Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Software Reviews

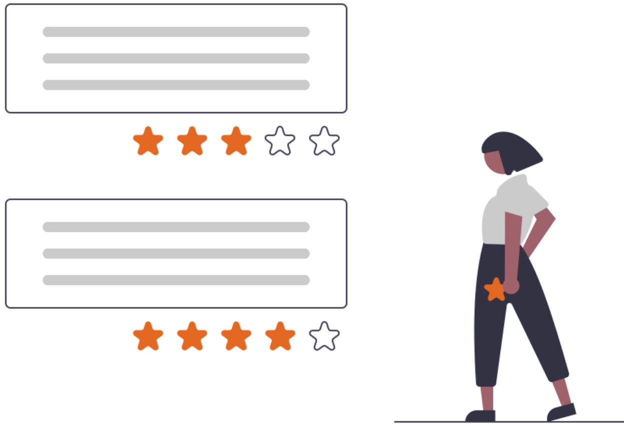
Scalable Software Engineering
WS 2022/23

Enterprise Platform and Integration Concepts

Image courtesy *@EthicsInBricks* in Twitter: https://twitter.com/EthicsInBricks/status/1430556314556669956 (with permission)

Software Reviews

> “a **software** product is [**examined** by] project personnel, managers, users, customers, user representatives, or other interested parties **for comment or approval**”
>
> —IEEE1028



Principles

- Generate **comments** on software
- Several sets of eyes check
- Emphasis on **people over tools**
- **Lower cost** of fixing defects in review than in the field

Software Reviews

Motivations

- **Improve code quality**
  (e.g. maintainability, readability, uniformity)
- Discuss alternative solutions, **generate ideas** for the future
- **Knowledge transfer** regarding codebase
- Increase sense of **Collective Code Ownership**
- Find **defects**
- Check **compliance** (e.g. legal)

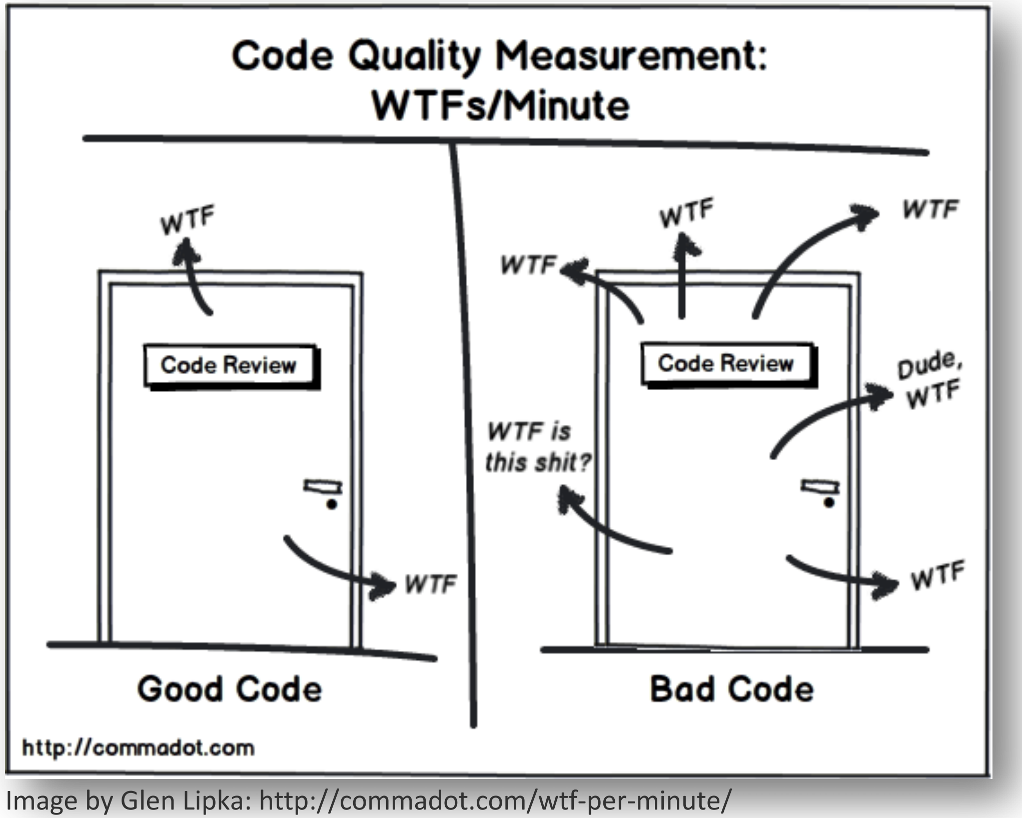
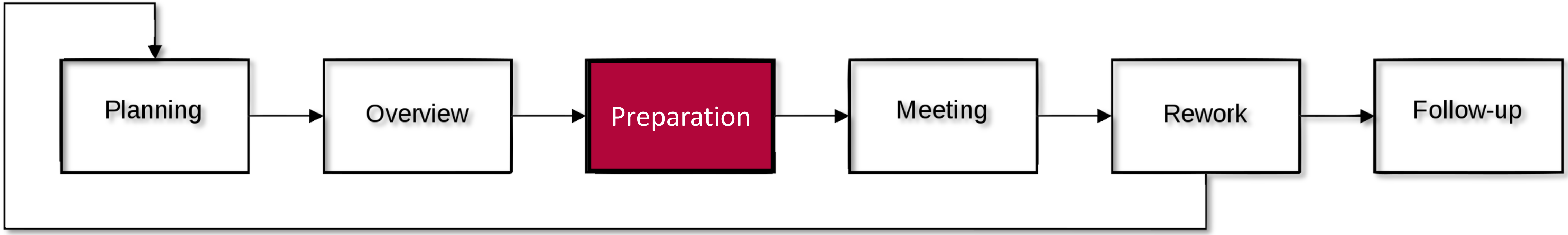


