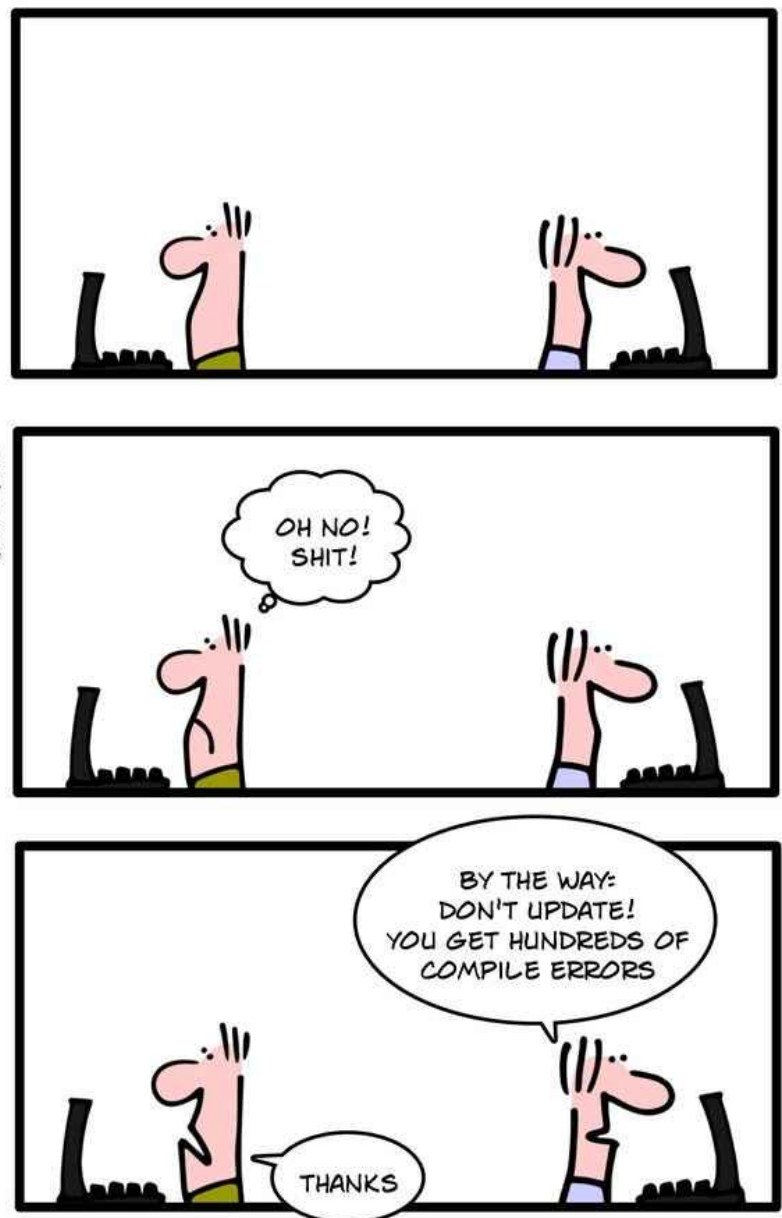
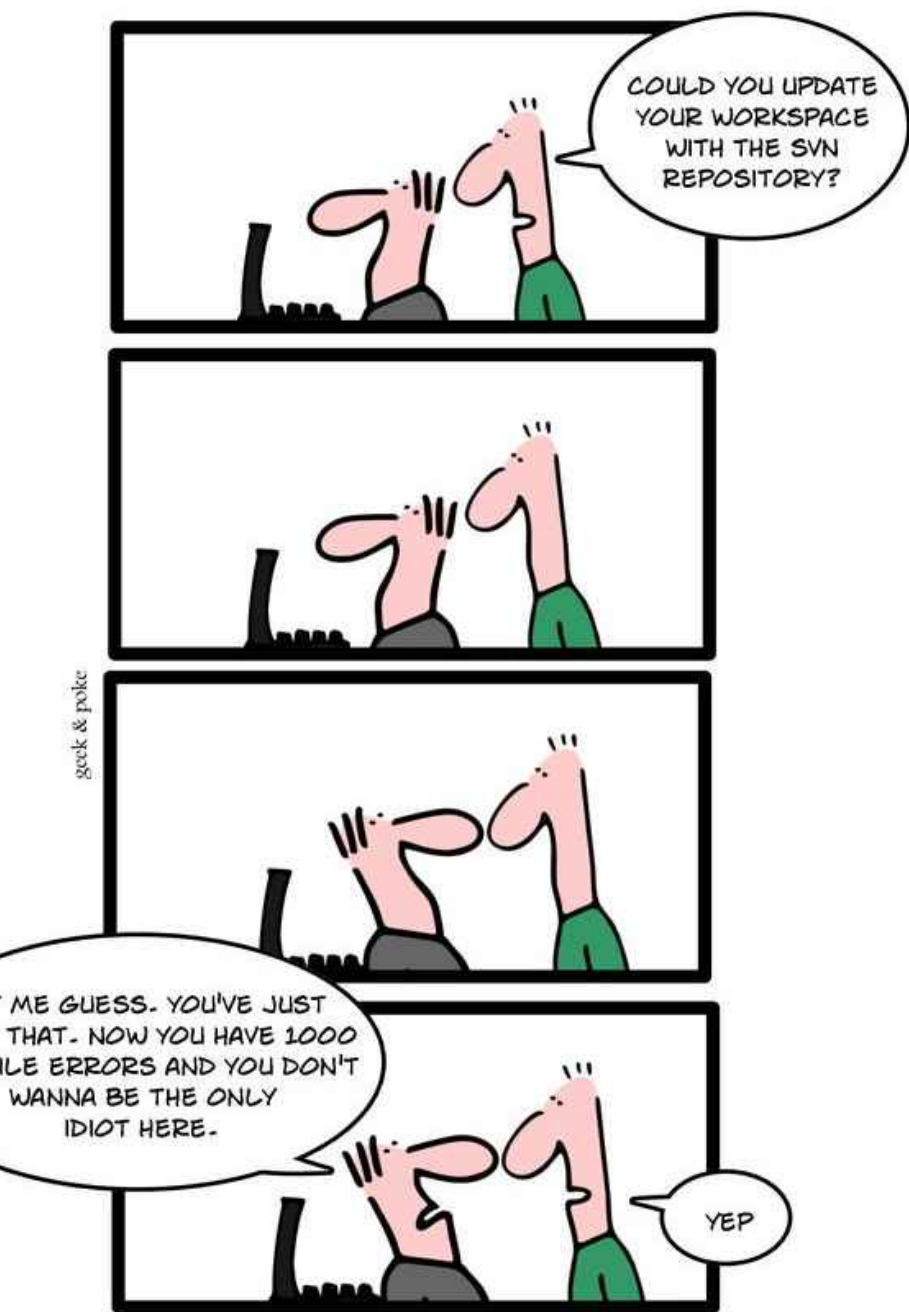


# Inspections, Code Reviews



The Edward S. Rogers Sr. Department  
of Electrical & Computer Engineering  
**UNIVERSITY OF TORONTO**

# REAL CODERS HELP EACH OTHER



1000 AM: UPDATING THE WORKSPACE

# Learning Goals

- Understand different forms of peer reviews with different formality levels.
- Engage in constructive modern code review using a typical commit review system.
- Describe the benefits and properties of good checklists in code review.

# Formal Inspections



Software inspection is a process of finding defects and deficiencies in a developed software product and ensuring that it fulfills the client's requirements.

# Formal Inspections

- Idea popularized in 70s at IBM
- Group of developers meets to formally review code or other artifacts
- Most effective approach to find bugs
  - Typically 60-90% of bugs found with inspections

# Inspection Team and Roles

- Typically 4-5 people (min 3)
- Author
- Inspector(s)
  - Find faults and broader issues
- Reader
  - Presents the code or document at inspection meeting
- Scribe
  - Records results
- Moderator
  - Manages process, facilitates, reports

# Roles

## Author

Programmer or designer who is responsible for producing the program/document that is being inspected.

Leader of the inspection who is responsible for planning the inspection & coordinating it.

## Moderator

## Reader

Present the code at an inspection meeting, where they read the document one by one.

