DR. MICHAELA GREILER

CODE REVIEWS

WWW.MICHAELAGREILER.COM

CONTENT

03
ABOUT DR. GREILER

04
CODE REVIEW WORKSHOP

05
CODE REVIEW STUDY

09
CODE REVIEW BEST PRACTICES

10
CODE REVIEWS AT MICROSOFT

11
CODE REVIEWS AT GOOGLE

CONTENT

12
MOTIVATIONS FOR CODE REVIEWS

13
CODE REVIEW PITFALLS

14
CODE REVIEW TOOL NEEDS

15
CODE REVIEW CHECKLIST - PROCESS

16
CODE REVIEW CHECKLIST - CODE

Hi, I'm Dr. Michaela Greiler

I help companies and teams to implement world-class software engineering practices to ensure they build high-quality software in an efficient and effective way.

I worked for corporations such as Microsoft and Flextronics. But, I also help smaller businesses and start-ups to ensure a productive, satisfying and efficient software engineering process.

The most well-known engineering teams I worked with have been Office, Visual Studio and Windows. I helped them increase velocity and quality during software development.

Dr. Michaela Greiler

DO YOU WANT TO BOOST YOUR SOFTWARE ENGINEERING PRACTICE?

Book me as consultant, keynote speaker or trainer.

Click for more information

MAKE CODE REVIEWS YOUR SUPERPOWER

This highly interactive full-day workshop is designed to give you superpowers through your boosted code review practice.

After this workshop you know

- proven code review best practices from leading software companies such as Microsoft, Google, or Facebook,
- how to boost productivity and decrease review turn-around time,
- best ways to use code reviews for mentoring and knowledge sharing,
- how to give and get valuable code review feedback,
- friction and bottlenecks of your and your team's practices,
- and solution approaches to overcome those problems.

The workshop is led by Dr. Michaela Greiler, who is an expert in the field of code reviews. She works at Microsoft, where she conducted multiple large-scale studies on code reviews, and published her findings in highly reputable scientific publications. She leads product teams such as Office, Windows and Visual Studio to improved and optimized engineering processes.

This training is for you if you

- already successfully practice code reviews and want to make them your superpower.
- experience code review pitfalls such as slow code review turn-around times, low feedback quality, unclear review guidelines, and want improvements.
- haven't started with code reviews, but want learn how to best make use of this practice.

Learn more here!

CODE REVIEW STUDY

At Microsoft, we conducted a study to understand developers' needs and best practices during code reviewing.

For this large-scale study, surveyed more than 900 developers. This survey was a follow-up from interviews and observations we did with 18 developers from 4 different projects.

Especially this combination of quantitativ results of hundreds of developers with the qualitative and in-depth findings from sitting with teams for over one week to observe and understand their code review activities in detail, is a very powerful one.

This report highlights some of the outcomes of this in-depth study.

The observations, interviews and survey helped us to get a very clear understanding of the challenges software engineers face when conducting code reviews.

We also distilled several best practices that can help teams to be more productive and get the most value from code reviewing.

To learn more about this study:
Code Review Study

SURVEY FOCUS

In our survey at Microsoft, we asked over 900 engineers about their demographics, their team policies and of course which review tools they use.

We also asked them about their motivation for code reviews, and the challenges they face.

Finally, we asked them concrete questions on their reviewing steps and also inquired about a number of personal issues based on code review activities.

Here are some results:

SURVEY DEMOGRAPHICS

75%

of respondents are software engineers or senior software engineers

25%

are either managers in technical roles or product managers

HOW OFTEN DO YOU REVIEW CODE?

39%

indicate to review changes of others at least once a day.

21%

of the respondents review multiple changes per day.

36%

review code at a couple of times per week

13%

either review once per week or do not participate in code reviews in the last week

MOTIVATIONS

The most important reasons for developers to do code reviews are:

- to improve the code,
- to find defects,
- to transfer knowledge about the code,
- and to find alternative solutions.