Image by Glen Lipka: http://commadot.com/wtf-per-minute/

Types of Reviews [IEEE1028-2008]

One type of review: Inspection

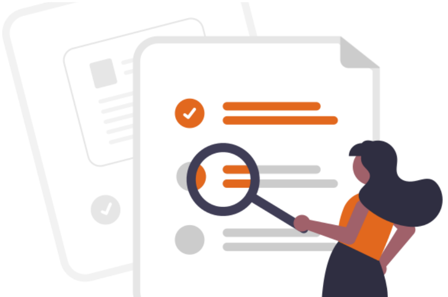
- Identify software product anomalies
- Since the 1970’s, aka “Fagan Inspection”
- **Formal process**, can involve hard copies of the code and documents
- Review team checks artifacts independently before, consolidation meeting with developers



Fagan inspection basic model: https://en.wikipedia.org/wiki/Fagan_inspection#/media/File:Fagan_Inspection_Simple_flow.svg (CC0)

Focus in Reviews

Reviewed first	Reviewed later
Implementations of complex algorithms	Code in well-understood problem domains
Code where faults or exceptions lead to system failure	Code which won’t break the functionality if faults occur
Parts using new technologies/libraries	Parts similar to those previously reviewed
Parts written by new or inexperienced team members	Reused and already reviewed parts
Code that features high code churn	Code with few changes

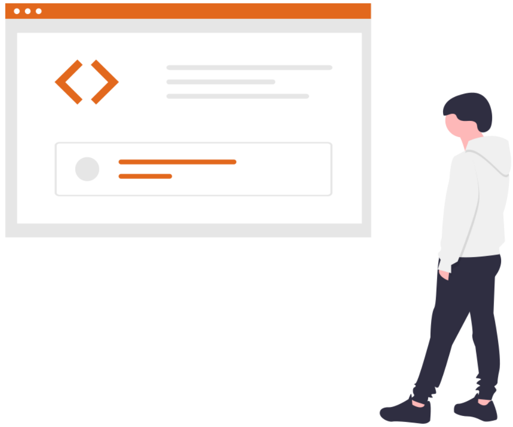


Change-based Code Reviews

[Rigby'13]
[Bacchelli'13]