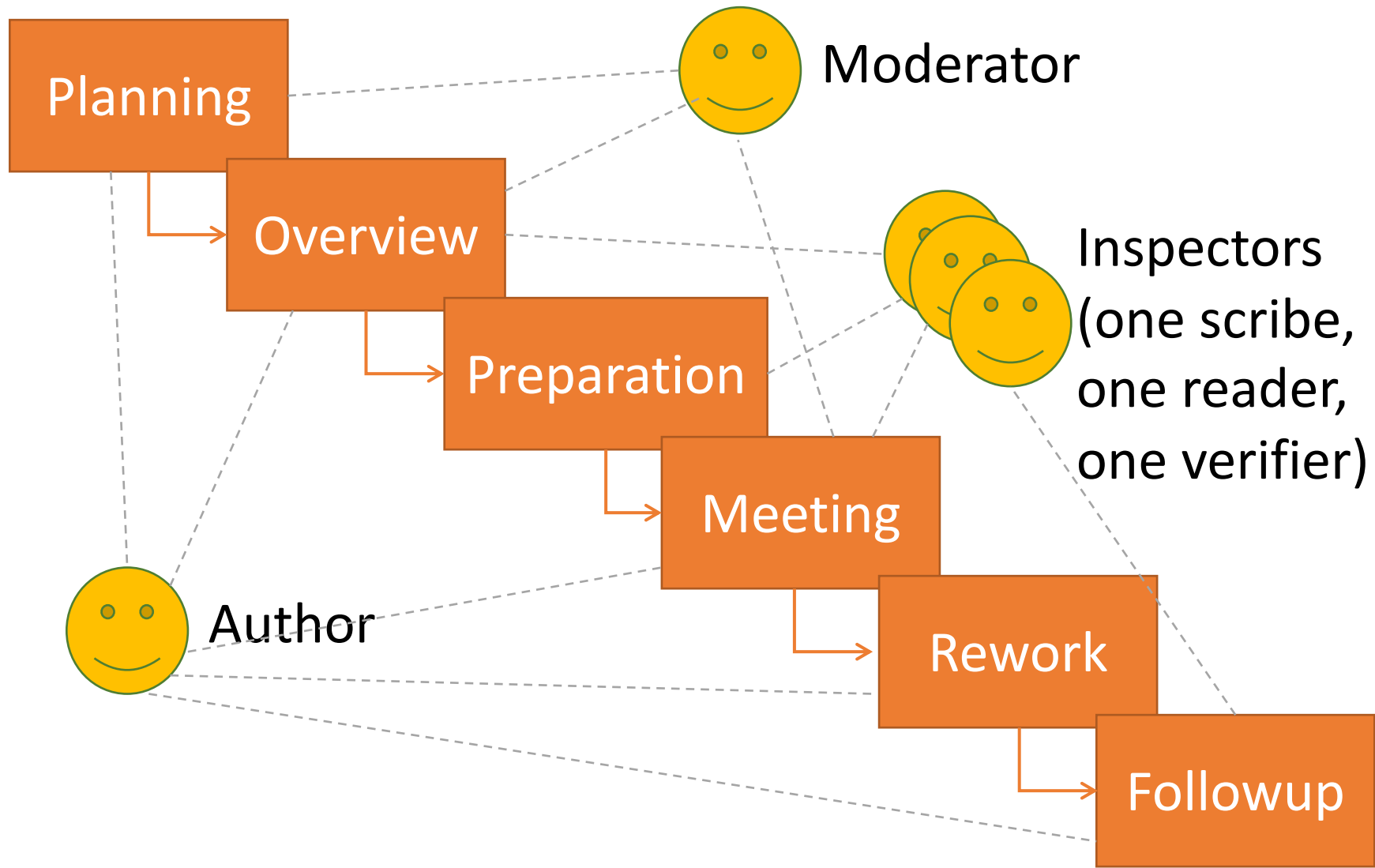
A participant who is responsible for documenting the defects found during the inspection process.

## Recorder/ Scribe

## Inspector

Inspectors are allotted the task of examining the work product to identify possible defects.

<https://www.professionalqa.com/inspection-roles-participants-and-process>





# Checklists

- Include issues detected in the past
- Preferably focus on few important items
- Examples:
  - Are all variables initialized before use?
  - Are all variables used?
  - Is the condition of each if/while statement correct?
  - Does each loop terminate?
  - Do function parameters have the right types and appear in the right order?
  - Are linked lists efficiently traversed?
  - Is dynamically allocated memory released?
  - Can unexpected inputs cause corruption?
  - Have all possible error conditions been handled?
  - Are strings correctly sanitized?



# Process details

- Authors do not explain or defend the code – not objective
  - Author != moderator, != scribe, !=reader
  - Author should still join the meeting to observe questions and misunderstandings and clarify issues if necessary
- Reader (optional) walks through the code line by line, explaining it
  - Reading the code aloud requires deeper understanding
  - Verbalizes interpretations, thus observing differences in interpretation

# BENEFITS:

- Finds defects or faults.
- Improves quality, such as coding style, best practices, etc.
- Assists with debugging.
- Educates and trains developers.
- Communication is improved.
- Collects data for metrics.



# Software Inspection

## Disadvantages:

- Logistics and scheduling can become an issue since multiple people are involved.
  - Time-consuming as it needs preparation as well as formal meetings.
  - It is not always possible to go through every line of code with several parameters and their combination to ensure the correctness of the logic, side effects and appropriate error handling.
- 
- <https://www.educba.com/software-inspection/>





# Modern Code Review

## HOW TO MAKE A GOOD CODE REVIEW

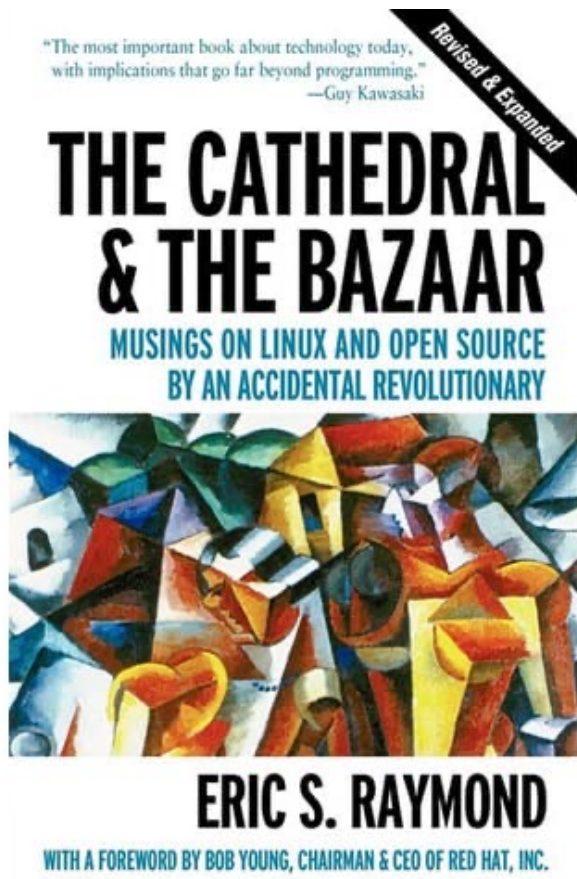


**RULE 1: TRY TO FIND AT LEAST SOMETHING POSITIVE**



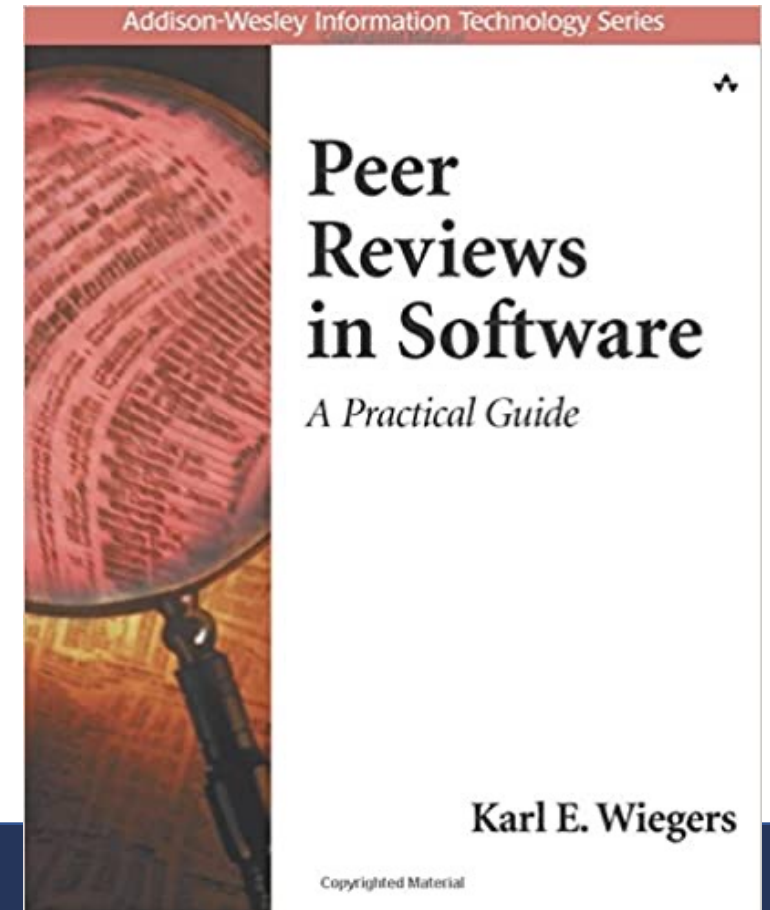
# Code Review

Feedback without noise



**Linus's law:** *"Many eyes make all bugs shallow"*  
----Standard Refrain in Open Source

*"Have peers, rather than customers, find defects"*  
--- Karl Wieggers





# Isn't testing sufficient?

- Only completed implementations can be tested (esp. scalability, performance)
- Design documents cannot be tested
- Tests don't check code quality
- Many quality attributes (eg., security, compliance, scalability) are difficult to test

