Code Inspection and Code Review

One-Slide Summary

- In a **code review**, another developer examines your proposed change and explanation, offers feedback, and decides whether to accept it. Modern code reviews have significant tool support.
- In a **(formal) code inspection**, a team of developers meets and examines existing code, following a process to understand it and spot issues.
- Both of these static quality assurance approaches have costs and benefits.

The Story So Far …

- Quality assurance is critical to software engineering.
- Testing is the most common dynamic ("run the program") approach to QA.
  - We can generate some test inputs and oracles, but testing remains very expensive.

- What about static ("look at the program") approaches to QA?

Intuition

“Given enough eyeballs, all bugs are shallow.”

– Linus's Law

“Have peers, rather than customers, find defects.”

– Karl Wiegers

Example of Both: Twilight

[http://reasoningwithvampires.tumblr.com/]

The Cullens and the Hales sat at the same table as always, not eating, talking only among themselves. None of them, especially Edward, glanced my way anymore.

One person cannot do nothing more than other people who are also doing nothing.

Example:

None of them bought an apple, especially Edward.

I went ~~to go try~~ to watch TV ~~while I waited~~.

Not only did Meyer make the sentence far more complicated than necessary, but she also made the act of watching television complicated.

Our heroine is so insecure that she actually doubts her ability to succeed at watching television by herself.

Our Bodies, Ourselves: *A Mystery*

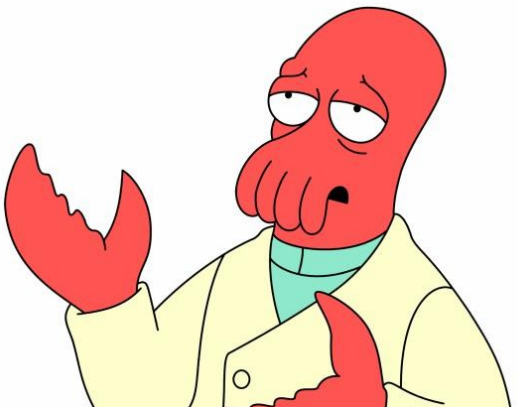
By Isabella M. Swan

Yes, I wanted to say. Anything. But I couldn't find my lips.

I tried to obey, though I couldn't quite locate my lungs.

Why not simply test?

- Faults can mask other faults at runtime
- Only completed implementations can be tested (esp. scalability, performance)
- Many quality attributes (eg., security, compliance, maintainability) are hard to test
- Non-code artifacts (e.g., design documents) cannot be tested



A Second Pair of Eyes

- Different background, different experience
- No preconceived idea of correctness
- Not biased by “what was intended”
- “Breadth of experience in an individual is essential to creativity and hence to good engineering. … Collective diversity, or diversity of the group - the kind of diversity that people usually talk about - is just as essential to good engineering as individual diversity. … Those differences in experience are the "gene pool" from which creativity springs.”

– Bill Wulf, Nat. Academy of Engineering President

What To Examine

- **Code Inspection**: Examine Whole Program
  - Expensive if the program changes
  - Good if a new concern arises
- **Code Review**: Examine Each Change
  - Inductive Argument:
    - V-0 is good, V-n is good → V-n+1 is good
  - Bad if the definition of “good” changes

Code Inspection Example: It's A Bug Hunt!

```
year = ORIGINYEAR; /* = 1980 */
while (days > 365) {
     if (IsLeapYear(year)) {
          if (days > 366) {
               days -= 366;
               year += 1;
          }
     } else {
          days -= 365;
          year += 1;
     }
}
```

Leap-year glitch freezes Zune MP3 players

STORY HIGHLIGHTS
- NEW: Microsoft says proble
- Thousands of older 30GB Z
- Message boards are buzzin
- User: "It seems that every Z

Next Article in Technology »

READ VIDEO

By Brandon Griggs
CNN

Decrease font Enlarge font

(CNN) -- A leap-year related glitch caused thousands of Zune MP3 players to simultaneously stop working late Tuesday and early Wednesday, Microsoft said on the product's Web site.

GETTY IMAGES/FILE

Microsoft issued the first Zune portable music player in 2006 to compete with the iPod.

The problem should resolve itself after 7 a.m. ET Thursday, Matt Akers of the Zune Product Team wrote on Zune.net. "A bug in the internal clock driver related to the way the device handles a leap year" is to blame, he said.

The issue was limited to older Zune 30GB models, the Web site said.

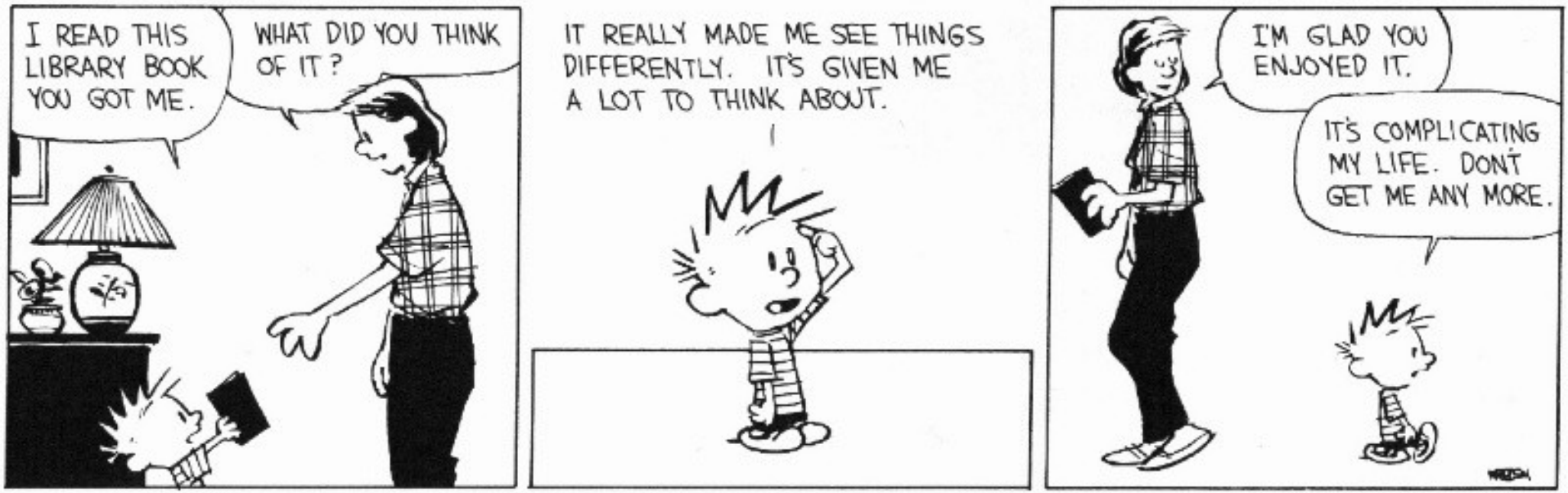
The Zune support page says users should allow the internal battery to fully drain. Then, after noon GMT on January 1, 2009 (7 a.m. ET), users should recharge by connecting the Zune to a computer or AC power.

Internet message boards were flooded with complaints about Zunes freezing, prompting Y2K-like speculation about end-of-year hardware or software problems.

"It seems that every Zune on the planet has just frozen up and will not work," posted a Mountain Home, Idaho, user on CNN's iReport.com. "I have 3 and they all in the same night stopped working."

Code Review

- What is code review?
- What is different between code review and code inspection in practice?



GitHub

- **Pull requests** let you tell others about changes you've pushed to a [Git] repository. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before the changes are merged into the repository.
- Other contributors can review your proposed changes, add review comments, contribute to the pull request discussion, and even add commits to the pull request.

Refactorings #28

New issue

Merged **joliebig** merged 17 commits into `liveness` from `CallGraph` 9 months ago

Conversation 3 Commits 17 Files changed 97 +1,149 −10,129

ckaestne commented on Jan 29 Owner

@joliebig
Please have a look whether you agree with these refactorings in CRewrite

key changes: Moved ASTNavigation and related classes and turned EnforceTreeHelper into an object

ckaestne added some commits on Jan 29

- remove obsolete test cases 02dddb6
- refactoring: move AST helper classes to CRewrite package where it is … f8fc311
- improve readability of test code 7e61a34
- removed unused fields ✔ f35b398

ckaestne commented on Jan 29 Owner

Can one of the admins verify this patch?

ckaestne added some commits on Jan 29

- introduce option for call graph in addition to CFG (no implementation… 946dd42

Labels
None yet

Milestone
No milestone

Assignee
No one assigned

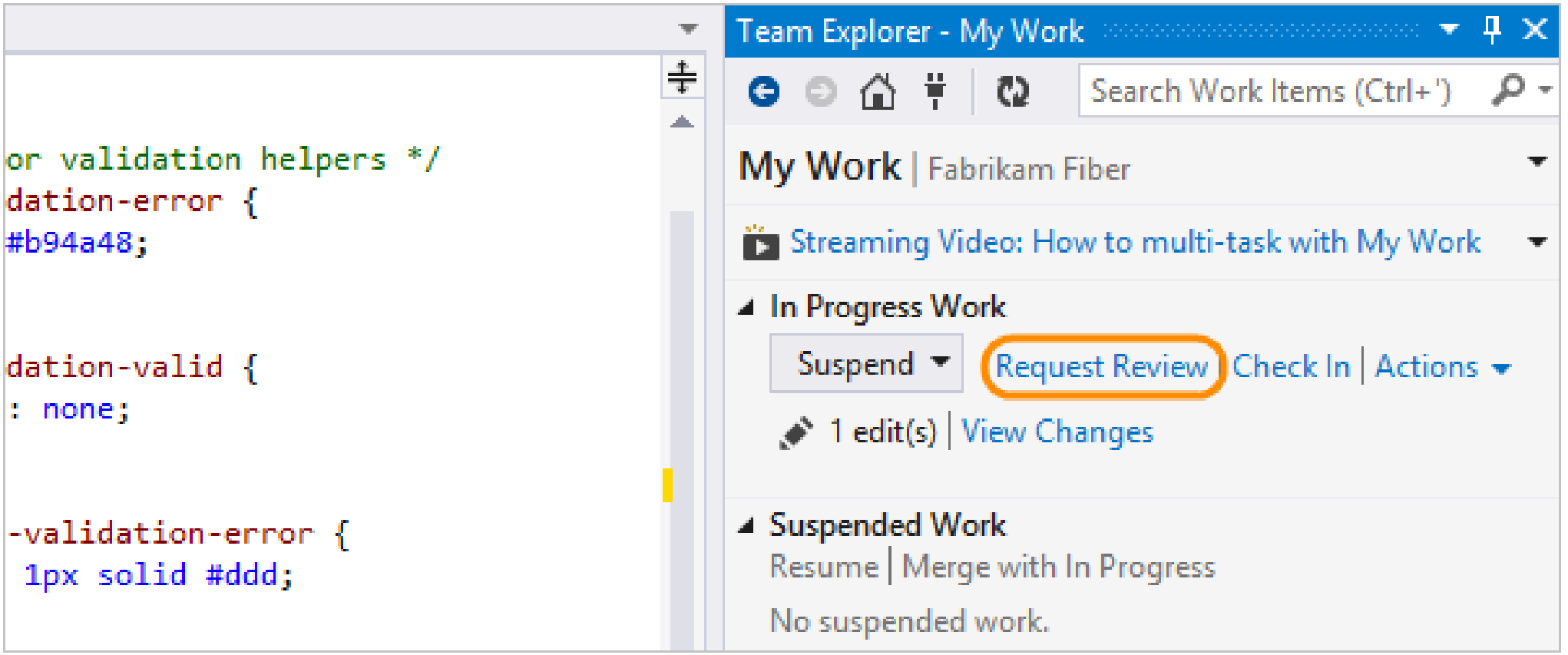
2 participants

Microsoft (Visual Studio, CodeFlow, etc.)