Change-based Reviews (e.g. in Pull Requests)

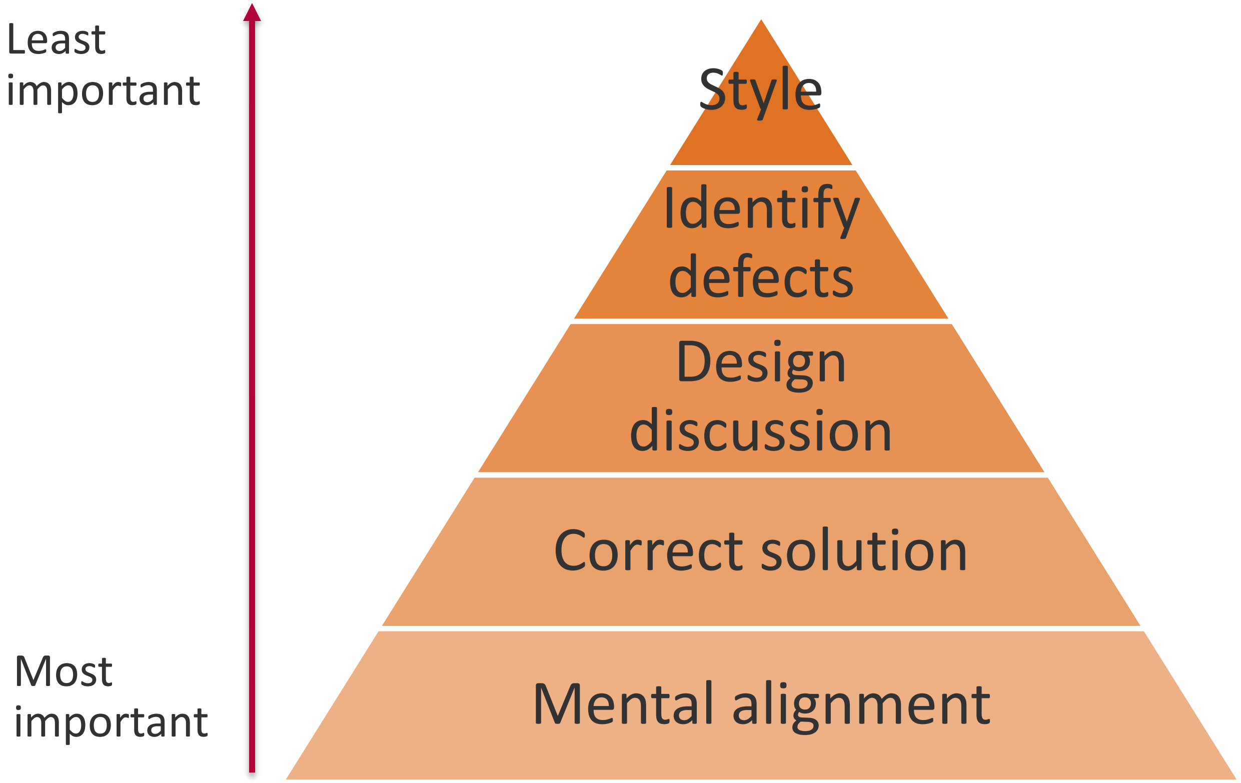
- **Lightweight** process
- Size of reviewed code is *(should be)* **small**
- Performed **regularly** and **quickly**,
  mainly before code enters main branch



Shift in Focus (Compared to Inspections)

- From defect finding to **group problem solving**
- Prefer discussing solutions over reporting defects