# A second pair of eyes

- Different background, different experience
- No preconceived idea of correctness
- Not biased by “what was intended”



# Expectations and Outcomes of Modern Code Reviews

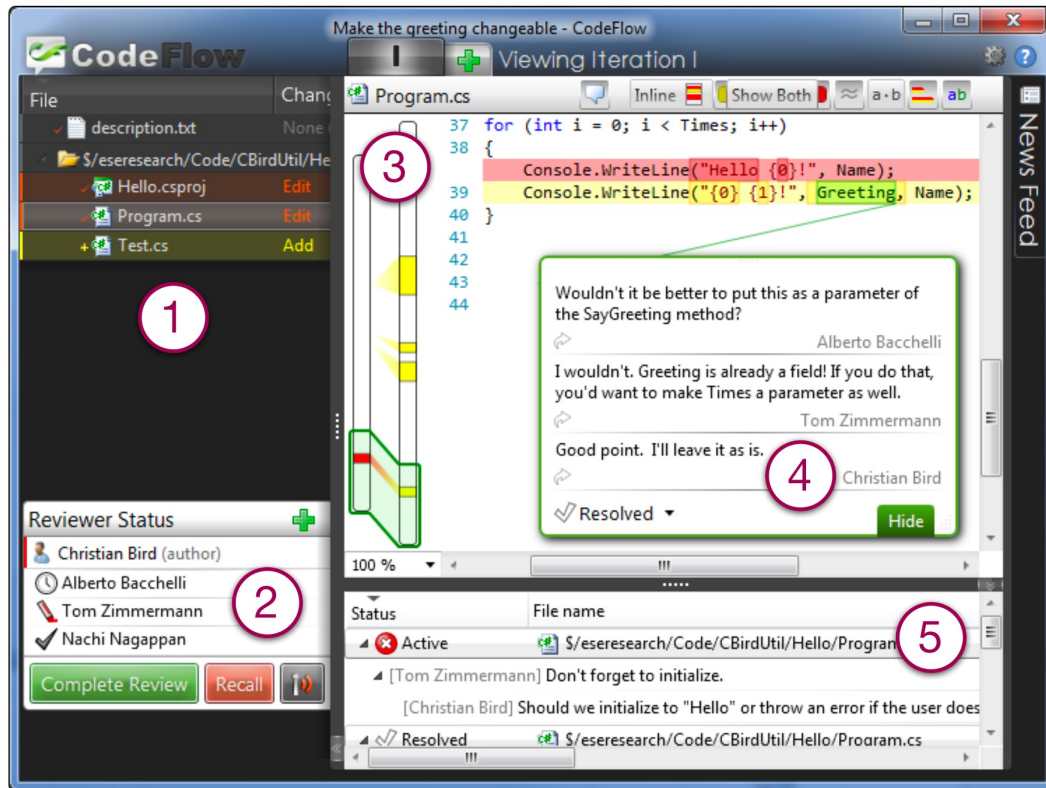
# Reasons for Code Reviews

- Finding defects
  - both low-level and high-level issues
  - requirements/design/code issues
  - security/performance/... issues
- Code improvement
  - readability, formatting, commenting, consistency, dead code removal, naming
  - enforce to coding standards
- Identifying alternative solutions
- Knowledge transfer
  - learn about API usage, available libraries, best practices, team conventions, system design, "tricks", ...
  - "developer education", especially for junior developers

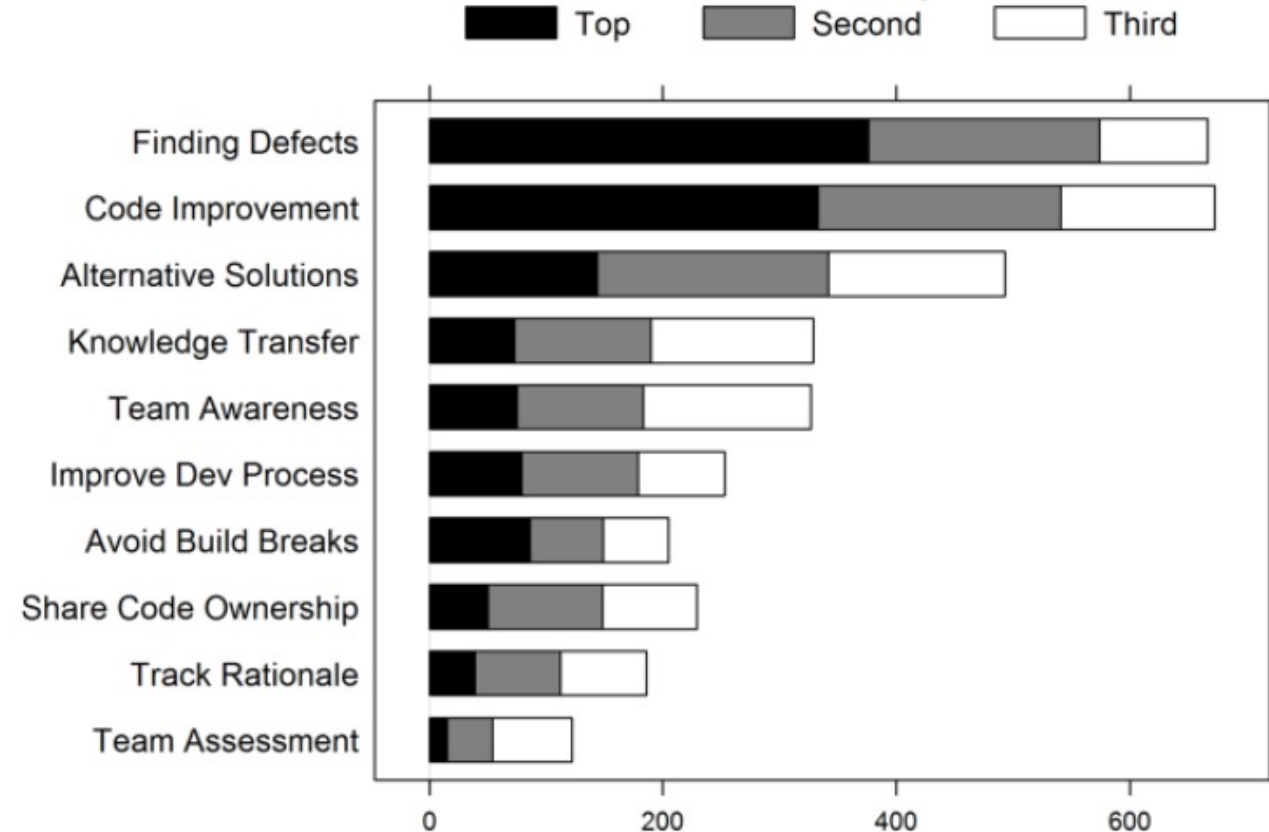
# Reasons for Code Reviews (continued)

- Team awareness and transparency
  - let others "double check" changes
  - announce changes to specific developers or entire team ("FYI")
  - general awareness of ongoing changes and new functionality
- Shared code ownership
  - shared understanding of larger part of the code base
  - openness toward critique and changes
  - makes developers "less protective" of their code

