Code Reviews

Claire Le Goues

November 5, 2019

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.
- Mitigate social and cultural issues in code review.
- Contrast motivations for and benefits of commit review at modern tech companies.
- Explain the motivations behind Ebay's use of static analysis in development, and several of the key factors that allowed the program to be successful.

FIRST: LET'S TALK ABOUT THE HOMEWORK

What are Code Reviews?

FORMAL INSPECTIONS

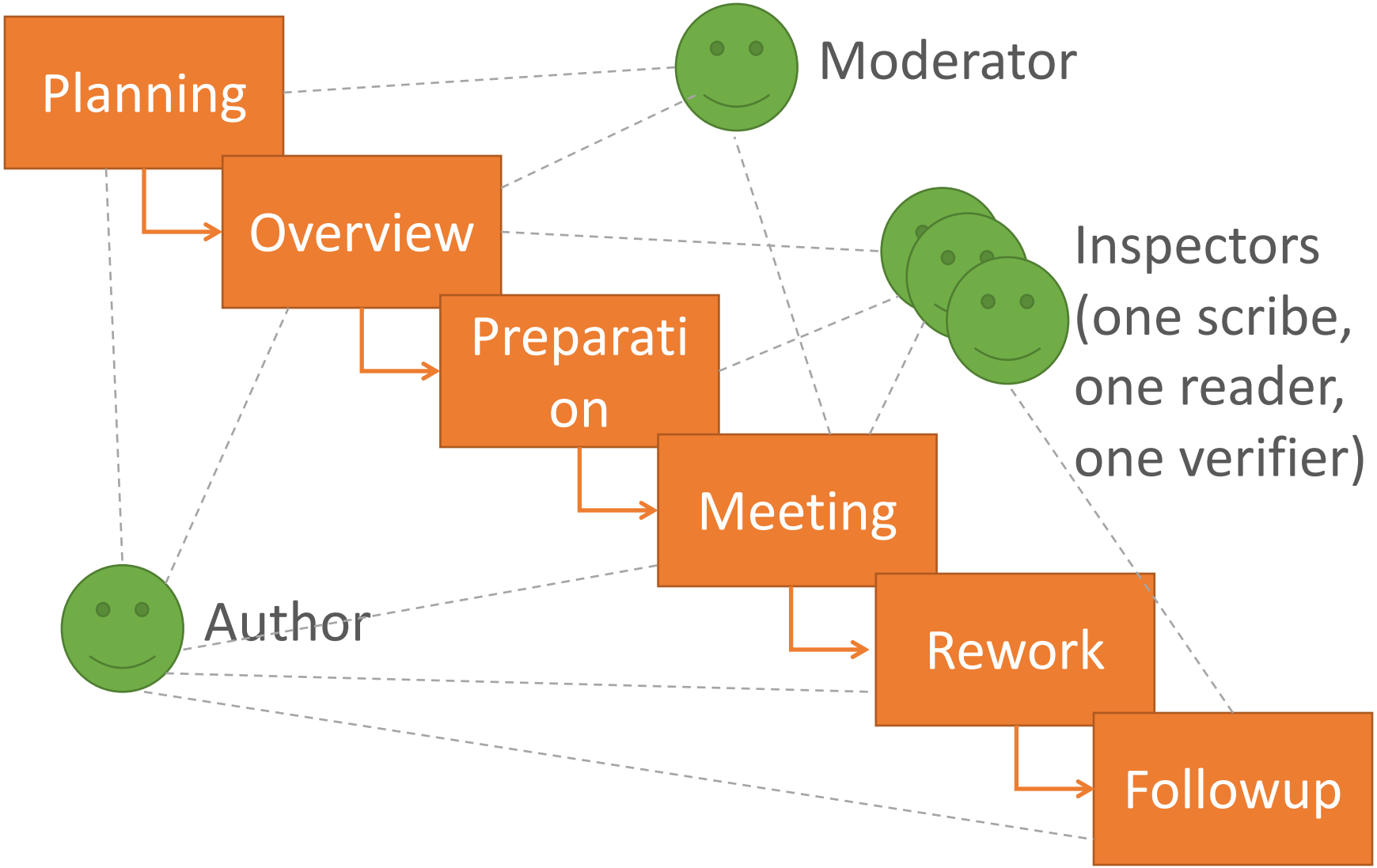
Formal Inspections

- Idea popularized in 70s at IBM
- Broadly adopted in 80s, much research
  - Sometimes replacing component testing
- 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
- Expensive and labor-intensive

