return builder().reviewPermaId(reviewPermaId).build();
}
```

Piotr Święcicki
providing null/empty reviewPermaId results in criteria matching all reviews, I'd rather expect precondition failure in such circumstances

Cezary Zawadka
Null is ok as we search for all tracked branches regardless review - auto update feature
However non null perma id means we should return empty list if there is no review with the perma id - changed to be handled at ReviewPropertiesManager level

Cezary Zawadka
My mistake - for withReview() precondition makes sense

```
Modern code review tools: Alphabetical Ordering Of Files

Most developers start with the tool order when inspecting code under review.
Instead start reviewing the tests, to:
1. understand what the code really **does**,
2. have higher quality tests,
3. and find **more bugs**!
Next generation code review tools: Risk-guided Code Review

- Risk detector
- In-line warnings
- Re-ordered files
- Versioning system
- Code review data
- Issue tracking system
Next generation code review tools: Code Change Visualization
Next generation code review tools: Code Change Visualization
Current code review tools are only scratching the surface of what can be done to support reviewers.
But.. why are we doing code review at all?

Our code review tools are great! Aren't they?

Are we really in this together?
Who should review a change?

Piotr Świecicki
providing null/empty reviewPermaId results in criteria matching all reviews, I'd rather expect precondition failure in such circumstances

Cezary Zawadka
Null is ok as we search for all tracked branches regardless review - auto update feature
However non null perma id means we should return empty list if there is no review with the perma id - changed to be handled at ReviewPropertiesManager level

Piotr Świecicki
fine for reviewPermaId property, but if client calls withReview method to build criteria, I'd assume he wanted to filter by particular perm id and was not expecting to pass null.

Cezary Zawadka
My mistake - for withReview() precondition makes sense
Who should review the change? Reviewer recommender

- mostly developers do not need it
- always the same people are recommended
- workload is not considered
- diversity is not considered
Who should review the change? Reviewer recommender

Reviewer recommender

most appropriate reviewer #1

most appropriate reviewer #2

most appropriate reviewer #3

Tools and teams should be more mindful and supportive when it comes to reviewing code.
But.. why are we doing code review at all?

Our code review tools are great! Aren't they?

Are we really in this together?

Can we find security problems in code review?
Can we find security problems in code review?

**Code to Review**

We are now going to show you the code changes to review.

**Instructions**

- Take the review task very seriously (this is critical for the scientific validity of this experiment).
- The old version of the code is on the left, the new version is on the right.
- Assume that the code compiles and that all tests pass.
- **Review comments**
  - To add a review comment, click on the corresponding line number.
  - To modify/delete a review comment, click on the corresponding line number again and modify/delete the comment's text...
Can we find security problems in code review?

CWE-89: SQL Injection: Here there is a risk of SQL injection when, for example, an employeeID "' or '1'='1" is used. There are 2 conditions in the query. (1) employeeID = ": It will be evaluated to false as there is no empty employees in the table. (2) '1'='1': It will be evaluated to true as this is static string comparison. Now combining all 2 conditions i.e. false or true => Final result will be true.

```java
/**
 * Get the level for an employee, given their employee ID
 * @param employeeID
 * @return the current level of the specified employee
 * @throws SQLException in case of persistence-related issues (e.g., employee not found)
 */

protected int getEmployeeLevel(String employeeID) throws SQLException {
    String query = "SELECT * FROM tblemployees WHERE employeeID='" + employeeID + "'";

    int employeeLevel = rs.getInt("employeeLevel");
    rs.close();
    return employeeLevel;
}
```
Can we find security problems in code review?

- Not found
- Found

SQLI
- 35% (28)
- 65% (52)
Can we find security problems in code review?

- Not found
- Found in the first review
- Found after the warning

SQLI
(80)

- 19% (15) found in the first review
- 16% (13) found after the warning
- 65% (52) not found
Can we find security problems in code review?

- Not found
- Found in the first review
- Found after the warning

**SQLI**
- 19% (15)
- 16% (13)
- 65% (52)

**IVQI**
- 21% (14)
- 11% (7)
- 68% (45)
Can we find security problems in code review?

- Not found
- Found in the first review
- Found after the warning

Most factors related to low knowledge and practice contribute to missing vulnerabilities during code review, also after prompting.
Other Uncomfortable Questions On Code Review We Are Investigating

- How should a change be **split for review**?
- Can we **measure the effect** of code review?
- What **cognitive biases** are influencing our review?
- How do **top reviewers** go about reviewing code?

...
Empirical Software Engineering?

**Empirical software engineering** involves the scientific use of quantitative and qualitative data to understand and improve software product, software development process and software management.

Empirical software engineering **starts with a good question:**
- Does pair programming work?
- Is static typing really good?
- What are the advantages of properly following continuous integration?
- How does using GitHub influence open-source projects?

Empirical software engineering **leads to actionable results:**
- The creation of new tools
- The improvement of existing tools
- The improvement of existing development and engineering processes
- More questions 😊

**Joint Space — Code Review**

**Asking Uncomfortable Questions On Code Review**

- But.. why are we doing code review at all?
- Our code review tools are great! Aren't they?
- Are we really in this together?
- Can we find security problems in code review?

**Next generation code review tools: Code Change Visualization**

**Who should review the change? Reviewer recommender**

- Reviewer recommender
  - Most appropriate reviewer #1
  - Most appropriate reviewer #2
  - Most appropriate reviewer #3

**Can we find security problems in code review?**

- Not found
- Found in the first review
- Found after the warning

Tools and teams should be more mindful and supportive when it comes to reviewing code.

**Code review at Microsoft in 2012: Expectations vs. Reality**

*If you want to use any Software Engineering process PROPERLY, you need to evaluate it and reflect on it. Even if you are Microsoft.*

---

**Software Engineering as a Socio-Technical Space**

*Productivity Paradox — Margaret-Anne Storey ICSE 2019*
Thank you!
Alberto

http://zest.ifi.uzh.ch
zest@ifi.uzh.ch

Software Engineering as a Socio-Technical Space

Research
- Software quality
- Peer code review
- Software testing
- Software security
- Fundamentals of Data Science for Software Engineering
- Predictive Analytics
- Data-driven Tools

Education
- Software Design & Construction
- Software Testing
- Software Analytics
  (aka Data Science for S.E.)

Joint Space — Code Review

Next generation code review tools: Code Change Visualization

 Asking Uncomfortable Questions On Code Review

But, why are we doing code review at all?

Our code review tools are great! Aren’t they?

Are we really in this together?

Can we find security problems in code review?

Who should review the change? Reviewer recommender

reviewer recommender

most appropriate reviewer #1
most appropriate reviewer #2
most appropriate reviewer #3

Current code review tools are only scratching the surface of what can be done to support reviewers.

Tools and teams should be more mindful and supportive when it comes to reviewing code.

Code review at Microsoft in 2012: Expectations vs. Reality

expectation reality

hot chocolate

If you want to use any Software Engineering process PROPERLY, you need to evaluate it and reflect on it. Even if you are Microsoft.

Most factors related to low knowledge and practice contribute to missing vulnerabilities during code review, also after prompting.