# Code Review at Microsoft



Ranked Motivations From Developers



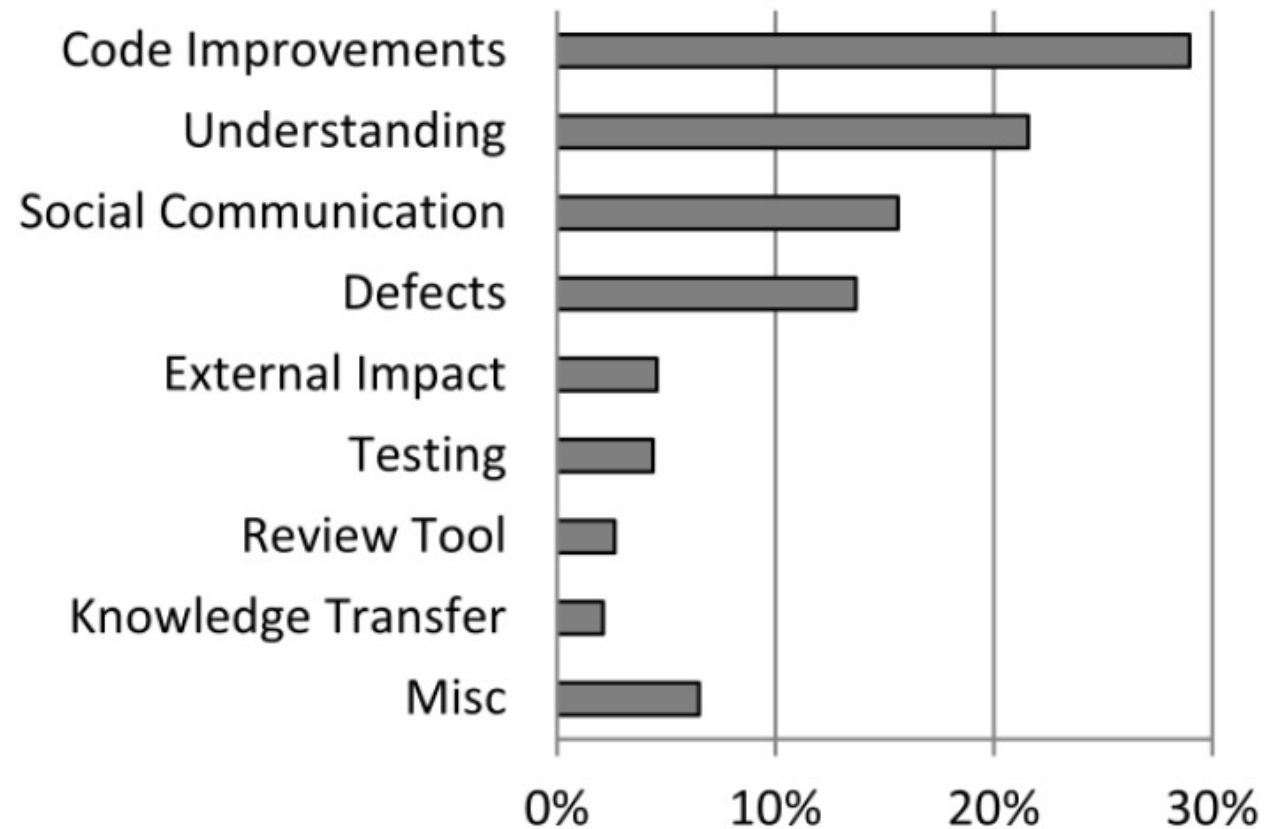
# Outcomes

(at Microsoft analyzing 200 reviews with 570 comments)

- Most frequently code improvements (29%)
  - 58 better coding practices
  - 55 removing unused/dead code
  - 52 improving readability
- Defect finding (14%)
  - 65 logical issues (“uncomplicated logical errors, eg., corner cases, common configuration values, operator precedence)
  - 6 high-level issues
  - 5 security issues
  - 3 wrong exception handling
- Knowledge transfer
  - 12 pointers to internal/external documentation etc



# Outcomes (Analyzing Reviews)





# Mismatch of Expectations and Outcomes

- Low quality of code reviews
  - Reviewers look for easy errors, as formatting issues
  - Miss serious errors
- Understanding is the main challenge
  - Understanding the reason for a change
  - Understanding the code and its context
  - Feedback channels to ask questions often needed
- No quality assurance on the outcome



# Code Review at Google

- Introduced to “*force developers to write code that other developers could understand*”
- 3 Found benefits:
  - checking the consistency of style and design
  - ensuring adequate tests
  - improving security by making sure no single developer can commit arbitrary code without oversight

Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko and Alberto Bacchelli. 2018. Modern Code Review: A Case Study at Google. International Conference on Software Engineering



Google™

<https://www.niallkennedy.com/blog/2006/11/google-mondrian.html>

<https://www.youtube.com/watch?v=sMql3Di4Kgc>

**Peter requests a code review**

Team Explorer - My Work | TestScrum

My Work | TestScrum

In Progress Work

Suspend ▾

**Request Review** | Finish | Actions ▾

23 - Fix paid invoice flagged as not paid

2 edit(s) | View Changes

Suspended Work

Available Work Items

Code Reviews

Team Explorer - New Code Review | TestScrum

New Code Review | TestScrum

2 edit(s) | View Changes

Select one or more reviewers to review your changes and enter a comment for them if appropriate

Add Recent Reviewers ▾

Adam Barr (Fabrikam) ✕

Julia Ilyina (Fabrikam) ✕

Enter the name of a reviewer <optio... ▾

Add Reviewer | Press Enter to add this reviewer

Code Review for Task 23: Fix paid invoice flagged as not paid

TestScrum ▾

EqualTo now allows for rounding errors.

Submit Request Cancel

<https://docs.microsoft.com/en-us/azure/devops/repos/tfvc/day-life-alm-developer-suspend-work-fix-bug-conduct-code-review?view=azure-devops#request-a-code-review>



ninabikes and Copybara-Service Add missing comma. eaff1ef on Aug 4 54 commits

review	Add missing comma.	2 months ago
LICENSE	Internal change	8 months ago
README.md	Internal change	2 months ago
_config.yml	Internal change	13 months ago

README.md

# Google Engineering Practices Documentation

Google has many generalized engineering practices that cover all languages and all projects. These documents represent our collective experience of various best practices that we have developed over time. It is possible that open source projects or other organizations would benefit from this knowledge, so we work to make it available publicly when possible.

Currently this contains the following documents:

- [Google's Code Review Guidelines](#), which are actually two separate sets of documents:
  - [The Code Reviewer's Guide](#)
  - [The Change Author's Guide](#)

## About

Google's Engineering Practices documentation

[google.github.io/eng-practices/](https://google.github.io/eng-practices/)

Readme

View license

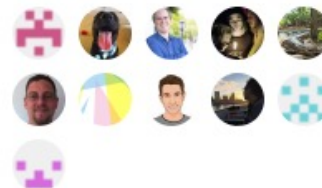
## Releases

No releases published

## Packages

No packages published

## Contributors 11



# Google's Code Review Policy



- All change lists must be reviewed. Period.
- Any CL can be reviewed by any engineer at Google.
- Each directory has a list of owners. At least one reviewer or the author must be an owner for each file that was touched in the commit. If the author is not in the owners file, the reviewer is expected to pay extra attention to how the code fits in to the overall codebase.
- [... readability review ...] If the author does not have readability review, the reviewer is expected to pay extra attention to coding style (both the syntax and the proper use of libraries in that language).
- One can enforce that any CLs to that directory are CC'd to a team mailing list.
- Reviews are conducted either by email, or using a web interface called Mondrian
- In general, the review must have a positive outcome before the change can be submitted (enforced by perforce hooks). However, if the author of the changelist meets the readability and owners checks, they can submit the change TBR, and have a post-hoc review. There is a process which will harass reviewers with very annoying emails if they do not promptly review the change.

source: <https://www.quora.com/What-is-Googles-internal-code-review-policy-process>, 2010



```
or validation helpers */
dation-error {
#b94a48;

dation-valid {
: none;

-validation-error {
```

Team Explorer - Home

Home | Fabrikam Fiber

Project

Web Portal | Task Board | Team Room

My Work ...

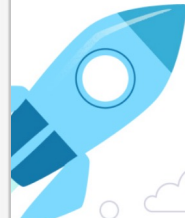
Pending Changes ...

Source Control Explorer

Work Items ...

Builds

Settings



Team Explorer - New Code Review

New Code Review | Fabrikam Fiber

Streaming Video: Using Code Review to improve qual

1 edit(s) | View Changes

Select one or more reviewers to review your changes and enter a comment for them if appropriate

Johnnie McLeod

Add Reviewer | Press Enter to add this reviewer

Hello World border color

Fabrikam Fiber

Changed the border color to #ddd

Submit Request Cancel

Related Work Items

```
or validation helpers */
dation-error {
#b94a48;

dation-valid {
: none;

-validation-error {
1px solid #ddd;

"checkbox"].input-validation-error {
0 none;

-summary-errors {
#b94a48;

-summary-valid {
: none;
```

Team Explorer - My Work

My Work | Fabrikam Fiber

Streaming Video: How to multi-task with My Wo

In Progress Work

Suspend Request Review Check In | Action

1 edit(s) | View Changes

Suspended Work

Resume | Merge with In Progress

No suspended work.

```
or validation helpers */
dation-error {
#b94a48;

dation-valid {
: none;

-validation-error {
1px solid #ddd;
```

Team Explorer - My Work

My Work | Fabrikam Fiber

Streaming Video: How to multi-task with My Wo

In Progress Work

Suspend Request Review Check In | Action

1 edit(s) | View Changes

Suspended Work

Resume | Merge with In Progress

No suspended work.