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

Inspection Process

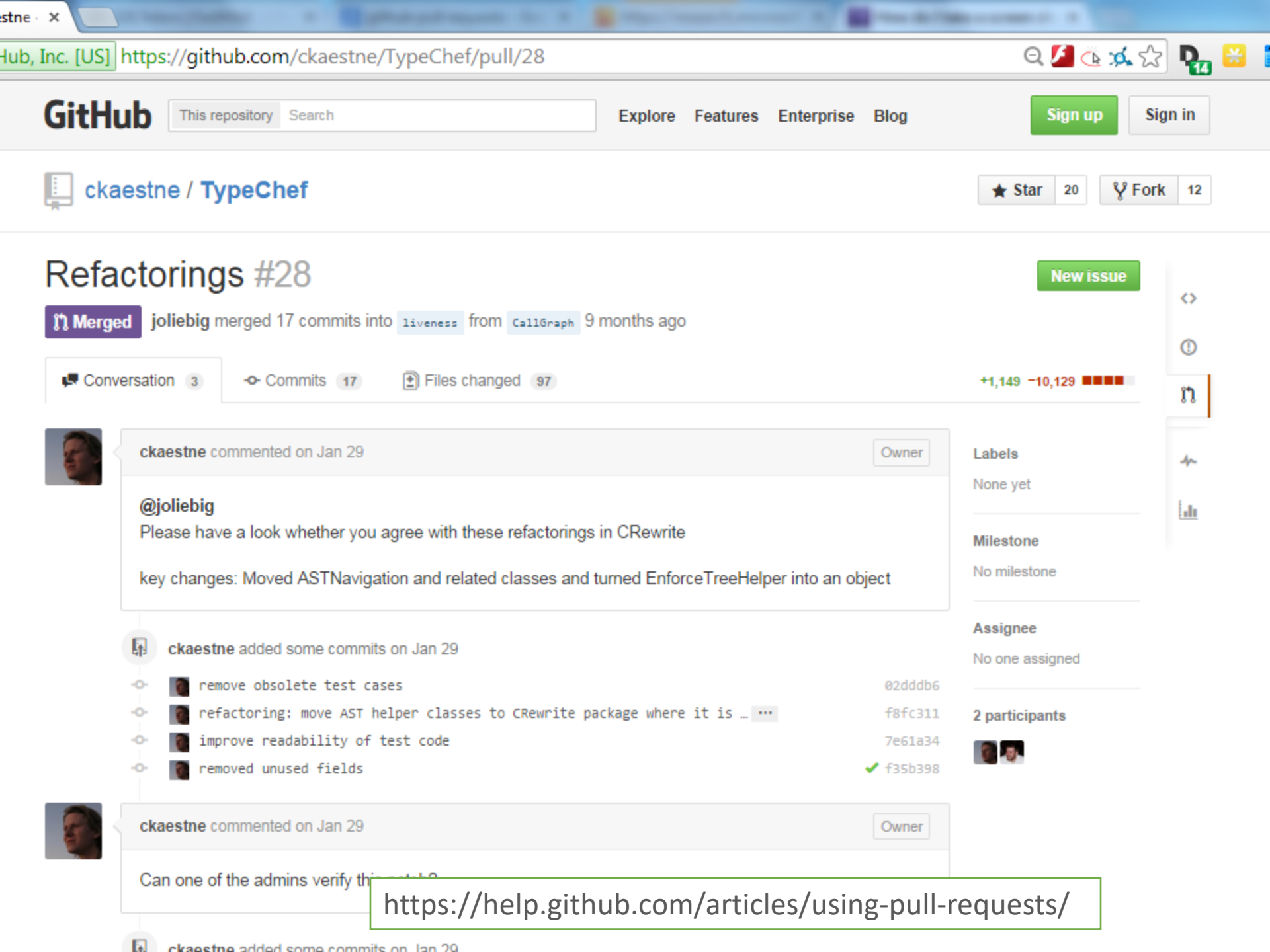


Checklists

- Reminder 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?
  - 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


https://github.com/ckaestne/TypeChef/pull/28
GitHub
This repository
Search
Explore
Features
Enterprise
Blog
Sign up
Sign in
ckaestne / TypeChef
Star 20
Fork 12
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?
Labels
None yet
Milestone
No milestone
Assignee
No one assigned
2 participants
https://help.github.com/articles/using-pull-requests/