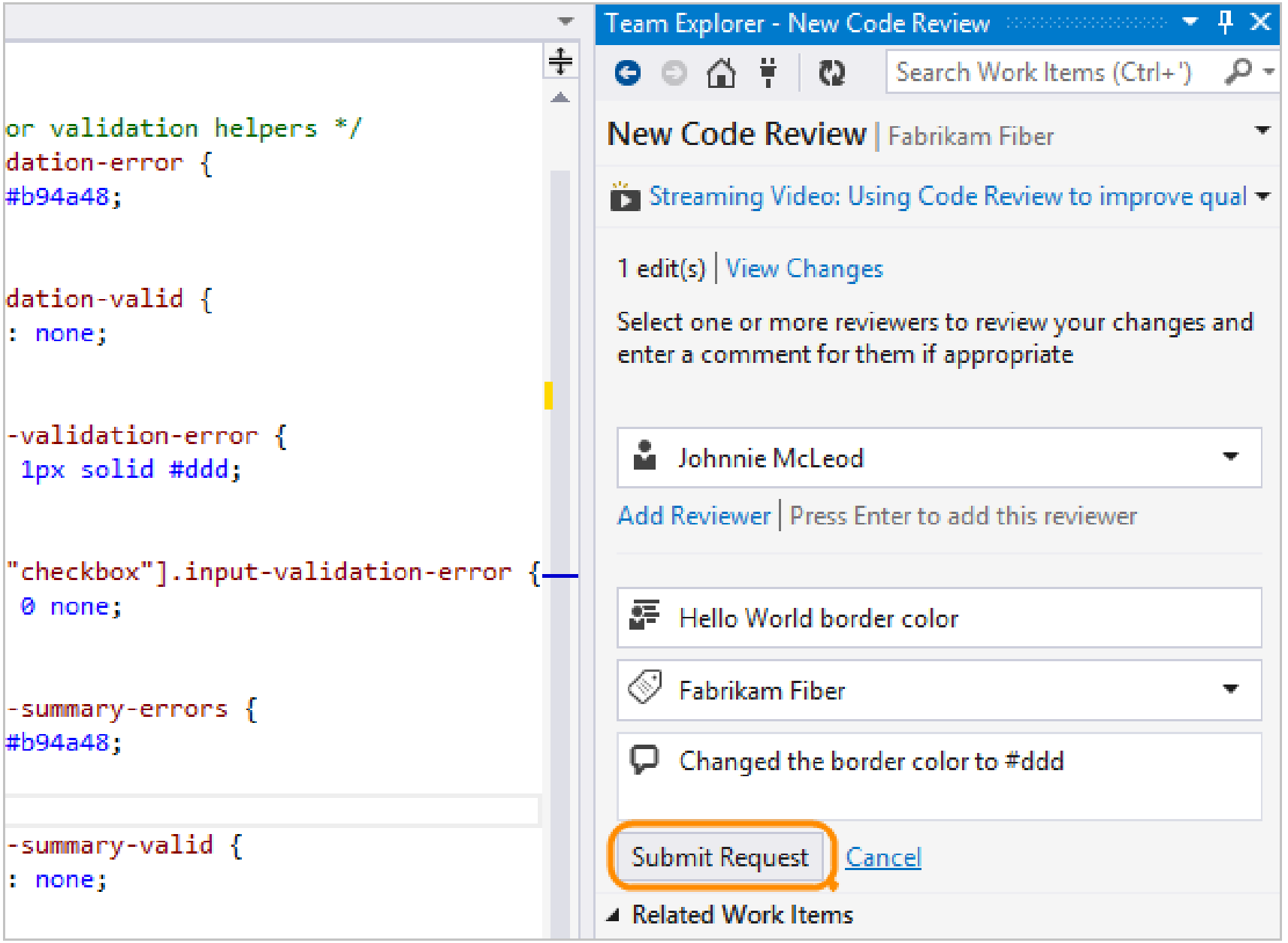
- Before you check in your code, you can use Visual Studio to ask someone else from your team to review it. Your request will show up in the Team Explorer, in the “My Work” page.

- (Are you using Git to share your code? If so, then use a pull request.)