Code Review Goals



Priorities of Code Reviews

- Build a **shared mental model**
- Ensure **sane design**
- Find defects vs. understanding code

http://blakesmith.me/2015/02/09/code-review-essentials-for-software-teams.html

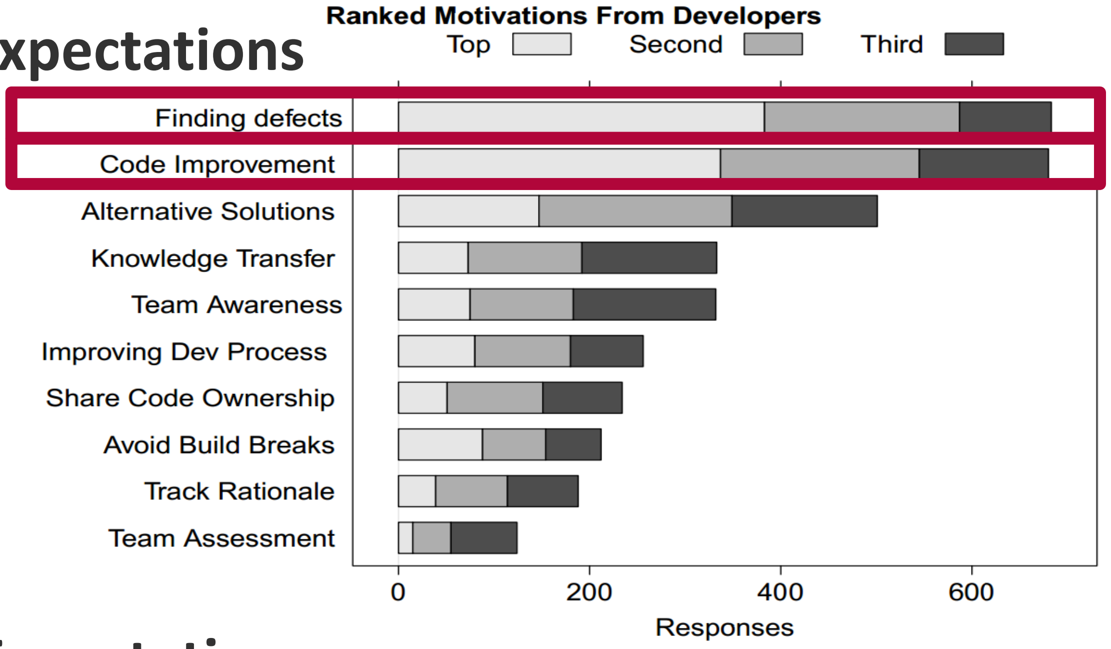
Recent Research

[Bosu’17]
[McIntosh’14]
[Bacchelli ‘13]

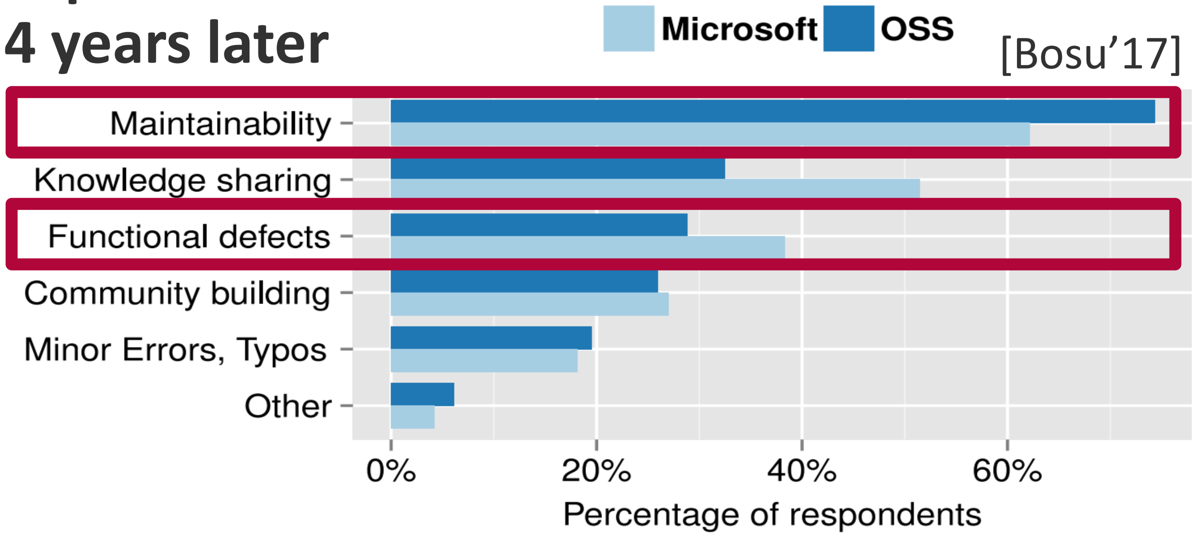
- Code review coverage and review participation share **significant link with software quality**
- Most comments concern code improvements, understandability, social communication
- Only ~15% of comments indicate possible defects
- Developers spend approximately five hours per week (10-15% of their time) in code reviews