CODE REVIEW MOTIVATION

Many developers expressed their motivations to do and the benefits they get from code reviews in the free text field.

A large number of the developers mentioned that teaching and mentoring junior developers is one of the key reasons to engage in code reviewing.

Another often expressed motivation is self-improvement and learning.

Finally, many engineers mentioned that code reviews allow the team to develop a coding culture, to develop best practices and to ensure a high quality of their code.

Code Review Best Practices

Code Author

- Read through the changes carefully before submitting the code review
- Aim for small, incremental changes
- Cluster related changes
- Describe the purpose and motivation of the change
- Run tests before submitting a code review
- Automate what can be automated
- Skip unnecessary reviews
- Do not select too many reviewers
- Add experienced reviewers if you need them
- Add junior developers to let them learn
- Notify people that benefit from this review
- Don't notify too many people
- Give reviewers a heads-up before the review
- Be open to suggested changes
- Show respect and gratitude to the reviewers

Code Reviewer

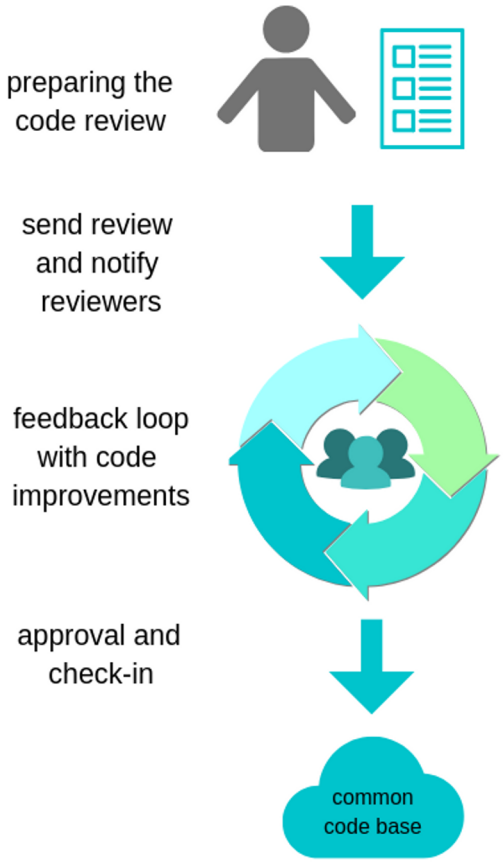
- Give respectful and constructive feedback
- Go and talk in person if necessary
- Ensure traceability for decisions
- Always explain why you rejected a change
- Integrate code review into your daily routine
- Reduce task switching as it kills productivity
- Give feedback in a timely manner
- Review frequently not in a big bang fashion
- Focus on core issues, less nit-picking
- Use a review checklists

Read more about **Code Review Best Practices** here.

CODE REVIEWS AT MICROSOFT

Code reviews at Microsoft are an integral part of the development process

One of the important facts when it comes to code reviews at Microsoft is that it is a highly adopted engineering practice. Thousands of engineers perceive it as a great best practice. And most high-performing teams spend a lot of time doing code reviews.



But how does a developer typically do code reviews?

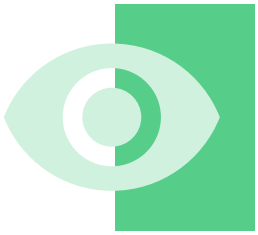
Code reviews can be performed in many ways. Sometimes, it is as informal as one developer walking over to another developer's desk to look at some code together. Other times, teams review code together in groups. But the most likely scenario you will encounter for code reviews at Microsoft is that code reviews are done with the help of tools.

A typical code review process

A tool-based code review starts when the developer finished a piece of code. As a first step, the developer reads again carefully through the code and then selects reviewers. The reviewers give comments on the code and the developer works on those comments and improves the code accordingly. Once everybody is satisfied the code is checked into the code base.

Read more about Code Reviews at Microsoft here.