The image shows a stack of three browser windows from the Phabricator application. The top window displays the Phabricator dashboard with a search bar and navigation icons. The middle window shows a code editor with PHP code for a task view function. The bottom window shows a 'Workboard' view with columns for 'Backlog (Default)', 'Future', 'Far Future', and 'v3', each containing task cards with IDs and descriptions.

```
57 private function buildTasksView(PhabricatorUser $user) {  
58     $viewer = $this->getViewer();  
59  
60     $open = ManiphestTaskStatus::getOpenStatusConstants();
```

Projects > Projects > Workboard

Backlog (Default) 18 | 0

- T12144 Ability to reorder milestones on a project's workboard (Feature Request)
- T11974 Projects should allow setting of any icon
- T11923 Unhandled Exception - Cant import columns for new workboard

Future 3 | 0

Far Future 5 | 0

- T10266 Firefox body scroll fails when dragging tasks (Workboards, Firefox)
- T7190 Project templates/Copying tasks (Maniphest)
- T6502 Policy to define who can move

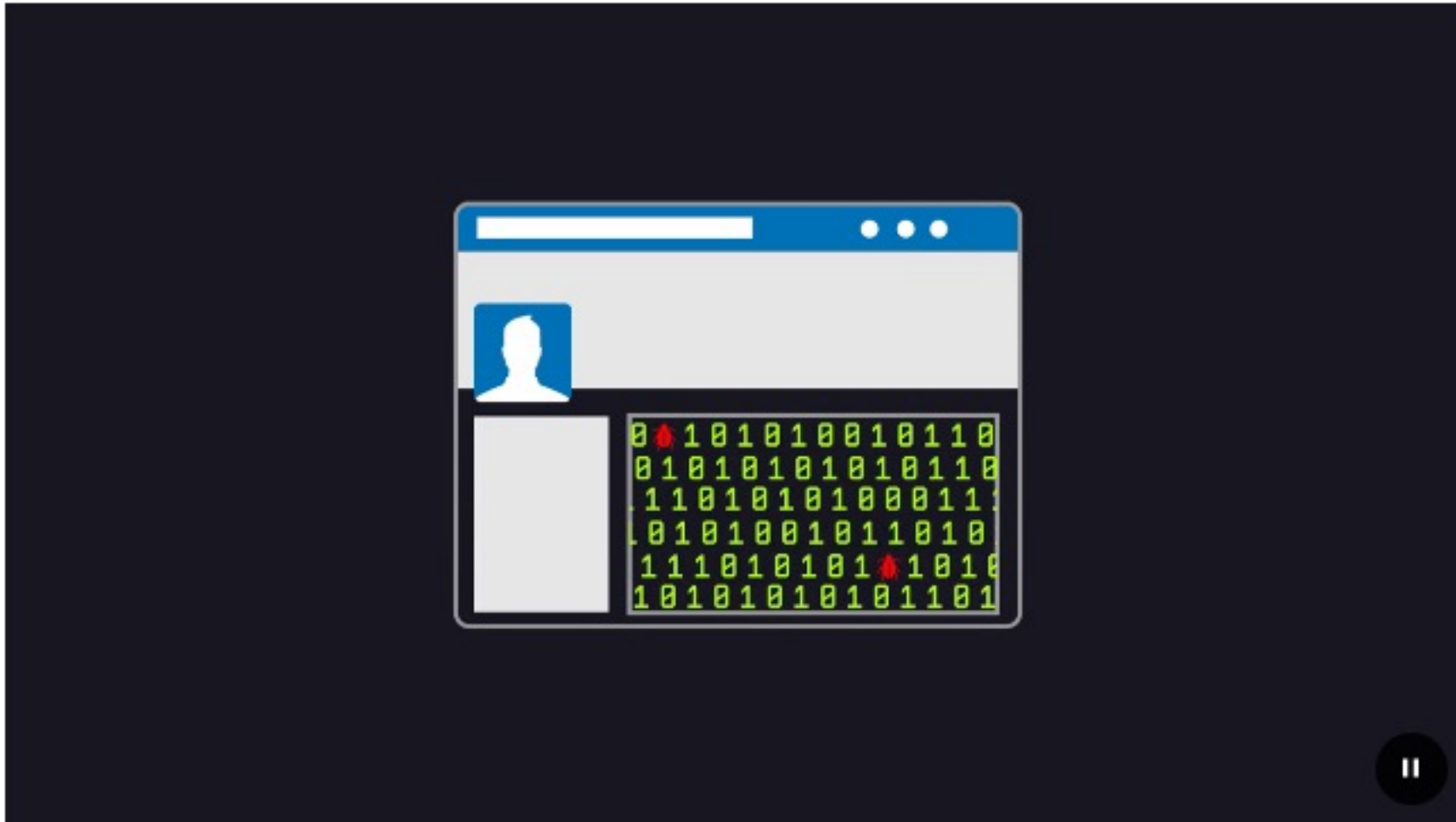
v3 6 | 12

- T11036 Put subproject columns on workboards
- T10350 Scripts to migrate old "points" fields and move projects beneath other projects
- T10349 Projects v3 Errata

<https://github.com/phacility/phabricator>

# How Facebook Catches Bugs in Its 100 Million Lines of Code

For the past four years, Facebook has quietly used a homegrown tool called Zoncolan to find bugs in its massive codebase.

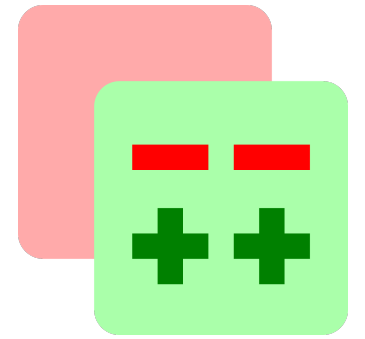


ELENA LACEY

<https://www.wired.com/story/facebook-zoncolan-static-analysis-tool/>



# Gerrit Code Review



rietveld-codereview / rietveld

Watch 67 Star 465 Fork 177

Code Issues 235 Pull requests 3 Actions Projects Wiki Security

master Go to file Add file Code About

andialbrecht Merge pull request #569 from cedk/oa... on Apr 28, 2019 1,450

codereview Remove XMPP notification 3 years ago

codereview.appspot.com

## Rietveld Code Review Tool

Issues Repositories Search

Open Issues | Closed Issues | All Issues | Sign in with your Google Account to create issues and add comments