Research Findings

Expectations

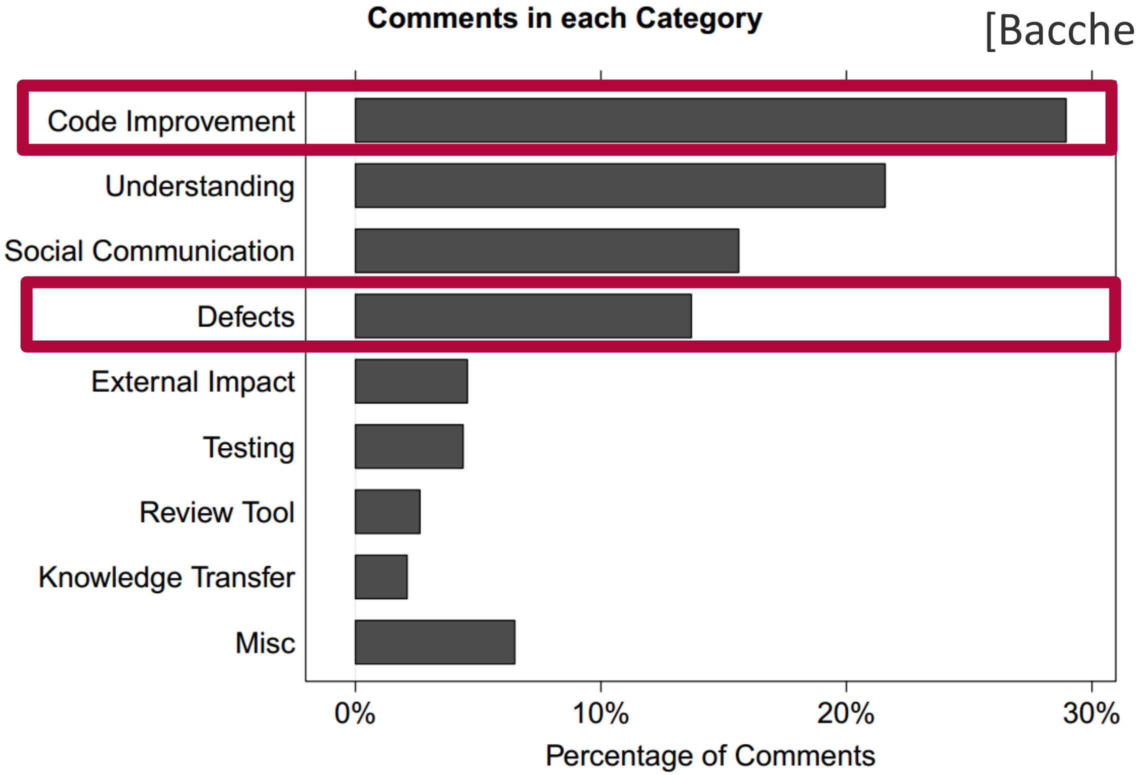


Expectations 4 years later



[Bosu'17]