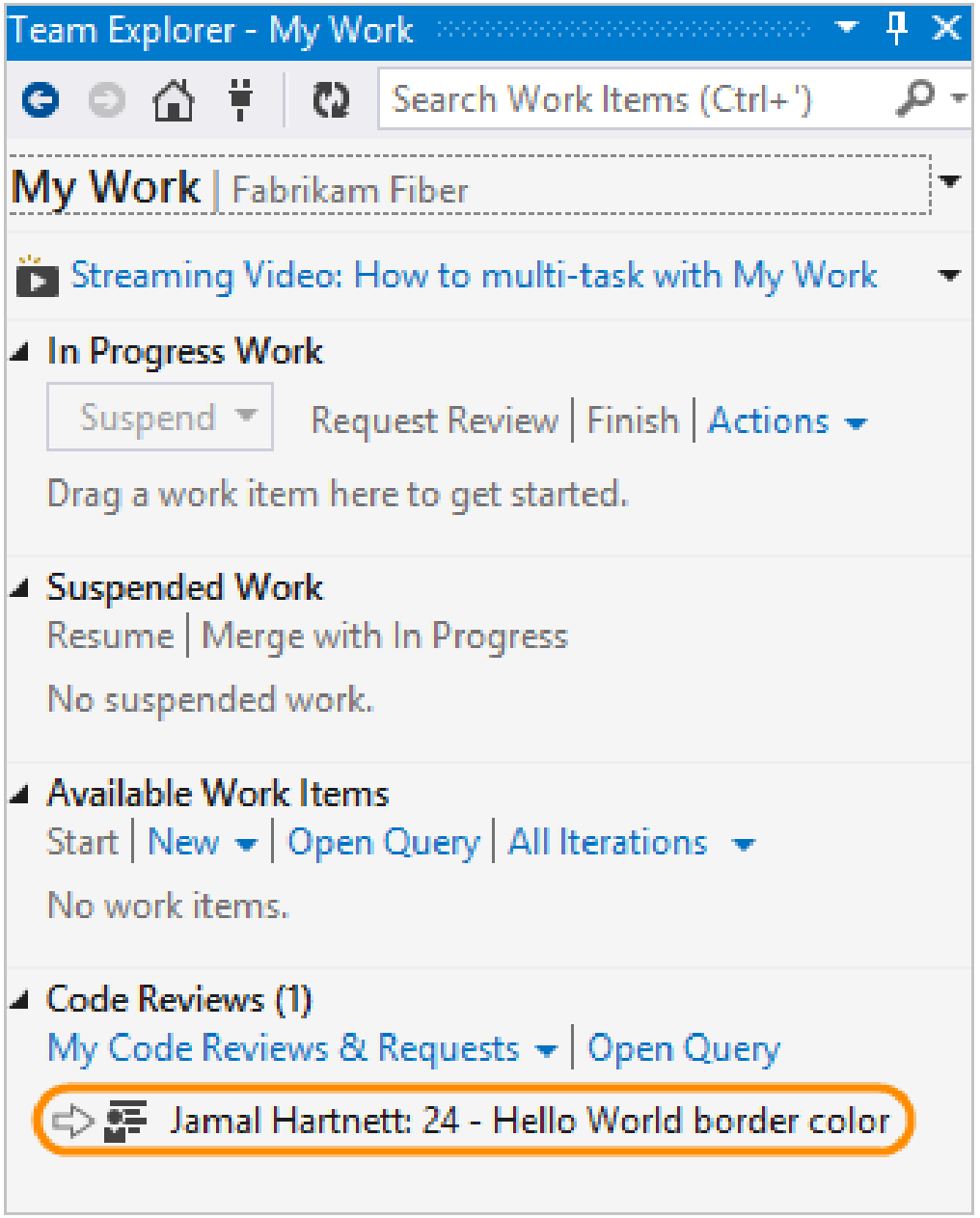
Dev #1 - Request Review

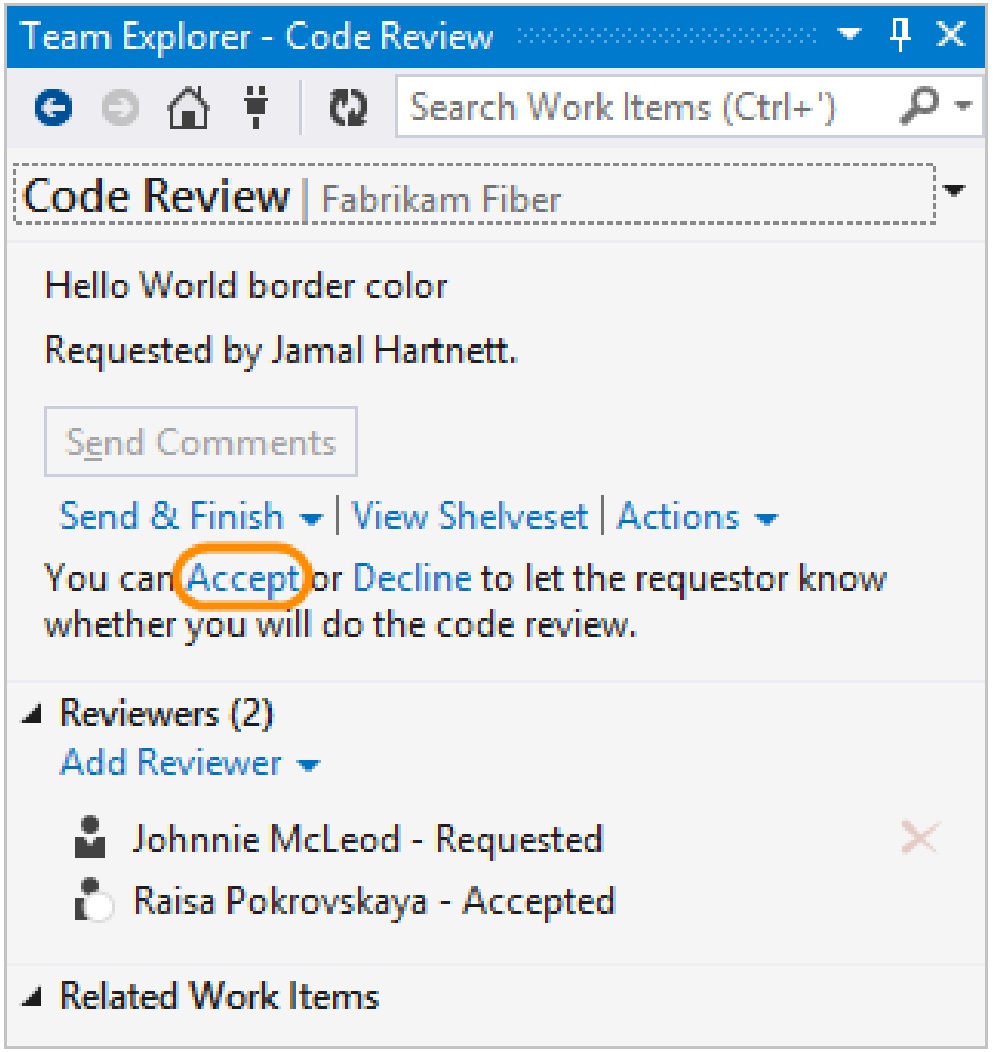


Dev #1 – Submit Request to Dev #2

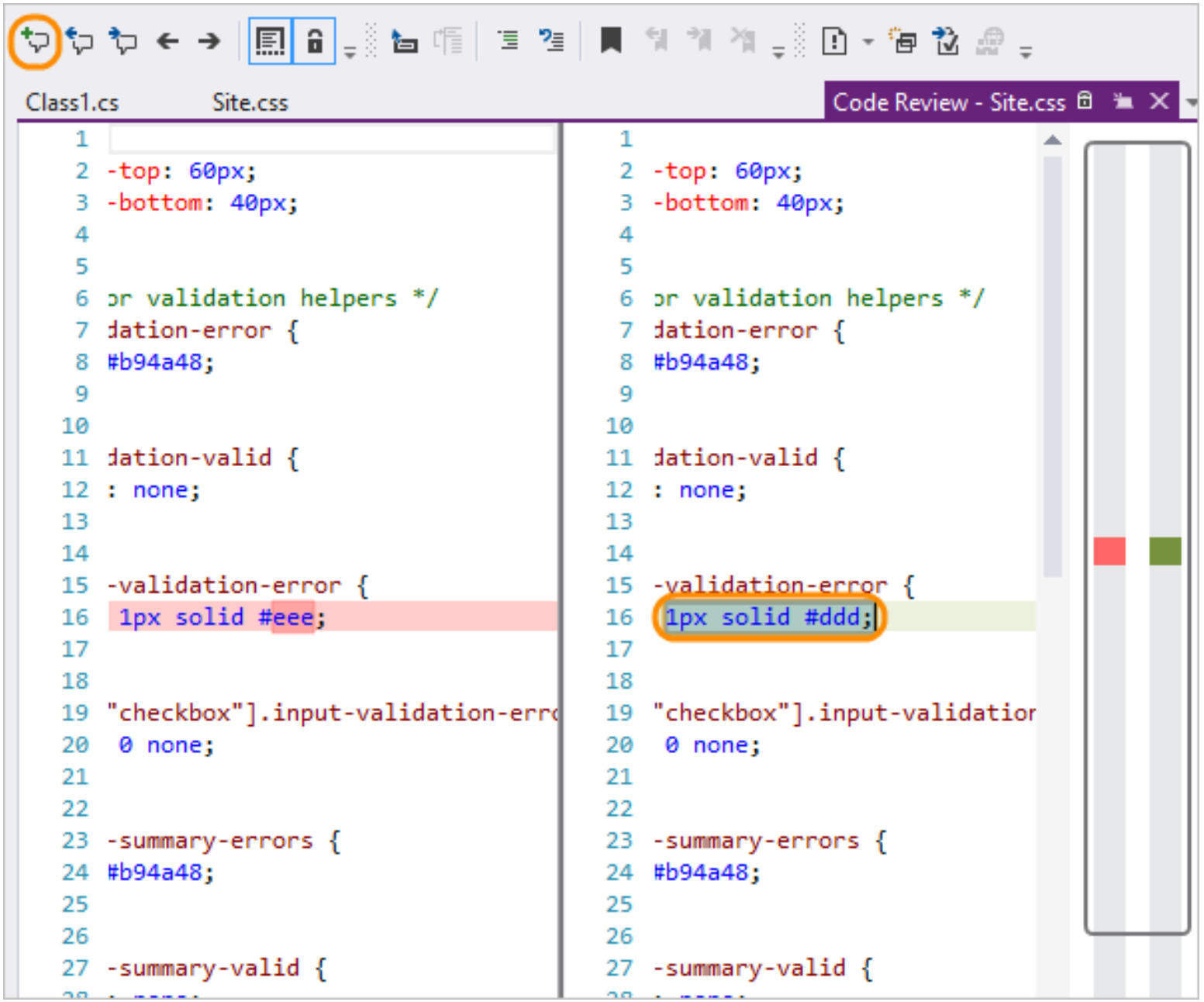


Dev #2 - See and Accept Request

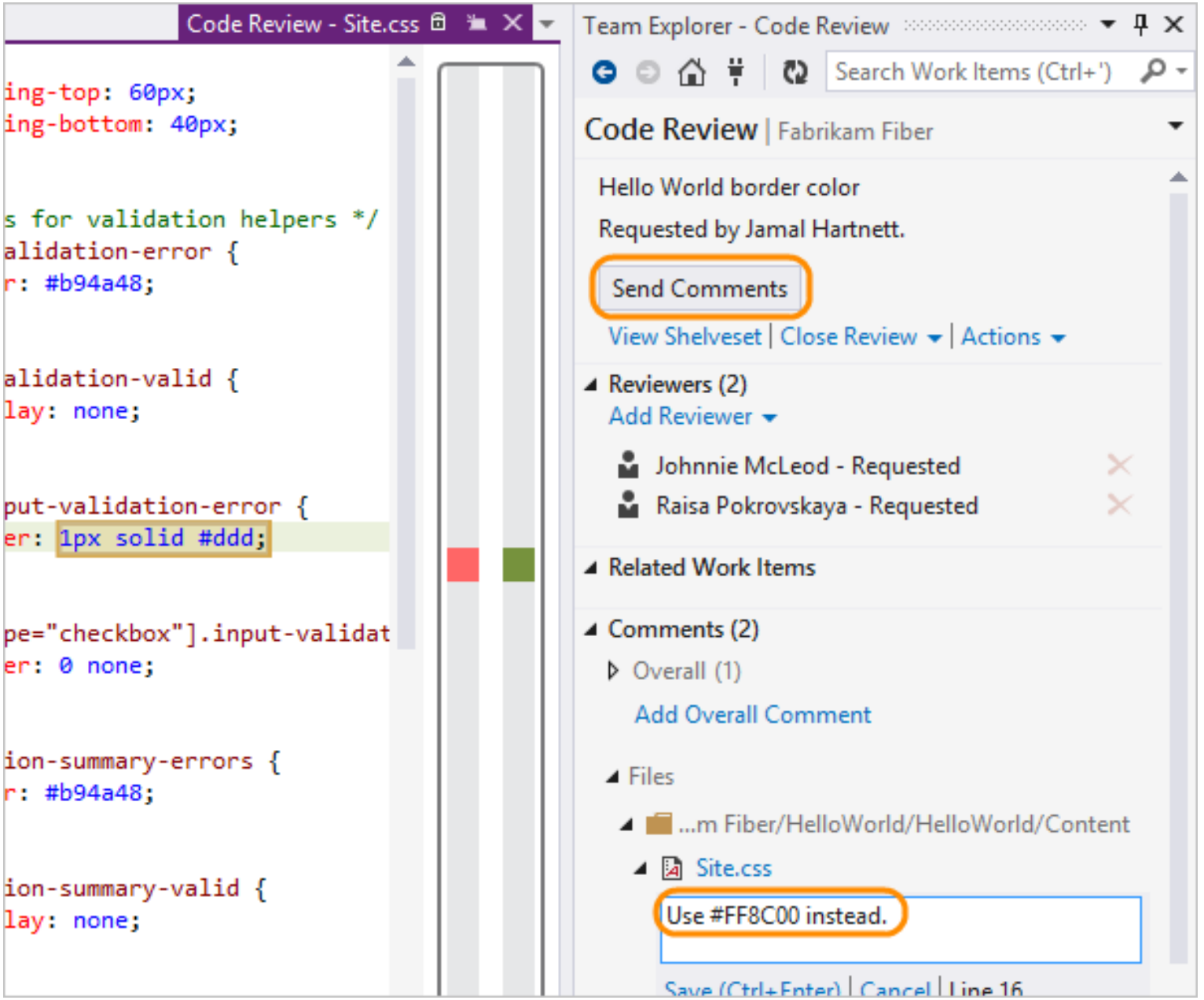




Dev #2 – View Details



Dev #2 – Suggest Improvements



Google's Code Review Policy

- All change lists (“CLs”) 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.
- 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 “To Be Reviewed”, 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.

Google, Facebook

- “In broad strokes, code review processes in Google and Facebook are similar. In both companies it is practically required that every change to production code is reviewed by at least one team member.