## Recent Open Issues

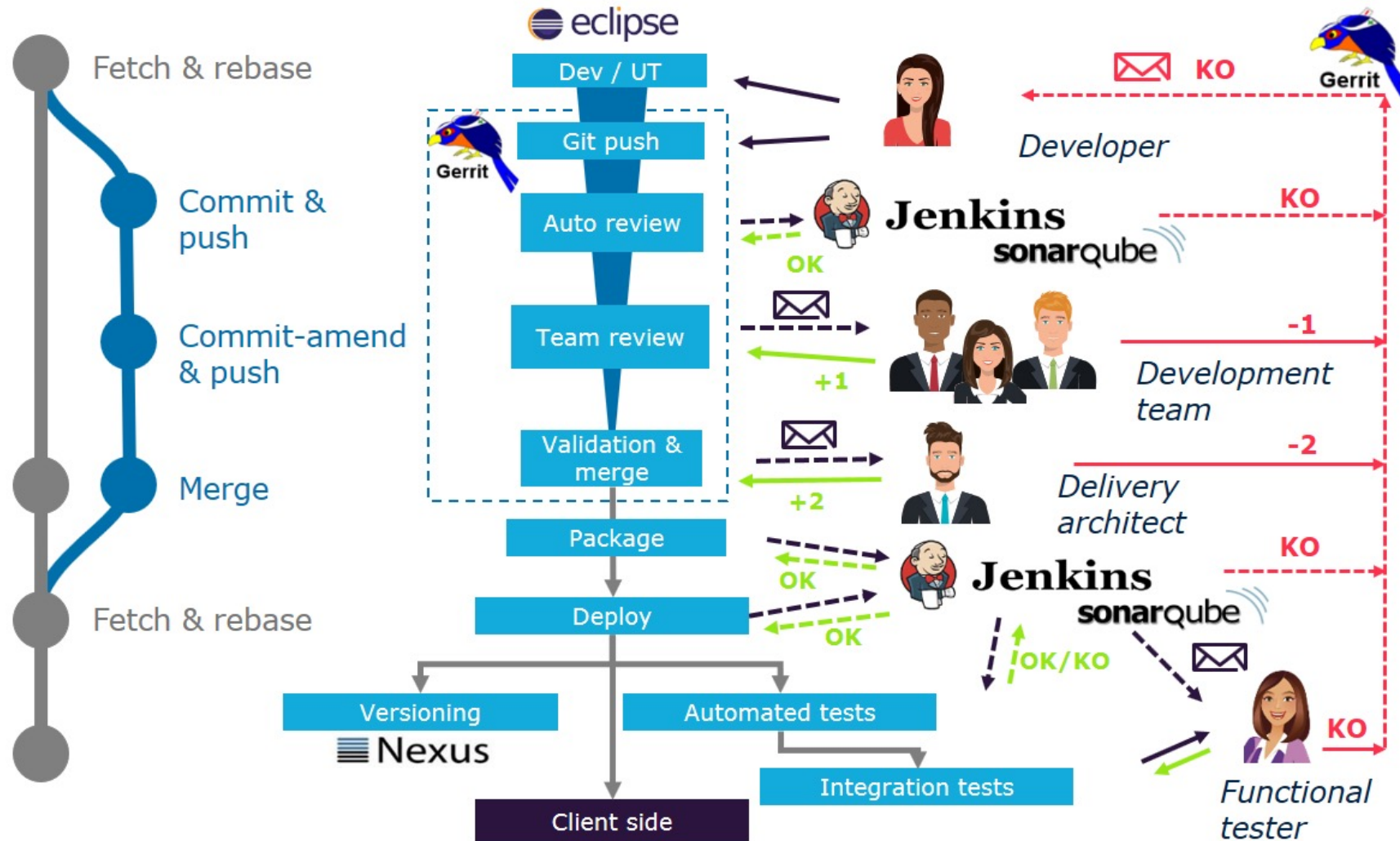
Id	Subject	Owner	Reviewers
566080043	Clean up and fix glyph contour generation	hanwenn	Dan Eble, lemwerg, hahnjo, aaireill111
277230044	class of service	Tom Henderson	Stefano Avallone, Tommaso Pecorella
555820043	flask-tryton: Convert records ids to string before joining them	pokoli	ced
6493067	Test code review post	ItzmE29	gallion342, baby546833, itzme29, bichph
1758041	code review 1758041: io/ioutil: add CopyFile	Kyle Lemons	eliarconcepcion, adg, 周
545890043	flask-tryton: Add support for 5.6 series	pokoli	ced
554030043	Issue 4182: avoid checking the offset of cross-staff stems too early	barrykp	Dan Eble, lemwerg
569700043	Split glyph contours in up/down segments for skylines	hanwenn	dak, lemwerg, hahnjo
560030044	Remove deprecated context properties	Valentin Villenave	thomasmorley651, lemwerg
576090043	Fix #5964: MM Rests shouldn't segfault when there's no StaffSymbol.	Valentin Villenave	Dan Eble, dan, carl.d.sorensen
576410043	break substitution cleanup	hanwenn	

*“As I've learned over the last two years at Google, where I developed a similar tool named Mondrian, proper code review habits can really improve the quality of a code base, and good tools for code review will improve developers' life.”*

<https://github.com/rietveld-codereview/rietveld>

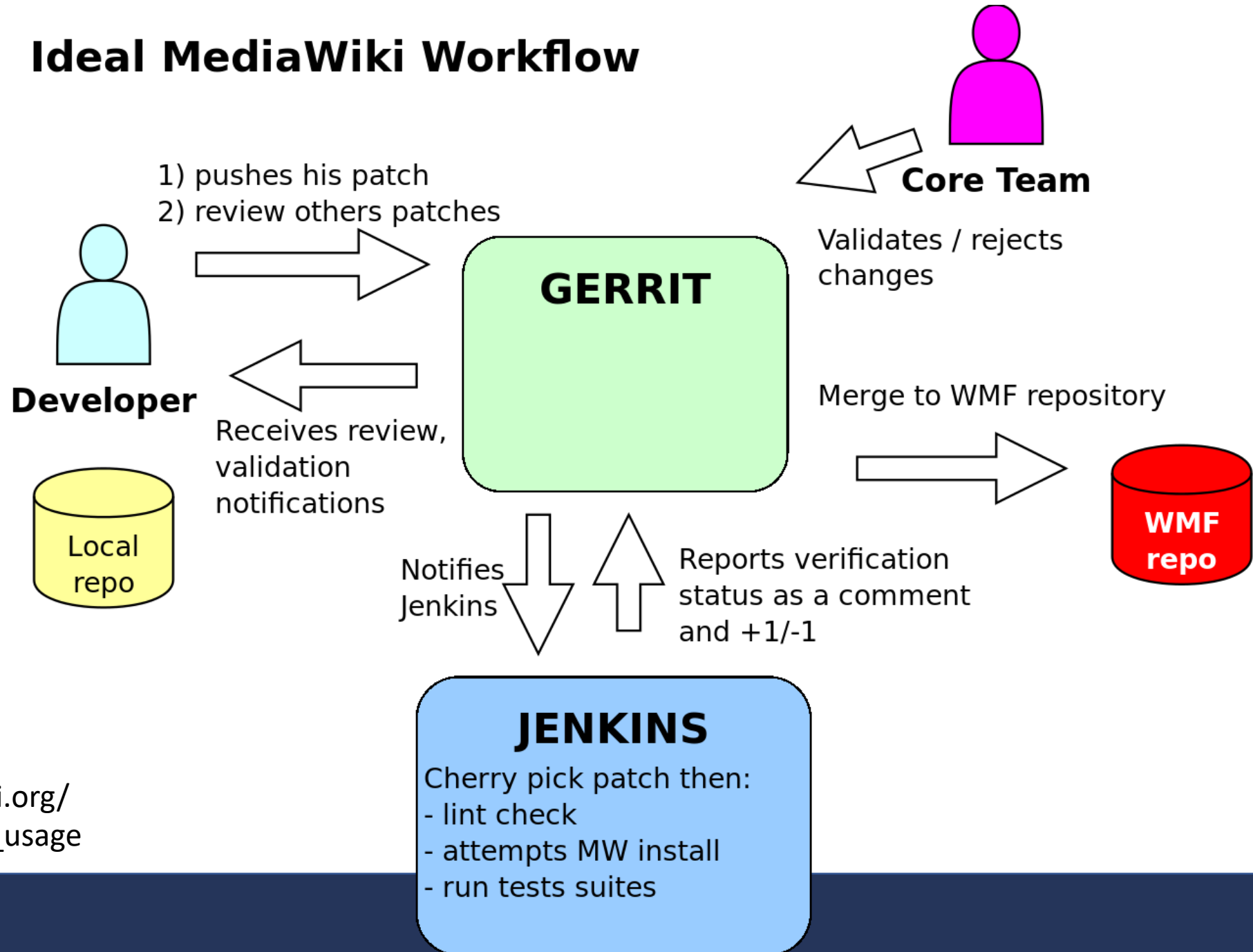


# Gerrit Code Review



<https://bcouetil.gitlab.io/academy/BP-gerrit.html>

# Ideal MediaWiki Workflow



[http://www.mediawiki.org/wiki/Gerrit/Advanced\\_usage](http://www.mediawiki.org/wiki/Gerrit/Advanced_usage)



# Process: Checklists!



The Checklist:

<https://www.newyorker.com/magazine/2007/12/10/the-checklist>

## OFFICIAL A.A.F. PILOT'S CHECK LIST

B-17F AND B-17G

For detailed instructions see Pilot's Handbook AN 01-20EF-1 or  
AN 01-20EG-1 in data case

### PILOT

#### BEFORE STARTING

1. Pilot's Pre-flight — Complete.
2. Form IA, Form F, Weight and Balance — Checked.
3. Controls and Seats — Checked —  
Checked.
4. Fuel Transfer Valves and Switch —  
Off.
5. Intercoolers — Cold.
6. Gyros — Uncaged.
7. Fuel Shut-off Switches — Open.
8. Gear Switch — Neutral.
9. Cowl Flaps — Open Right — Open  
Left — Locked.
10. Turbos — Off.
11. Idle cut-off — Checked.
12. Throttles — Closed.
13. High RPM — Checked.
14. Auto Pilot — Off.
15. De-icers and Anti-icers Wing and  
Prop. — Off.
16. Cabin heat — Off.
17. Generators — Off.