Empirical study outcomes



[Bacchelli '13]

Maintainability and code improvements identified as most important aspects of modern code reviews

Challenges of Change-Based Review

- **Delay** the shipping of implemented features
- Force reviewers to **switch context**
- Little feedback for **legacy code**

- **Overloading** (too many files), developers create large patches

- **Overcrowding** (too many reviewers), assigning too many reviewers may lower review quality

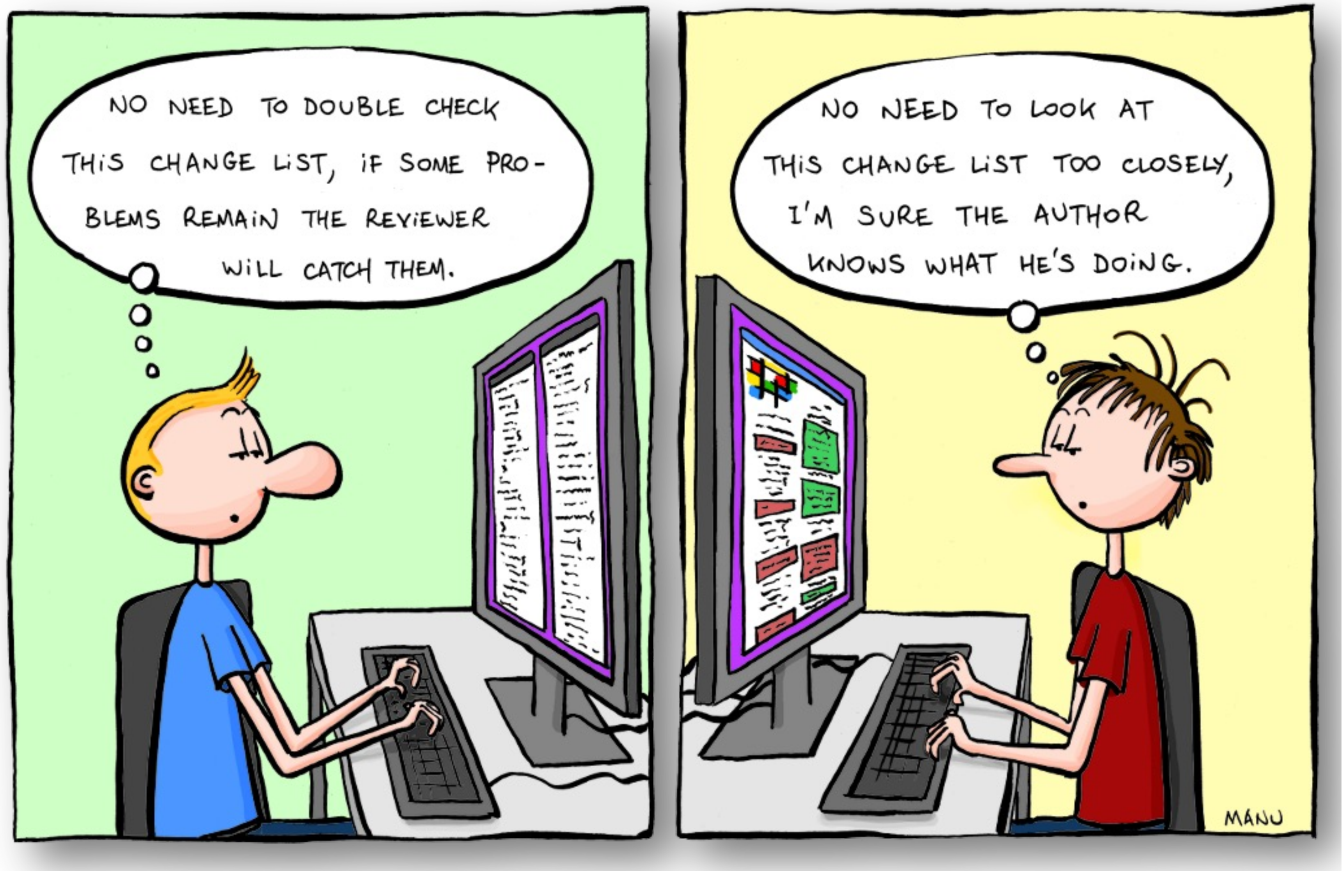

NO NEED TO DOUBLE CHECK THIS CHANGE LIST, IF SOME PROBLEMS REMAIN THE REVIEWER WILL CATCH THEM.