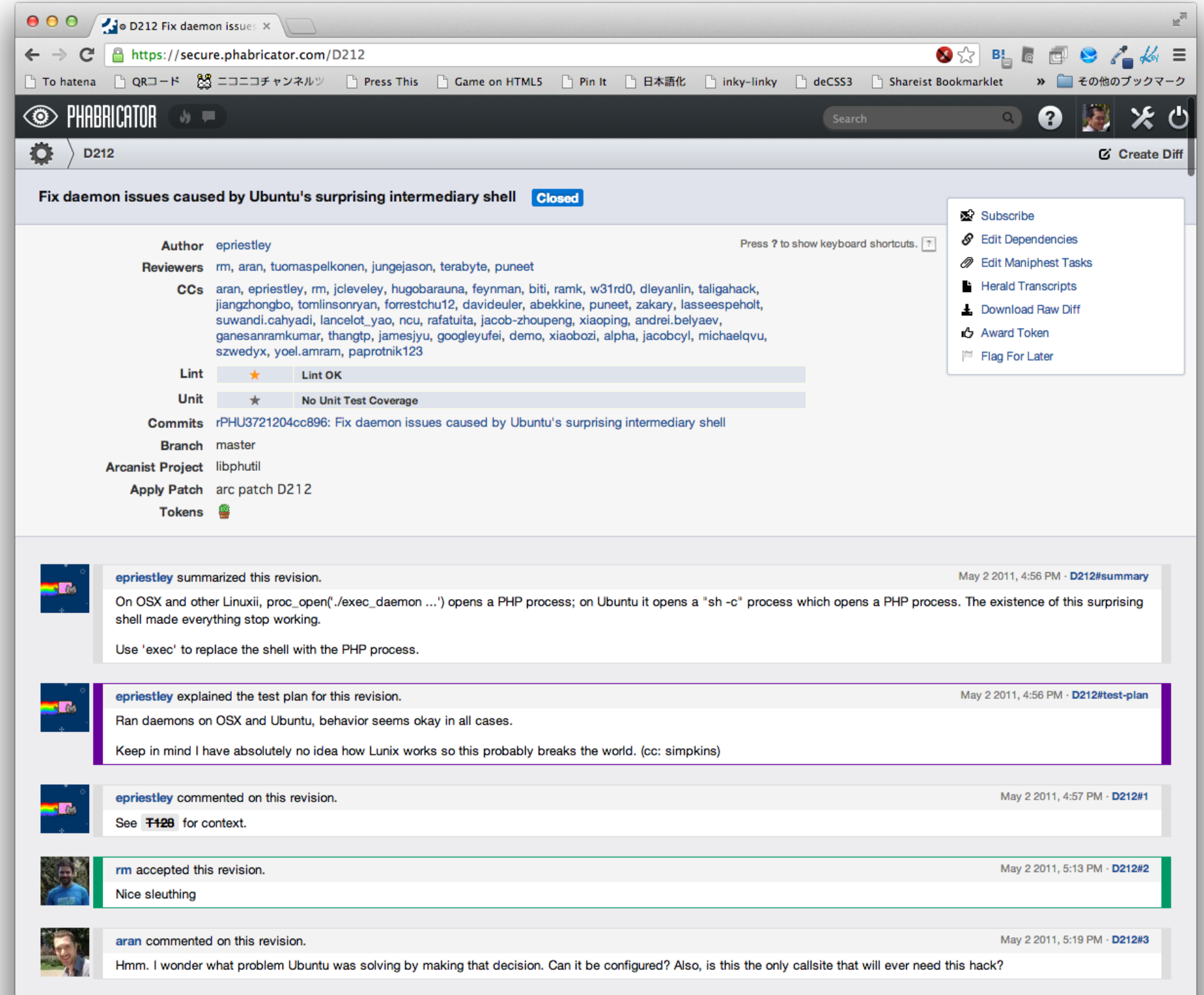
  Google has this readability process where you need to earn a privilege to commit in a given programming language. Readability is literally a badge on your profile that the code review system checks to see if you can commit the code yourself or you need to ask for an extra review for the compliance with company-wide language style guides.”

  - Marcin Wyszynski 2017, worked at both companies

Tools

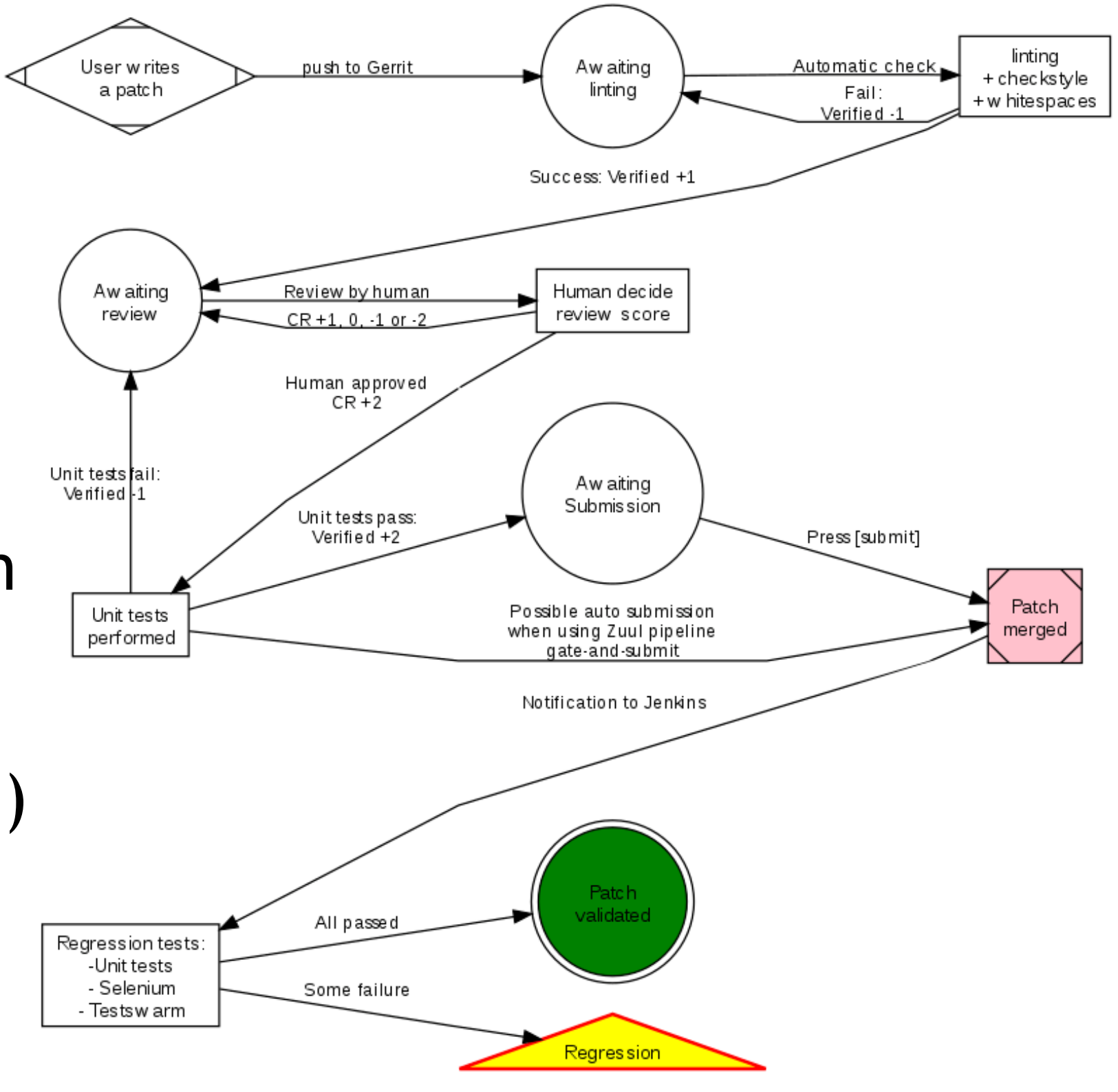
- Google uses Mondrian, an in-house tool
  - One of its authors later made https://www.gerritcodereview.com/
  - Reportedly, one of its authors later made https://reviewable.io/
  - Those give a taste of what Mondrian is like
- Facebook uses Phabricator
  - Developed in-house, later open-sourced

    https://www.phacility.com/


D212 Fix daemon issues
https://secure.phabricator.com/D212
PHABRICATOR
Search
D212
Create Diff
Fix daemon issues caused by Ubuntu's surprising intermediary shell
Closed
Author epriestley
Reviewers rm, aran, tuomaspelkonen, jungejason, terabyte, puneet
CCs aran, epriestley, rm, jcleveley, hugobarauna, feynman, biti, ramk, w31rd0, dleyanlin, taligahack, jiangzhongbo, tomlinsonryan, forrestchu12, davideuler, abekkine, puneet, zakary, lasseespeholt, suwandi.cahyadi, lancelot_yao, ncu, rafatuita, jacob-zhoupeng, xiaoping, andrei.belyaev, ganesanramkumar, thangtp, jamesjyu, googleyufei, demo, xiaobozi, alpha, jacobcyl, michaelqvu, szwedyx, yoel.amram, paprotnik123
Lint Lint OK
Unit No Unit Test Coverage
Commits rPHU3721204cc896: Fix daemon issues caused by Ubuntu's surprising intermediary shell
Branch master
Arcanist Project libphutil
Apply Patch arc patch D212
Tokens
Press ? to show keyboard shortcuts.
Subscribe
Edit Dependencies
Edit Maniphest Tasks
Herald Transcripts
Download Raw Diff
Award Token
Flag For Later
epriestley summarized this revision.
May 2 2011, 4:56 PM · D212#summary
On OSX and other Linuxii, proc_open('./exec_daemon ...') opens a PHP process; on Ubuntu it opens a "sh -c" process which opens a PHP process. The existence of this surprising shell made everything stop working.
Use 'exec' to replace the shell with the PHP process.
epriestley explained the test plan for this revision.
May 2 2011, 4:56 PM · D212#test-plan
Ran daemons on OSX and Ubuntu, behavior seems okay in all cases.
Keep in mind I have absolutely no idea how Lunix works so this probably breaks the world. (cc: simpkins)
epriestley commented on this revision.
May 2 2011, 4:57 PM · D212#1
See T128 for context.
rm accepted this revision.
May 2 2011, 5:13 PM · D212#2
Nice sleuthing
aran commented on this revision.
May 2 2011, 5:19 PM · D212#3
Hmm. I wonder what problem Ubuntu was solving by making that decision. Can it be configured? Also, is this the only callsite that will ever need this hack?