CODE REVIEWS AT GOOGLE

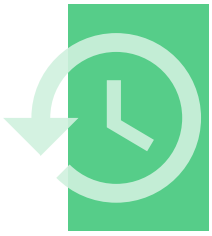


EVERY CODE CHANGE IS REVIEWED

75% of the code reviews are approved by only one reviewer.

COMPANY-WIDE APROVAL CRITERIA

Approver needs ownership rights and readability certificate



4 HOURS TO REVIEW COMPLETION

Small reviews are completed within one hour. Large reviews within five hours.

SMALL AND FREQUENT REVIEWS

90% of all code changes comprise less than 10 files and 24 lines of code.

Read more about **Google's Code Review Practices** here.

5 REASONS GOOGLERS REVIEW CODE

Education

Mentoring, learning, knowledge disemination.

Accident prevention

Find bugs and defects, ensure high quality code

Gatekeeping

Prevent arbitrary code to be committed, security

Tracing & tracking

Understanding evolution and why and how code changed

Readable Code

Maintaining norms, consistent style and design, and having adequate tests

"*In general, reviewers should favor approving a CL once it is in a state where it definitely improves the overall code health of the system being worked on, even if the CL isn't perfect.*"

- Definition of Done for Reviews at Google

CODE REVIEW PITFALLS

Code reviewing isn't always a smooth process

Knowing which code review pitfalls and problems arise, can help you to ensure a productive and effective code review experience.

During the code review process there are quite a few pitfalls that can reduce the positive experience with code reviews for the whole team. If not done correctly, code reviewing can also take its tolls on the whole team's productivity. So, let's have a look at the difficulties and pitfalls of code reviews.

The two main types of code review pitfalls are about the time spent on code reviewing, and the value code reviewing provides.

Waiting for code review feedback is a pain

One of the main pitfalls code authors face is to receive feedback in a timely manner. Waiting for the comments to come in and not being able to work on the code in the meanwhile can be a huge problem. Even though developers can pick up other tasks to work on, if the code review takes too long, it impacts the developer's productivity and also the developer's satisfaction.

Not getting valuable feedback decreases the developers' benefit from and motivation for code reviews

There are several reasons why reviewers can't give insightful feedback. Large code reviews, and the developer not having the right expertise are common ones. Another reason is if the reviewer hasn't enough time to look through the change.

Read all Code Review Pitfalls here.

WHAT DEVELOPERS NEED WHEN CODE REVIEWING?

TOOL NEEDS AND OPPORTUNITIES

Often code reviews are done with the help of tools. In the early days, code reviews where done via emails. But since then, many code review tools have been developed that support collaboration and asynchronous code reviews.

In a large scale study with more than 900 engineers at Microsoft, we investigated which needs developer have with regards to code review tools.

"When selecting or designing tools, it is important to carefully address trade-off, as tools shape the practices."

WHAT DEVELOPERS WANT:

- ease of use and tool performance
- integration with other services and tools
- ability to edit and execute code during a review
- support during describing a change to the reviewers
- support of the review discussion for note taking and documenting
- support in notification and tracking of the code review life cycle
- support for informal communications

Code Review Checklist

Process

Code Author

Is the code okay?

- Have you looked through the code change carefully (using a diffing tool)?
- Is the code change small, incremental and coherent?
- Have you written a description about the code change?
- Have you tested and analyzed the the code change?
- Does this code need a review?

Who should review?

- How many reviewers are needed?
- Do I need experienced developers as reviewers?
- Do junior developers need to learn?
- Who should be notified about the code change?
- Did I give people a heads-up that a change is coming?

How do I handle feedback?

- Am I emotionally prepared to be asked to change the code?
- Did I tell reviewers how I have addressed and fixed the issues?
- Did I document the changes I made for traceability?
- Have I showed gratitude to the reviewers?

Code Reviewer

How do I structure my work day?

- When is a good time to do code reviews?
- Have I set dedicated time aside to do code reviews?
- How can I provide feedback in a timely-manner?
- Do I have a code review checklist?

How do I give valuable feedback?