NO NEED TO LOOK AT THIS CHANGE LIST TOO CLOSELY, I'M SURE THE AUTHOR KNOWS WHAT HE'S DOING.
MANU

Image: https://devops.com/dark-side-infrastructure-code/

Reviewer Assignment

Usually, **two reviewers** find optimal number of defects

Reviewer candidates

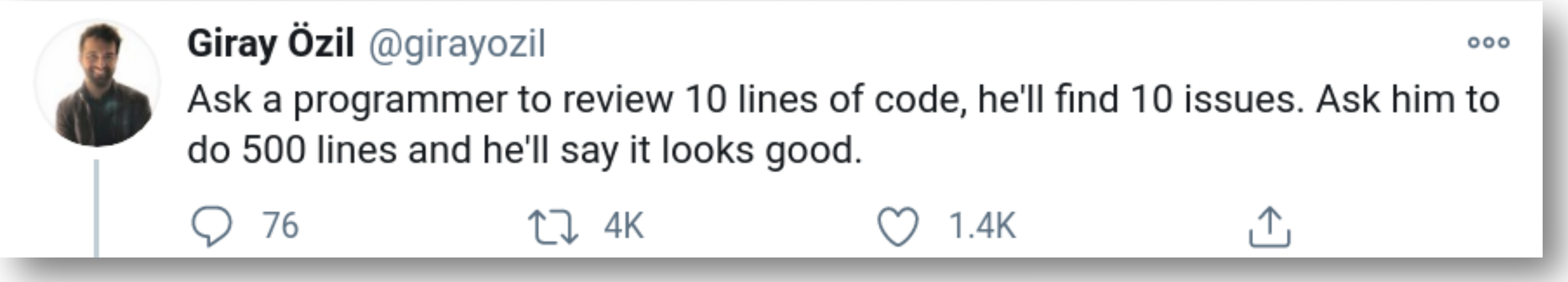
- People who contributed changes (find defects)
- New developers (transfer knowledge)
- Team members with a small review queue
- Reviewers with different fields of expertise

geek & poke
AT LEAST WE DON'T NEED TO OBFUSCATE IT BEFORE SHIPPING
RULE 1: TRY TO FIND AT LEAST SOMETHING POSITIVE

Image: http://geek-and-poke.com/geekandpoke/2010/11/1/how-to-make-a-good-code-review.html