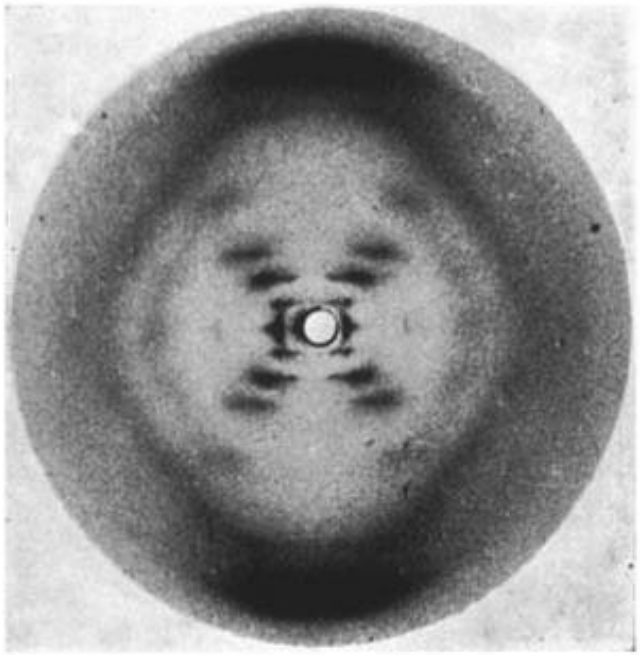
Code Review Integration Example

(MediaWiki)



Trivia: Chemistry, Biology

- This English chemist and X-ray crystallographer used X-ray diffraction images of DNA, leading to the discovery of its double helix structure (see "*Photo 51*" below). After dying at age 37 of cancer, other collaborators on the work were awarded the Nobel prize (controversy: not awarded posthumously).



Psychology: Group Decision Making

- 156 students read descriptions of three hypothetical candidates for student body president and then met in 4-person groups to elect a winner
  - Each candidate had 16 associated pieces of information (*unambiguously* positive, negative and neutral facts related to the job)
  - Collectively, each 4-person group had *all* the info
  - Individually, each person only had *some* info
  - Candidate A is *objectively twice as good* as B or C
    - Who wins the election?

- Starting individual information distribution breakdown by group condition:

Number of Items of Information About Each Candidate Received by Group Members Before Discussion

Condition and information valence	Candidate A	Candidate B	Candidate C
Shared			
Positive	8	4	4
Neutral	4	8	8
Negative	4	4	4
Unshared/consensus			
Positive	2	4	1
Neutral	4	5	8
Negative	4	1	1
Unshared/conflict			
Positive	2	4 [4]	4 [4]
Neutral	4	6 [4]	4 [6]
Negative	4	0 [2]	2 [0]

Note. In the unshared/conflict condition, 2 members of a 4-person group received configurations of information about Candidates B and C given by the numbers without brackets, whereas the other 2 members received configurations given by the numbers in brackets.

- Results

Condition	Candidate A	Candidate B	Candidate C	*n*
Shared info.	.67	.17	.17	72

- Results

Condition	Candidate A	Candidate B	Candidate C	*n*
Shared info.	.67	.17	.17	72
Unshared info./ consensus	.25	.61	.14	84
Unshared info./ conflict	.21	.46	.33	72

- Results

Condition	Candidate A	Candidate B	Candidate C	*n*
Pregroup preferences				
Shared info.	.67	.17	.17	72
Unshared info./ consensus	.25	.61	.14	84
Unshared info./ conflict	.21	.46	.33	72
Postgroup preferences				
Shared info.	.85	.11	.04	72

- Results

Condition	Candidate A	B	C	*n*
Pregroup preferences				
Shared info.	.67	.17	.17	72
Unshared info./ consensus	.25	.61	.14	84
Unshared info./ conflict	.21	.46	.33	72
Postgroup preferences				
Shared info.	.85	.11	.04	72
Unshared info./ consensus	.20	.75	.05	83[a]
Unshared info./ conflict	.17	.47	.36	72
Group decisions				
Shared info.	.83	.11	.06	18
Unshared info./ consensus	.24	.71	.05	21
Unshared info./ conflict	.12	.53	.35	17[b]

Group Decision Making

- “Even though groups could have produced unbiased composites of the candidates through discussion, they decided in favor of the candidate initially preferred by a plurality rather than the most favorable candidate. Group members' pre and postdicussion recall of candidate attributes indicated that discussion tended to perpetuate, not correct, members' distorted pictures of the candidates.”

Group Decision Making

[G. Stasser, W. Titus. *Pooling of Unshared Information in Group Decision Making: Biased Information Sampling During Discussion*. J. of Personality and Social Psychology, 48(6) 1985.]

- Implications for SE: Both “formal code inspection” and “modern multiperson passaround code review” are group decision making tasks. Reviewers/inspectors are unlikely to start with uniformly perfect information are are thus vulnerable to this bias.

Do Code Reviews Work?

Expectations, Outcomes, and Challenges Of Modern Code Review

Alberto Bacchelli
REVEAL @ Faculty of Informatics
University of Lugano, Switzerland
alberto.bacchelli@usi.ch

Christian Bird
Microsoft Research
Redmond, Washington, USA
cbird@microsoft.com

***Abstract*—Code review is a common software engineering practice employed both in open source and industrial contexts. Review today is less formal and more "lightweight" than the code inspections performed and studied in the 70s and 80s. We empirically explore the motivations, challenges, and outcomes of tool-based code reviews. We observed, interviewed, and surveyed developers and managers and manually classified hundreds of review comments across diverse teams at Microsoft. Our study reveals that while finding defects remains the main motivation for review, reviews are less about defects than expected and instead provide additional benefits such as knowledge transfer, increased team awareness, and creation of alternative solutions to problems. Moreover, we find that code and change**

when to use code review and how it should fit into their development process. Researchers can focus their attention on practitioners' challenges to make code review more effective.

We present an in-depth study of practices in teams that use modern code review, revealing what practitioners think, do, and achieve when it comes to modern code review.

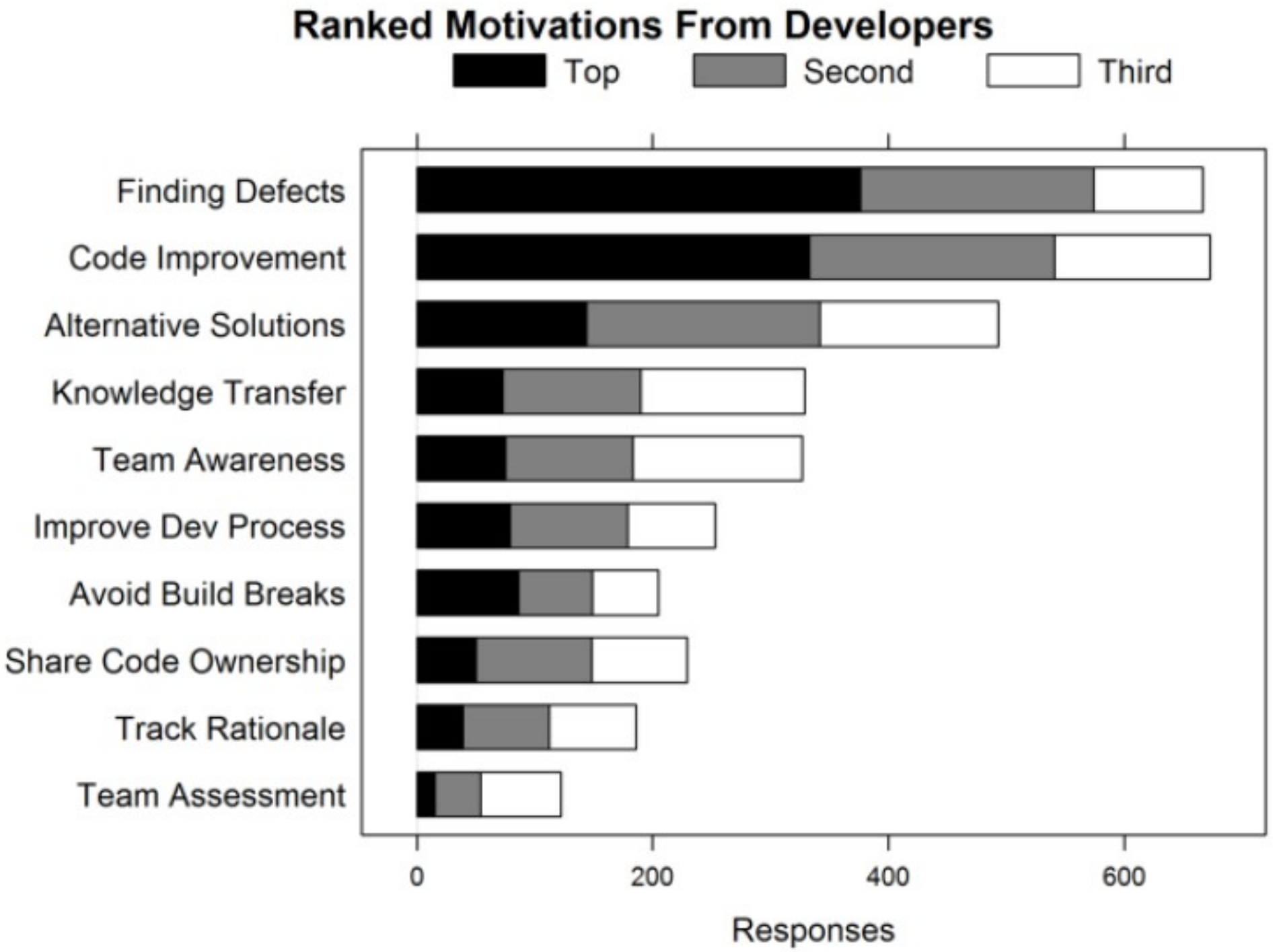
Since Microsoft is made up of many different teams working on very diverse products, it gives the opportunity to study teams performing code review *in situ* and understand their expectations, the benefits they derive from code review, the needs they have, and the problems they face.