#### STARTING ENGINES

1. Fire Guard and Call Clear — Left-  
Right.
2. Master Switches — On.
3. Battery Switches and Inverters —  
On and Checked.
4. Parking Brakes — Hydraulic Check-  
On — Checked.
5. Booster Pumps — Pressure — On  
and Checked.
6. Carburetor Filters — Open.
7. Fuel Quantity — Gallons per tank.
8. Start Engines
  - a. Fire Extinguisher Engine Selec-  
tor — Checked.
  - b. Prime — As Necessary.

### CO-PILOT

#### BEFORE TAKE OFF

1. Tail Wheel — Locked.
2. Gyro — Set.
3. Generators — On.

#### AFTER TAKE OFF

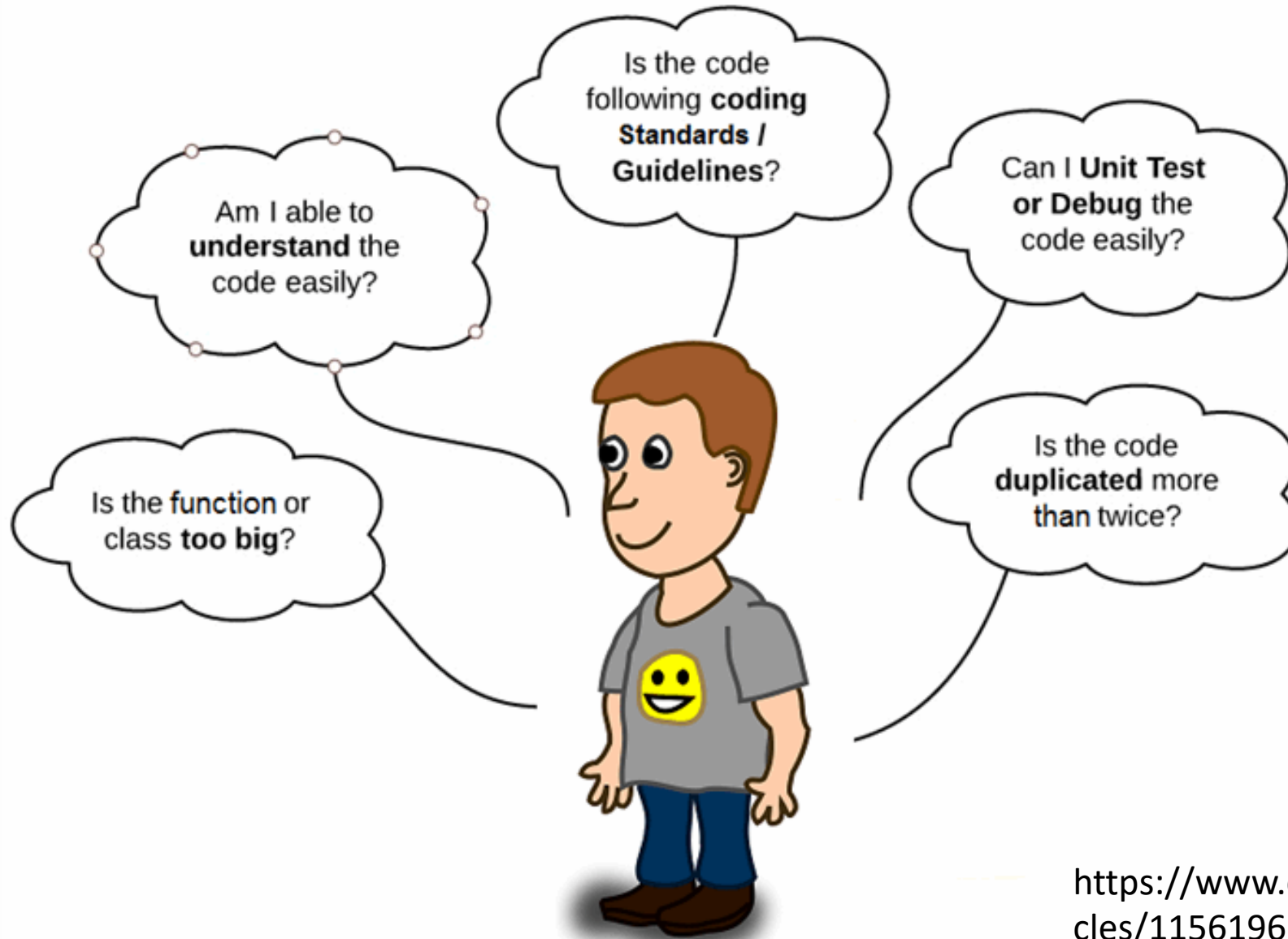
1. Wheels — Pilot's Signal.
2. Power Reduction.
3. Cowl Flaps.
4. Wheel Check — OK Right.  
OK Left.

#### BEFORE LANDING

1. Radio Call Altimeter — Set.
2. Crew Positions — OK.
3. Auto Pilot — Off.
4. Booster Pumps — On.
5. Mixture Controls — Auto Rich.
6. Intercooler — Set.
7. Carburetor Filters — Open.
8. Wing De-icers — Off.
9. Landing Gear
  - a. Visual — Down right  
Down left  
Tail wheel  
Down,  
Antenna In
  - b. Light — OK.
  - c. Switch Off — Neutral.
10. Hydraulic Pressure — OK. Valve  
closed.
11. RPM 2100 — Set.
12. Turbos — Set.
13. Flaps  $\frac{1}{3}$  —  $\frac{1}{3}$  Down

#### FINAL APPROACH

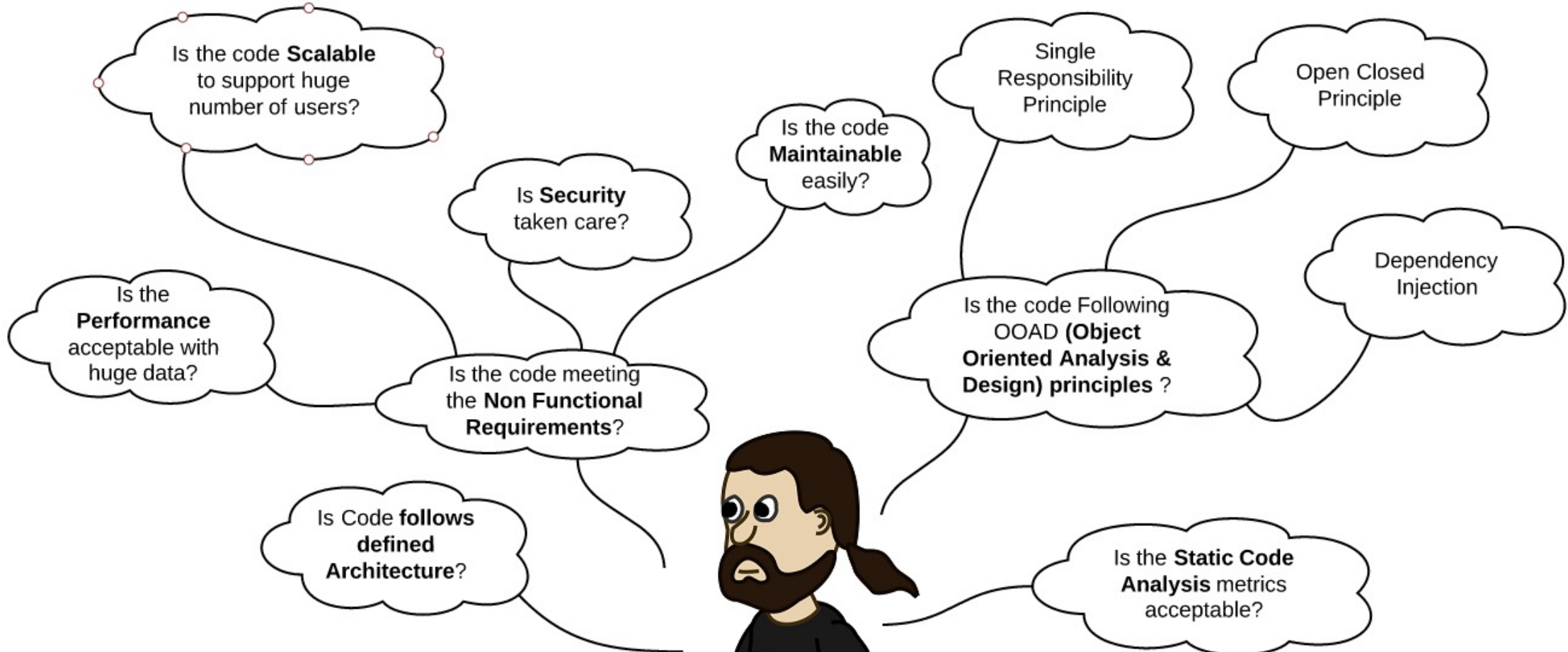
14. Flaps — Pilot's Signal.
15. High RPM — Pilot's Signal.