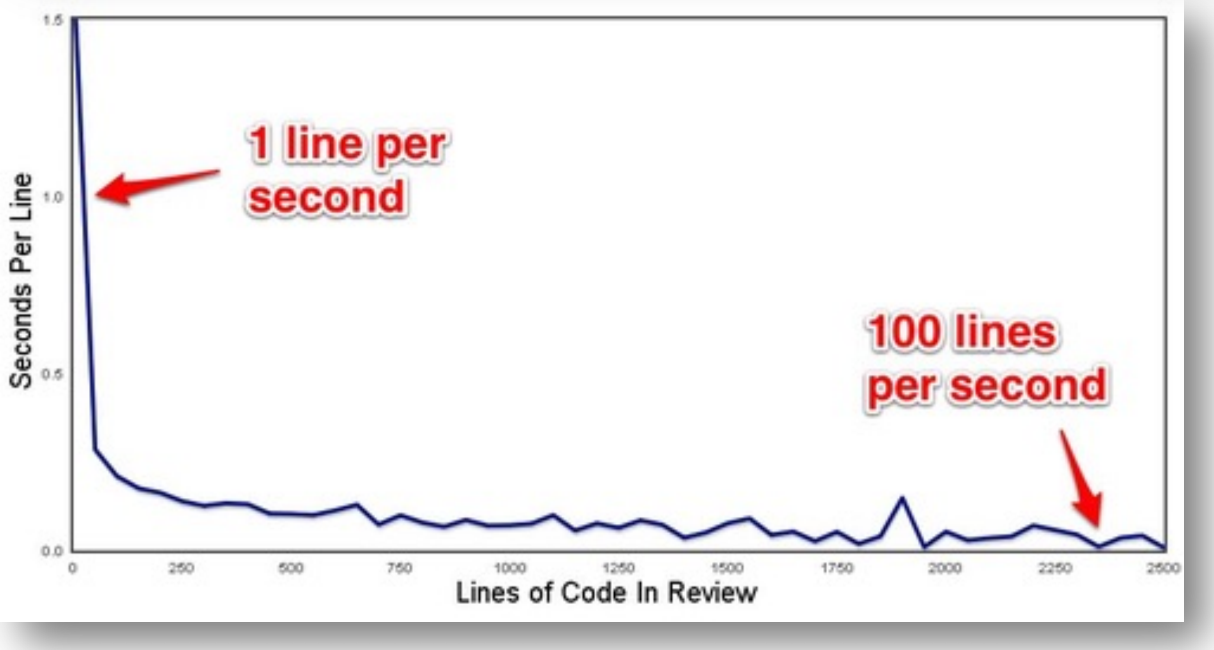
[Rigby'13]

Review Content



Giray Özil @girayozil

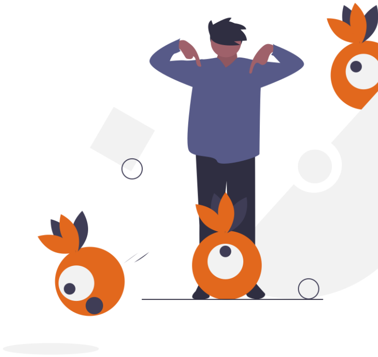
Ask a programmer to review 10 lines of code, he'll find 10 issues. Ask him to do 500 lines and he'll say it looks good.



- Size of artifact to review matters
- **Semantically coherent changes** easier to review than interleaved concerns

Images: http://atlassianblog.wpengine.com/developer/assets_c/2011/07/mt-perloc-thumb-500x263-7290.png
https://twitter.com/girayozil/status/306836785739210752?lang=en

Software Review Helpers



- Testing checks functionality via dynamic execution and assertions
- Code reviews manually check code via **static analysis**

Automated Static Analysis (aka "Linters") *(why's it called that?)*

- Coding conventions (e.g. RuboCop, https://github.com/rubocop-hq/rubocop)
- Code smells (e.g. reek, https://github.com/troessner/reek)

Summary

Software Reviews

- Not a new thing, good reasons to do them (goals & motivation)
- Focus and goals of software reviews
- Review techniques
  - Software Inspections
  - Change-based code reviews
- Reviewer assignment & best practices

References

[Bosu’17] Bosu, Amiangshu, et al. "Process Aspects and Social Dynamics of Contemporary Code Review: Insights from Open Source Development and Industrial Practice at Microsoft." *TSE* 43.1 (2017): 56-75.

[McIntosh’14] McIntosh, Shane, et al. "The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects." *MSR’14.*

[Rigby’13] Rigby, Peter C., and Christian Bird. "Convergent contemporary software peer review practices." *FSE’13*.

[Bacchelli‘13] Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *ICSE’*13.

[Feitelson‘13] Feitelson, Dror G., Eitan Frachtenberg, and Kent L. Beck. "Development and deployment at facebook." *IEEE Internet Computing* 17.4 (2013): 8-17.