Code Review Goals

- **Finding defects**
  - both low-level and high-level issues (requirements/design/code)
- **Code improvement**
  - readability, formatting, commenting, consistency, dead code removal, naming, 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

Code Review Goals (cont'd)

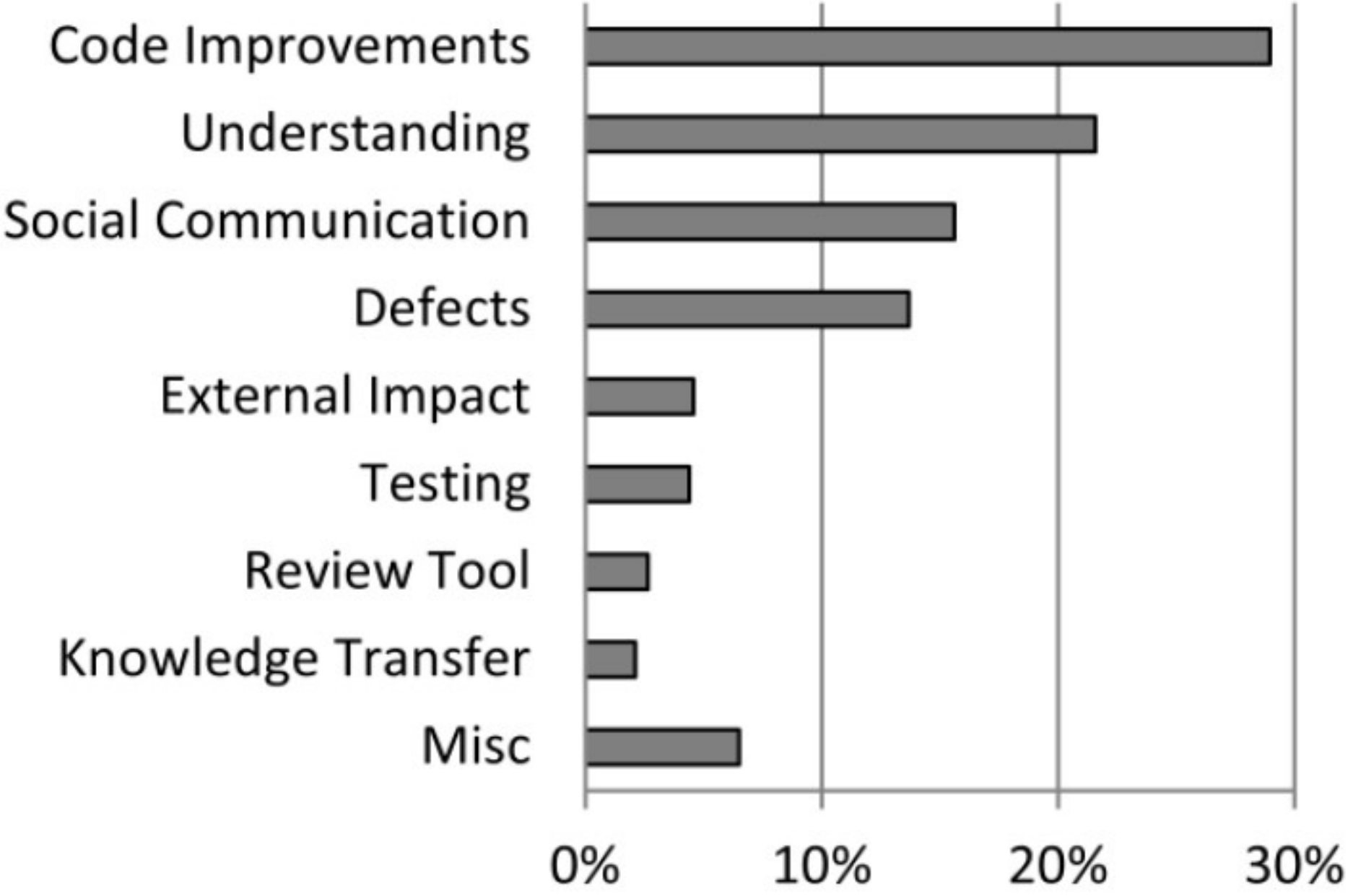
- Team awareness and transparency
  - let others "double check" changes
  - announce changes to specific developers or entire team ("FYI")
- Shared code ownership
  - openness toward critique and changes
  - makes developers "less protective" of their code


Ranked Motivations From Developers
Top
Second
Third
Finding Defects
Code Improvement
Alternative Solutions
Knowledge Transfer
Team Awareness
Improve Dev Process
Avoid Build Breaks
Share Code Ownership
Track Rationale
Team Assessment
0
200
400
600
Responses

Outcomes
(200 Microsoft reviews, 570 comments)

- Most frequent: **code improvements** (29%)
  - 58 better coding practices
  - 55 removing unused/dead code
  - 52 improving readability
- Moderate: **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
- Rare: **knowledge transfer**
  - 12 pointers to internal/external documentation etc

Outcomes


Code Improvements
Understanding
Social Communication
Defects
External Impact
Testing
Review Tool
Knowledge Transfer
Misc
0%
10%
20%
30%

Aside: Philosophy

- One definition of the source of **unhappiness** is unrealized desires
  - You are unhappy when you desire reality (or your experience) to have property *X* but it does not
  - Buddhism: “craving is the cause of all suffering”
- You can either change what you want
- … or try to change reality / your experiences
- Both are usually very difficult!

Expectation/Outcome Mismatch

- Low quality of code reviews
  - Reviewers look for easy errors (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

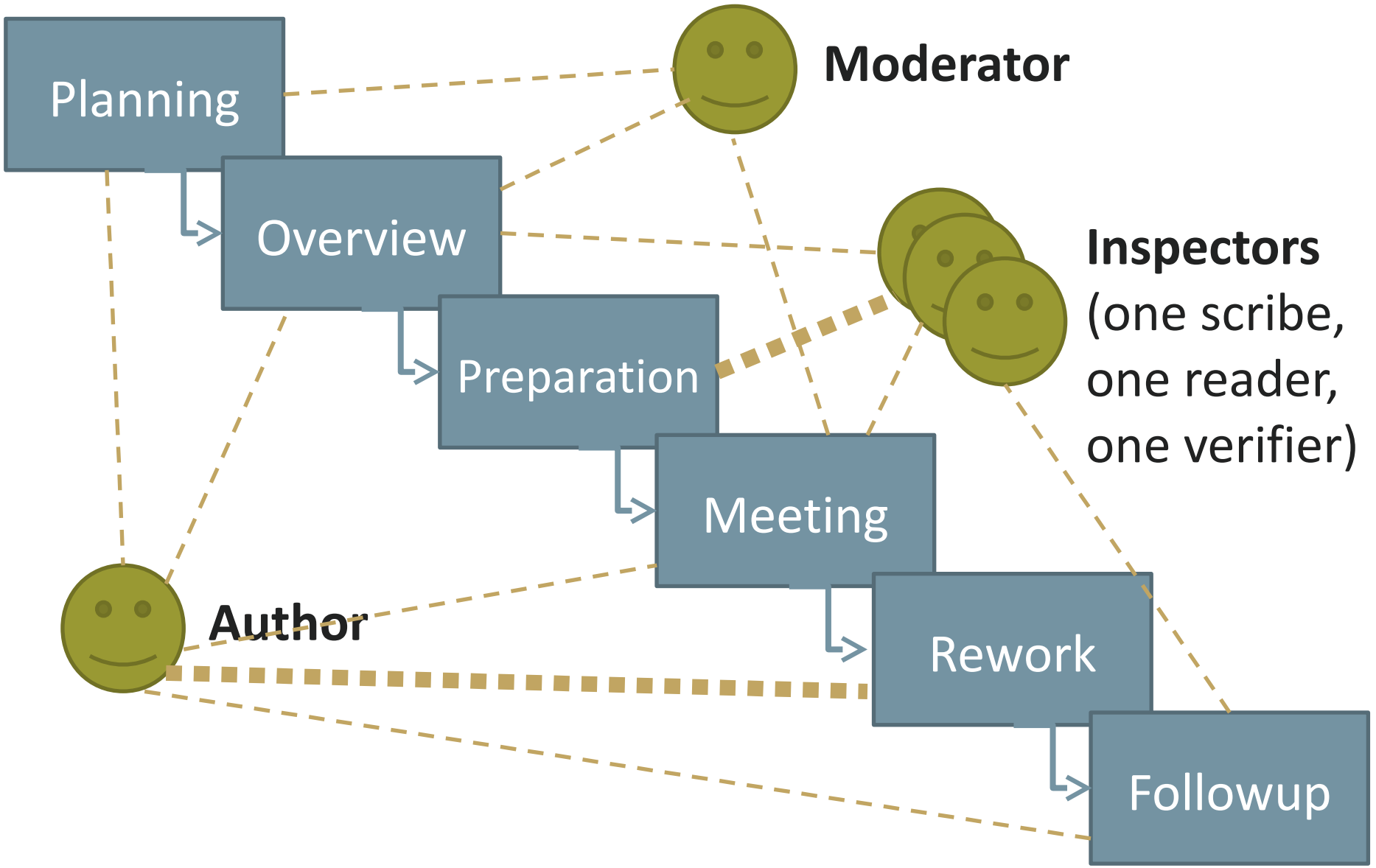
Formal Code Inspections

- In a **formal code inspection** a group of developers meets to review code or other artifacts
  - Popularized by IBM in the 1970s, broadly adopted in the 1980s, subject of much research
- Viewed as the most effective approach to finding bugs
  - 60-90% of bugs were found with inspections
- Very expensive and labor-intensive

Inspection Team and Roles

- Typically 4-5 people (at least 3 if “formal”)
  - 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

Inspection Process



Inspection Steps

- Planning (select Moderator)
- Overview (brief) – Author presents context in meeting
- **Preparation** (1-2h) – Every reviewer inspects the code separately
- **Meeting** (1h)
  - Reader presents the code
  - All reviewers identify issues
  - Meetings only discover issues, do not discuss solution or whether it really is an issue
- Rework
- Followup (Verifier checks changes)

Inspection Checklists

- Reminder of what to look for
- 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?