### Basic Code Reviewer

<https://www.codeproject.com/Articles/1156196/Code-Review-Checklist>





**Expert Code Reviewer**

# Your Code Review Checklist: 14 Things to Include

<https://www.codementor.io/blog/code-review-checklist-76q7ovkaqj>

Conduct code reviews on more than your gut instinct. Follow this checklist to conduct a great code review.

## Code Review Checklist: 14 Things to Include

 codementor



<https://www.coursera.org/lecture/introduction-git-github/how-to-use-code-reviews-in-github-QH52K>



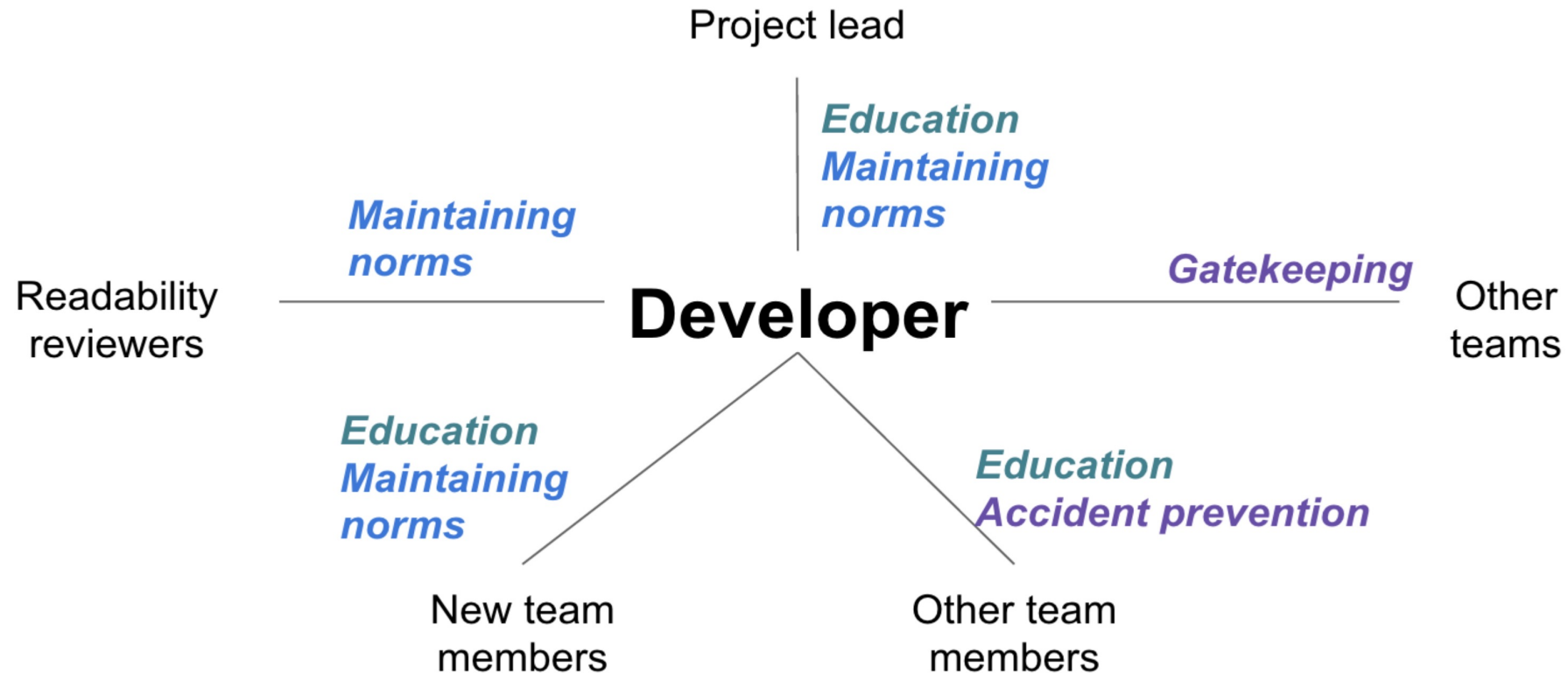
Upsource

# Code Review Best Practices

Trisha Gee



# Reviewing Relationships



# How to write code review comments

- Be kind.
- Explain your reasoning.
- Balance giving explicit directions with just pointing out problems and letting the developer decide.
- Encourage developers to simplify code or add code comments instead of just explaining the complexity to you.

***Bad:** “Why did **you** use threads here when there’s obviously no benefit to be gained from concurrency?”*

***Good:** “The concurrency model here is adding complexity to the system without any actual performance benefit that I can see. Because there’s no performance benefit, it’s best for this code to be single-threaded instead of using multiple threads.”*

<https://google.github.io/eng-practices/review/reviewer/comments.html>


# Social Issues: Egos in Inspections

- Author's self-worth in artifacts
- Identify defects, not alternatives; do not criticize authors
  - “you didn't initialize variable a” -> “I don't see where variable a is initialized”
- Avoid defending code; avoid discussions of solutions/alternatives
- Reviewers should not “show off” that they are better/smarter
- Avoid style discussions if there are no guidelines
- Author decides how to resolve fault

# Social Issues 2

- Moderator must move discussion along, resolve conflicts
- Meetings should not include management
- Do not use for HR evaluation
  - “finding more than 5 bugs during inspection counts against the author”
  - Leads to avoidance, fragmented submission, not pointing out defects, holding pre-reviews
- Responsibility for quality with authors, not reviewers
  - “why fix this, reviewers will find it”

# Types of Code Reviews by Formality

- 
- Ad hoc review
  - Passaround (“modern code reviews”)
  - Pair programming
  - Walkthrough
  - Inspection

More formal

Source: Wiegers. Peer Reviews in Software. Addison-Wesley 2002

# Differences among peer review types

Review Type	Planning	Preparation	Meeting	Correction	Verification
Formal Inspection	Yes	Yes	Yes	Yes	Yes
Walkthrough	Yes	Yes	Yes	Yes	No
Pair Programming	Yes	No	Continuous	Yes	Yes
Passaround	No	Yes	Rarely	Yes	No
Ad Hoc Review	No	No	Yes	Yes	No

Source: Wieggers. Peer Reviews in Software. Addison-Wesley 2002

# Summary

- Code reviews effective to identify bugs
- Additional benefits (e.g., knowledge transfer, shared code ownership, awareness)
- Reviews require understanding
- Different review types with different formality
- Formal inspection require planning & social skills, are expensive, but very effective



# Further Reading

- Sommerville. Software Engineering. 8<sup>th</sup> Edition. Addison-Wesley 2007. Chapter 22.2
  - Overview of formal inspections
- Wiegers. Peer Reviews in Software. Addison-Wesley 2002
  - Entire book on formal inspections; how to run them and how to introduce them
- Bacchelli and Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.
  - Detailed studies of modern code reviews at Microsoft
- Oram and Wilson (ed.). Making Software. O'Reilly 2010. Chapter 18
  - Overview of empirical research on formal inspections

→ ↻ google.github.io/styleguide/

## styleguide

# Google Style Guides



Every major open-source project has its own style guide: a set of conventions (sometimes arbitrary) about how to write code for that project. It is much easier to understand a large codebase when all the code in it is in a consistent style.

“Style” covers a lot of ground, from “use camelCase for variable names” to “never use global variables” to “never use exceptions.” This project ([google/styleguide](https://google.github.io/styleguide/)) links to the style guidelines we use for Google code. If you are modifying a project that originated at Google, you may be pointed to this page to see the style guides that apply to that project.

This project holds the [C++ Style Guide](#), [C# Style Guide](#), [Swift Style Guide](#), [Objective-C Style Guide](#), [Java Style Guide](#), [Python Style Guide](#), [R Style Guide](#), [Shell Style Guide](#), [HTML/CSS Style Guide](#), [JavaScript Style Guide](#), [AngularJS Style Guide](#), [Common Lisp Style Guide](#), and [Vimscript Style Guide](#). This project also contains [cplint](#), a tool to assist with style guide compliance, and [google-c-style.el](#), an Emacs settings file for Google style.

<https://google.github.io/styleguide/>

# Further Reading 2



## Awesome Code Review awesome

A curated list of tools, articles, books, and any other resource related to [code review](#)

Code review is the systematic examination (sometimes referred to as peer review) of computer source code.

### Contents

- [Academic Papers](#)
- [Articles](#)
- [Books](#)
- [Talks and Podcasts](#)
- [Tools](#)

- <https://github.com/joho/awesome-code-review>