- What are the core issues?
- Did I start nit-picking?
- Is my feedback helpful and constructive?
- Am I using the right channel (tool, email, video call or face-to-face) to give that feedback?
- Did I communicate in a respectful way?
- Can this code change be accepted?
- If I reject the change, did I explain why?
- Did I explain what the code author has to do to get the code accepted?

Read more about **Code Review Best Practices** here.

Code Review Checklist

Code (1/3)

Implementation

- Does this code change do what it is supposed to do?
- Can this solution be simplified?
- Does this change add unwanted compile-time or run-time dependencies?
- Was a framework, API, library, service used that should not be used?
- Was a framework, API, library, service not used that could improve the solution?
- Is the code at the right abstraction level?
- Is the code modular enough?
- Would you have solved the problem in a different way that is substantially better in terms of the code's maintainability, readability, performance, security?
- Does similar functionality already exist in the codebase? If so, why isn't this functionality reused?
- Are there any best practices, design patterns or language-specific patterns that could substantially improve this code?
- Does this code follow Object-Oriented Analysis and Design Principles, like the Single Responsibility Principle, Open-close Principle, Liskov Substitution Principle, Interface Segregation, Dependency Injection?

Dependencies

- If this change requires updates outside of the code, like updating the documentation, configuration, readme files, was this done?
- Might this change have any ramifications for other parts of the system, backward compatibility?

Security and Data Privacy

- Does this code open the software for security vulnerabilities?
- Are authorization and authentication handled in the right way?
- Is sensitive data like user data, credit card information securely handled and stored?
- Is the right encryption used?
- Does this code change reveal some secret information like keys, passwords, or usernames?
- If code deals with user input, does it address security vulnerabilities such as cross-site scripting, SQL injection, does it do input sanitization and validation?Is data retrieved from external APIs or libraries checked accordingly?

Code Review Checklist

Code (2/3)

Logic Errors and Bugs

- Can you think of any use case in which the code does not behave as intended?
- Can you think of any inputs or external events that could break the code?

Error Handling and Logging

- Is error handling done the correct way?
- Should any logging or debugging information be added or removed?
- Are error messages user-friendly?
- Are there enough log events and are they written in a way that allows for easy debugging?

Testing and Testability

- Is the code testable?
- Does it have enough automated tests (unit/integration/system tests)?
- Do the existing tests reasonably cover the code change?
- Are there some test cases, input or edge cases that should be tested in addition?

Readability

- Was the code easy to understand?
- Which parts were confusing to you and why?
- Can the readability of the code be improved by smaller methods?
- Can the readability of the code be improved by different function/method or variable names?
- Is the code located in the right file/folder/package?
- Do you think certain methods should be restructured to have a more intuitive control flow?
- Is the data flow understandable?
- Are there redundant comments?
- Could some comments convey the message better?
- Would more comments make the code more understandable?
- Could some comments be removed by making the code itself more readable?
- Is there any commented out code?

Code Review Checklist

Code (3/3)

Usability and Accessibility

- Is the proposed solution well designed from a usability perspective?
- Is the API well documented?
- Is the proposed solution (UI) accessible?
- Is the API/UI intuitive to use?

Performance

- Do you think this code change will impact system performance in a negative way?
- Do you see any potential to improve the performance of the code?

Experts Opinion

- Do you think a specific expert, like a security expert or a usability expert, should look over the code before it can be committed?
- Will this code change impact different teams? Should they have a say on the change as well?

Find more at michaelagreiler.com

- Code Review Best Practices
- Code Review Pitfalls
- Code Reviews at Microsoft
- Code Reviews at Google

Read more about **Code Review Checklists** here.

Get 10% off my workshop on code review

Book a free appointment

Or send an inquiry to mckayla@michaelagreiler.com

YOUR TECH PODCAST

Software Engineering **UNLOCKED**

WWW.SE-UNLOCKED.COM

FIND IT ON ITUNES, SPOTIFY AND MANY MORE