Process Details

- **Authors do not explain or defend** the code – not objective
  - Author != moderator, != scribe, !=reader
  - Author observes questions and misunderstandings and clarifies 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

Social Issues: Egos in Inspections

- Authors *should* separate self-worth from code
- Identify defects, not alternatives; do not criticize authors
  - "you didn't initialize variable *x*" → "I don't see where variable *x* is initialized"
- Avoid defending code. Avoid discussions of solutions or alternatives
- Reviewers should not "show off" as smarter
- Author decides how to resolve defects

Social Issues: Inspection Incentives

- Meetings should not include management
- Do not use code reviews for HR evaluations!
  - "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"
- cf. lecture on Metrics and Incentives

Root Cause Analysis

- An overarching goal is look beyond the immediate puzzle
- Identify way to improve the development process to avoid this problem in the future
  - Restructure the development process
  - Introduce new policies
  - Use new development tools, languages, analyses, etc.
- cf. “definition of insanity”

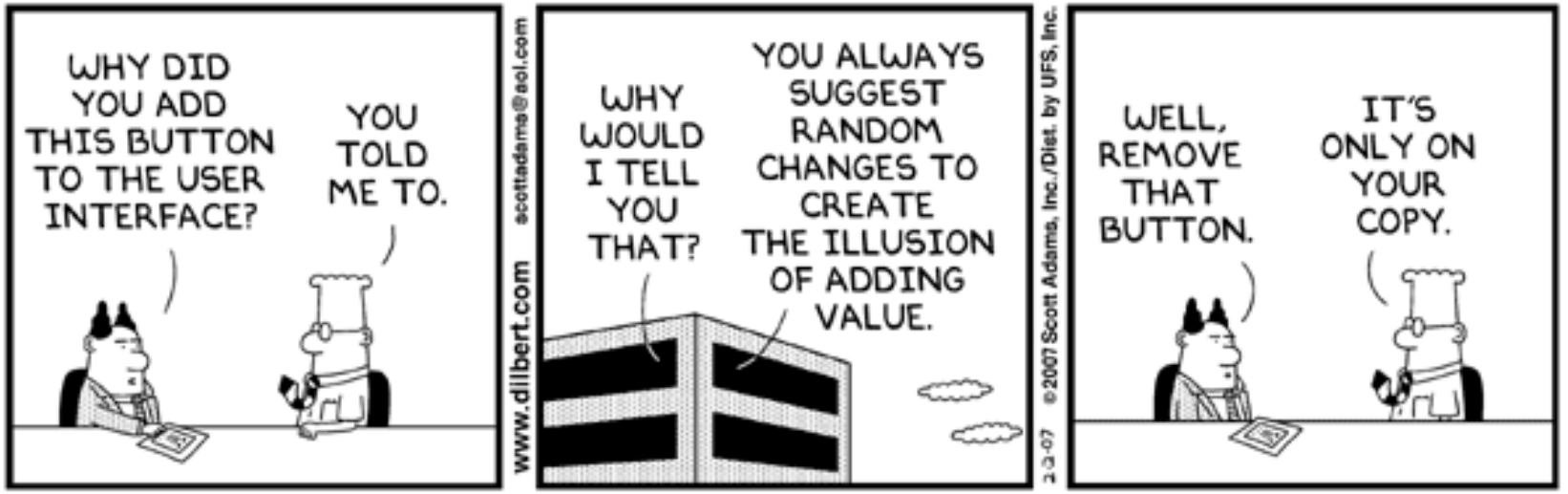
When to Inspect

- Inspect before milestones
- Incremental inspections during development
  - Earlier often better than later: smaller fragments, chance to influence further development
  - Large code bases can be expensive and frustrating to review
    - Break down, divide and conquer
    - Focus on critical components
    - Identify defect density in first sessions to guide further need of inspections

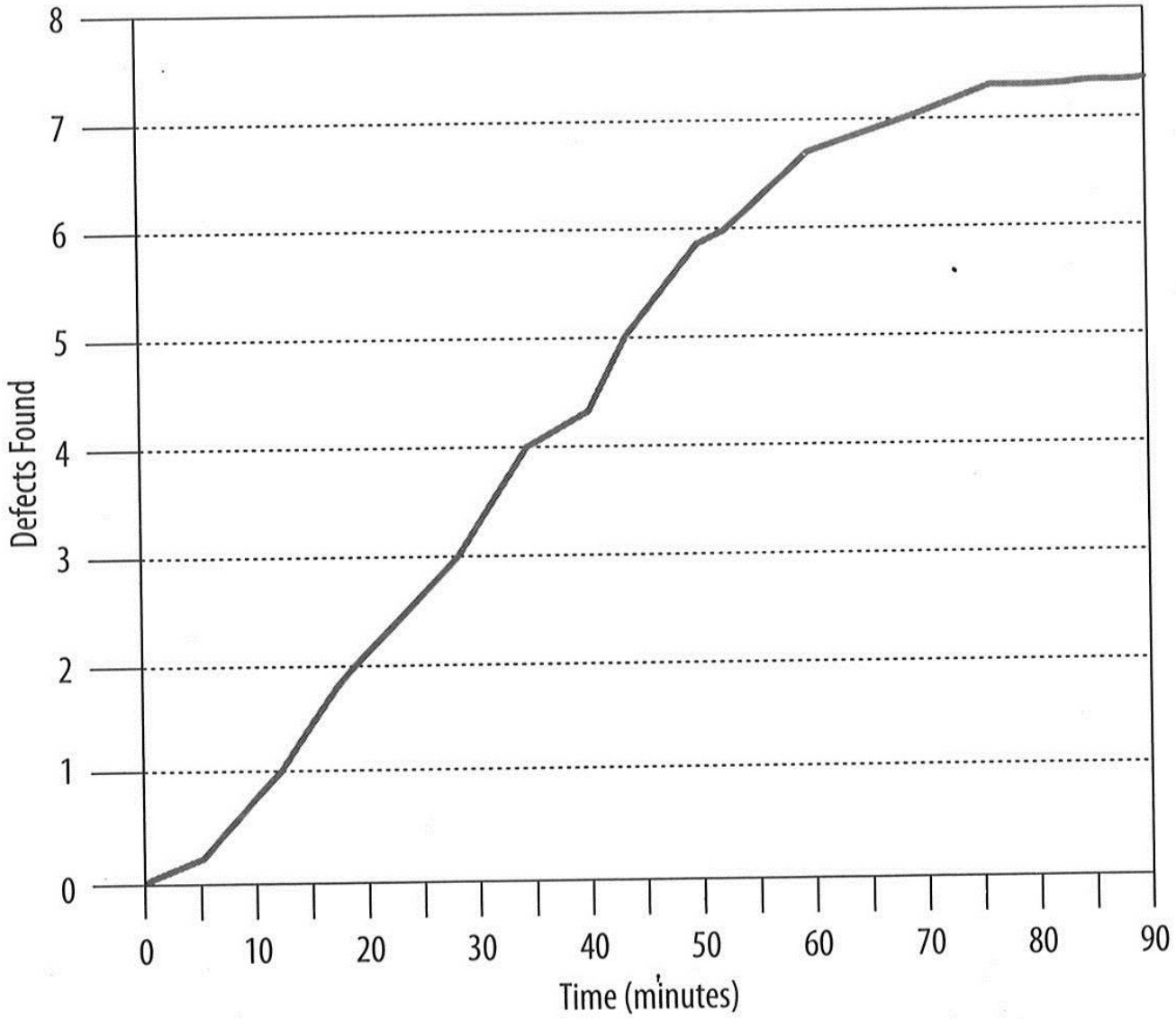
Guidelines for Inspections

- Collected over many companies in many projects and experiments
- Several metrics are easily measurable
  - Effort, issues found, lines of code inspected, etc.

[Oram and Wilson (ed.). Making Software. O'Reilly 2010. Chapter 18 and papers reviewed therein.]

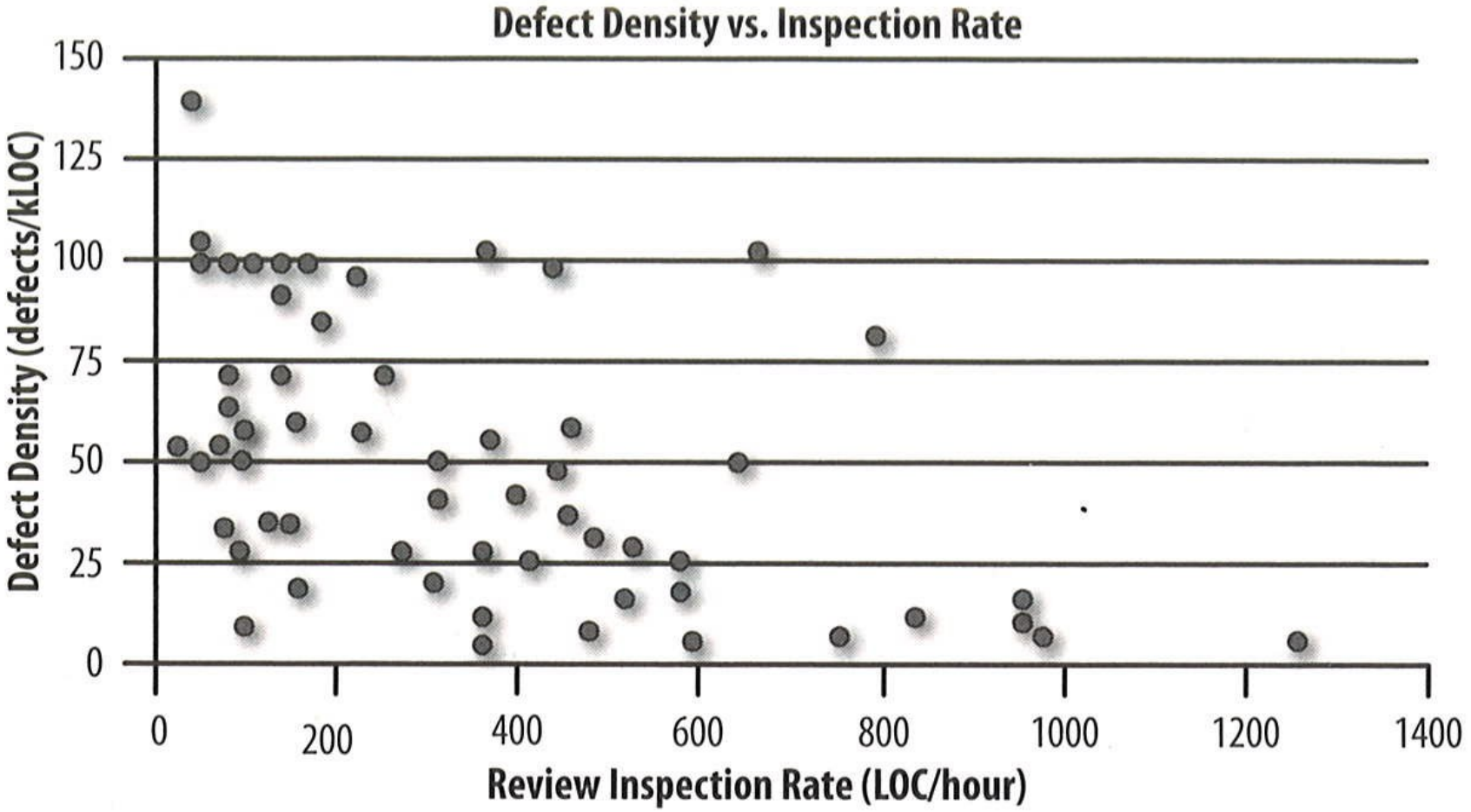


Focus Fatigue



Recommendation:
Do not exceed
60 minute session

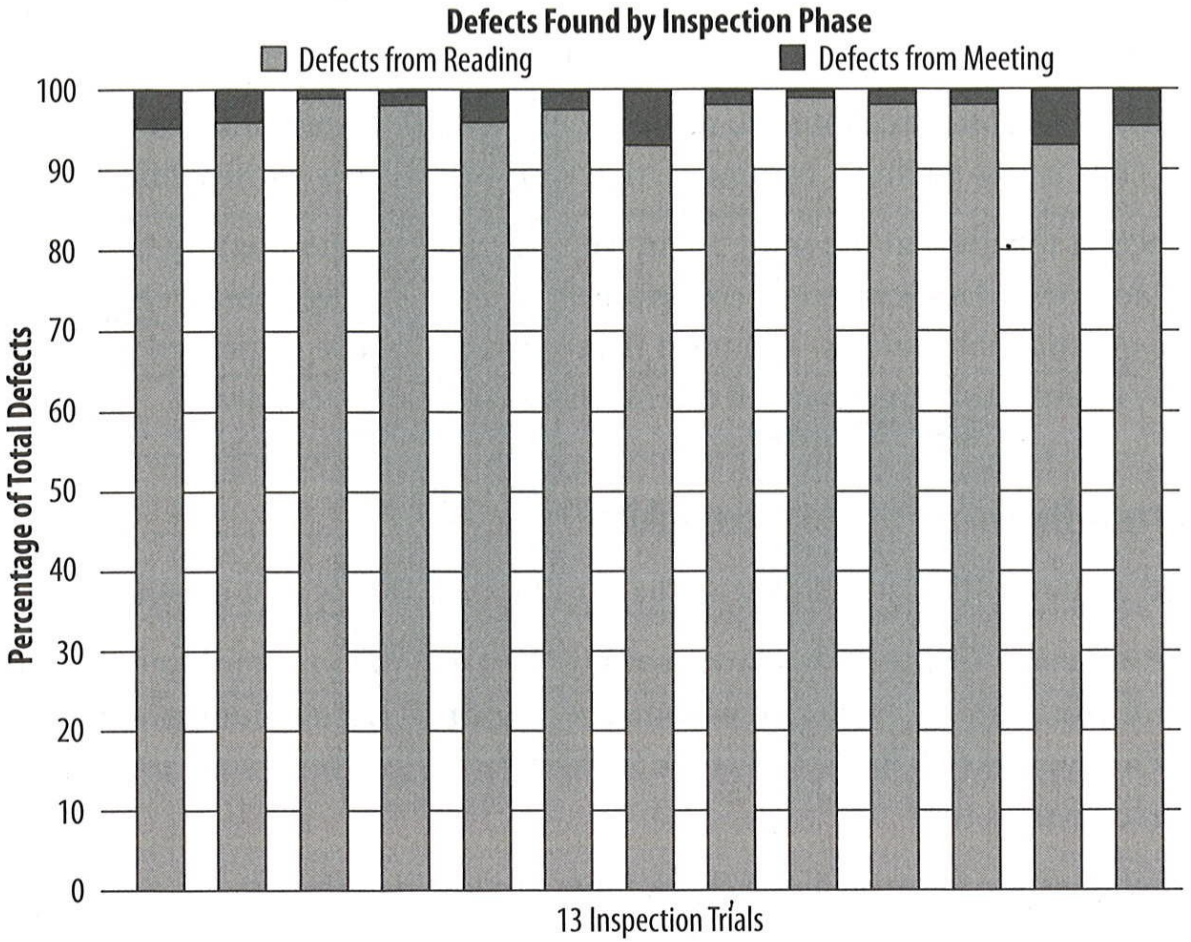
Inspection Speed



Above 400 LOC/h reviews get shallow

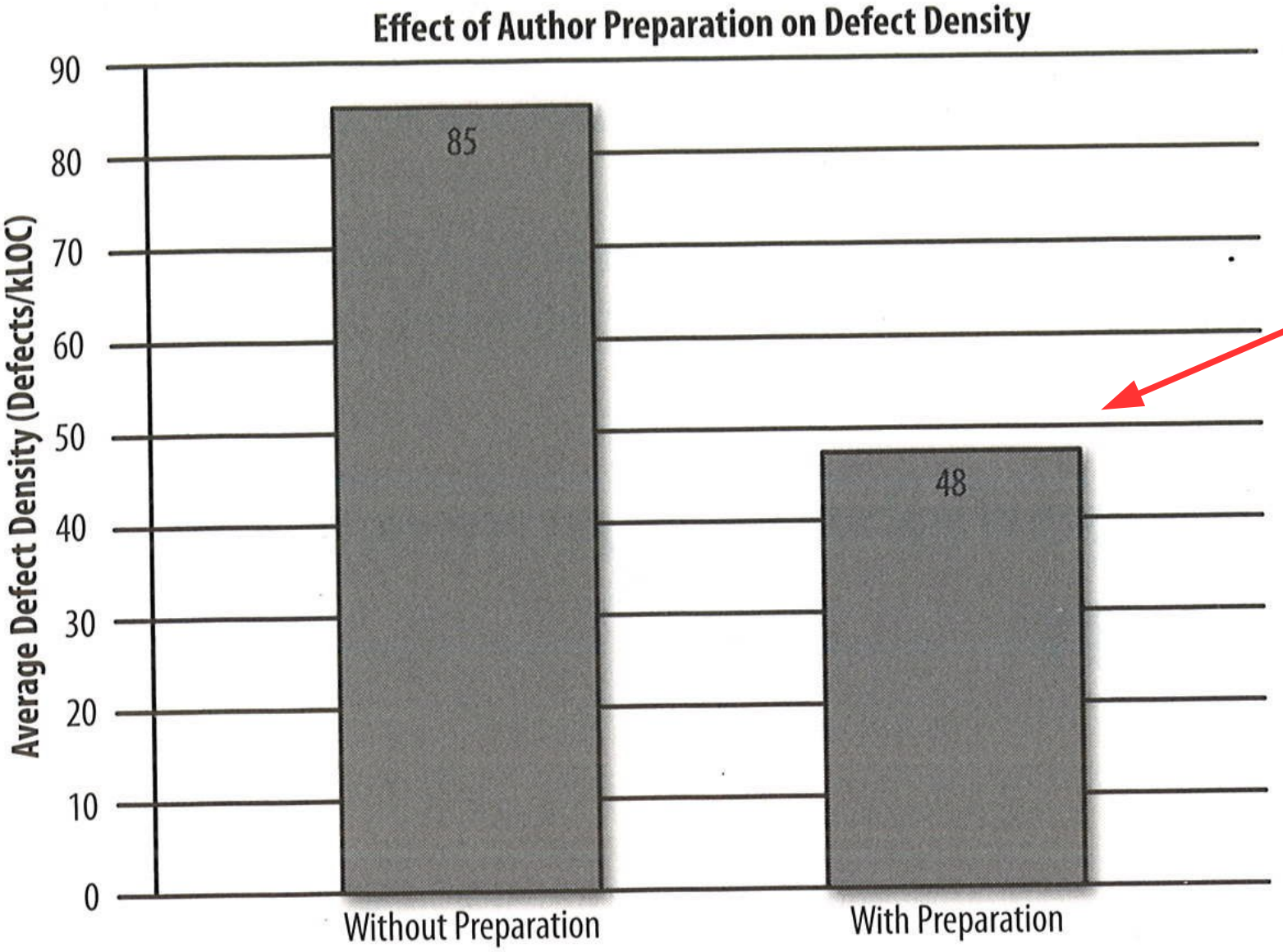
Recommendation: Schedule fewer than 400 LOC for a 1h review session

Inspection Meeting Efficacy



Most issues found during preparation, not in meeting
Suggested synergy seems to have only low impact
Claim: Defects found in meetings often more subtle

Self-Checks Matter



Authors have self-checked documents before inspection

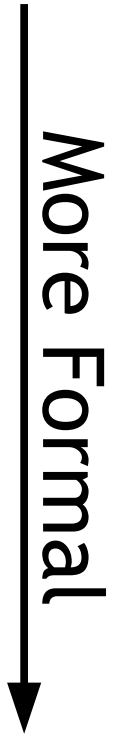
Inspection Accuracy

- About 25% of found issues are false positives
  - We'll return to this issue later in the course: it turns out humans are not perfect …
- Avoid discussing during meeting
- Confusion during meeting is an indicator that document could be clearer
  - For maintainability, if someone says “I don't think the code does X”, it does not actually matter if the code does X or not!

Inspections vs. Reviews: Costs

- Formal inspections and modern code reviews
  - Formal inspections very expensive (about one developer-day per session)
  - Passaround review is distributed, asynchronous
- Code reviews vs. testing
  - Code reviews claimed more cost effective
- Code reviews vs. not finding the bug

Code Review by Formality



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

(When should you use which type?)

Review Type and Differences

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 (modern code review)	No	Yes	Rarely	Yes	No
Ad Hoc Review	No	No	Yes	Yes	No

Studies, Claims, Results

- Raytheon review study
  - Reduced “rework” from 41% of costs to 20%
  - Reduced integration effort by 80%
- Paulk et al. – costs to fix a space shuttle software
  - $1 if found in inspection
  - $13 during system test
  - $92 after delivery
- IBM – 1h of inspection saves 20h of testing
- R. Grady – efficiency data from HP
  - System use 0.21 defects/h
  - Black box testing 0.28 defects/h
  - White box testing 0.32 defects/h
  - Reading/inspection 1.06 defects/h

Questions?

- Homework 1b, 1c, 1d all due!