"Many eyes make all bugs shallow"

Standard Refrain in Open Source

"Have peers, rather than customers, find defects"

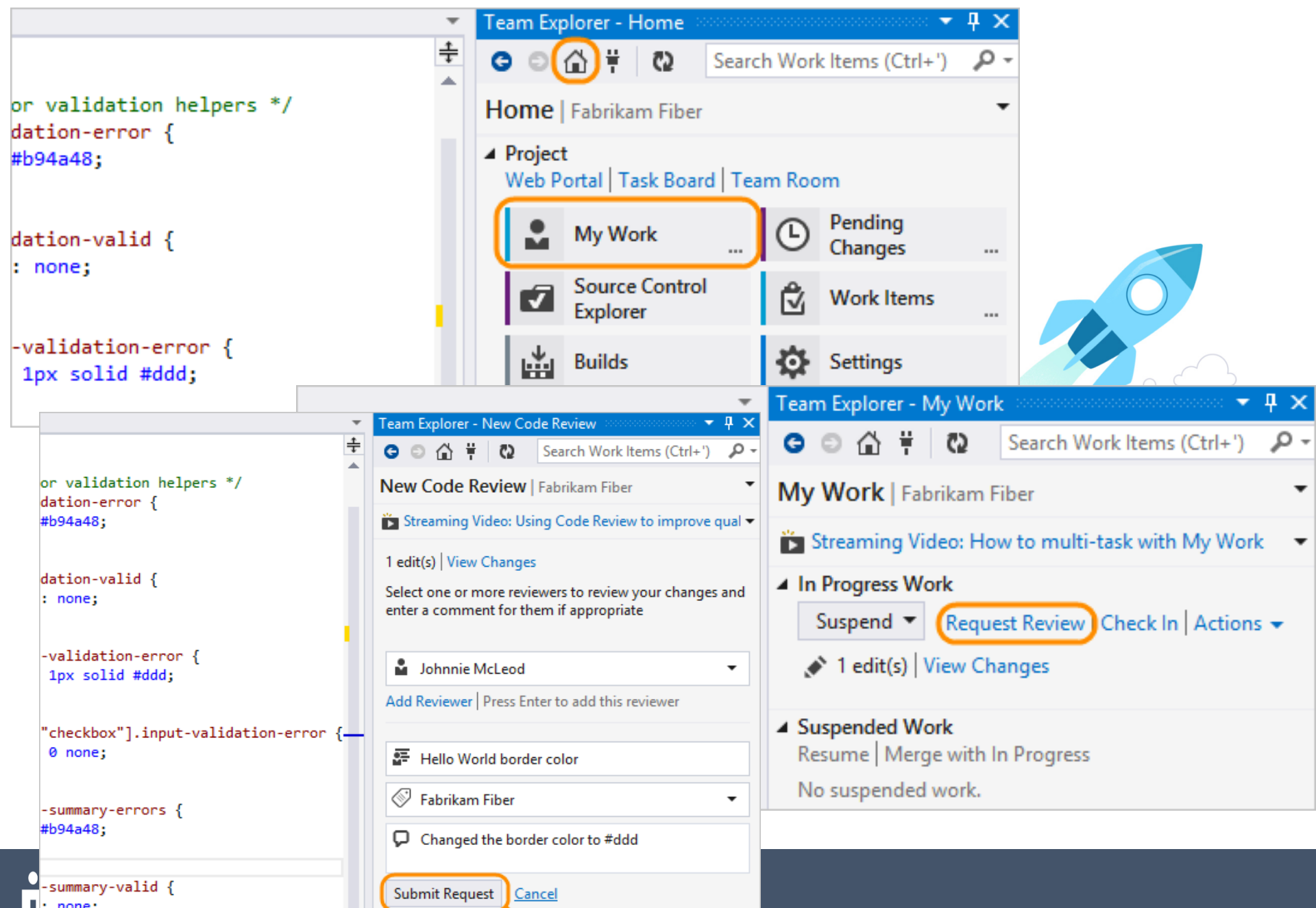
Karl Wiegers

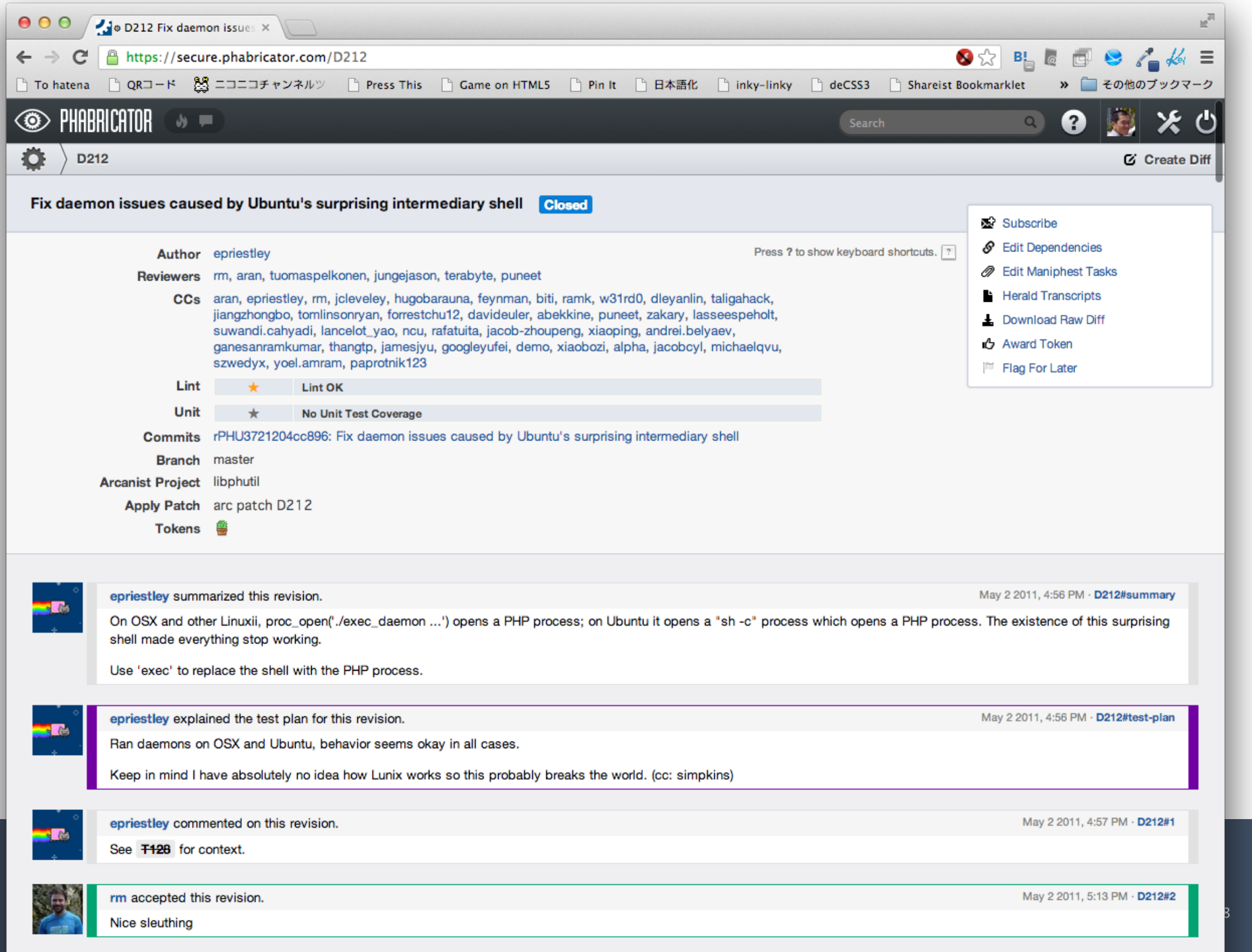
Isn’t testing sufficient?

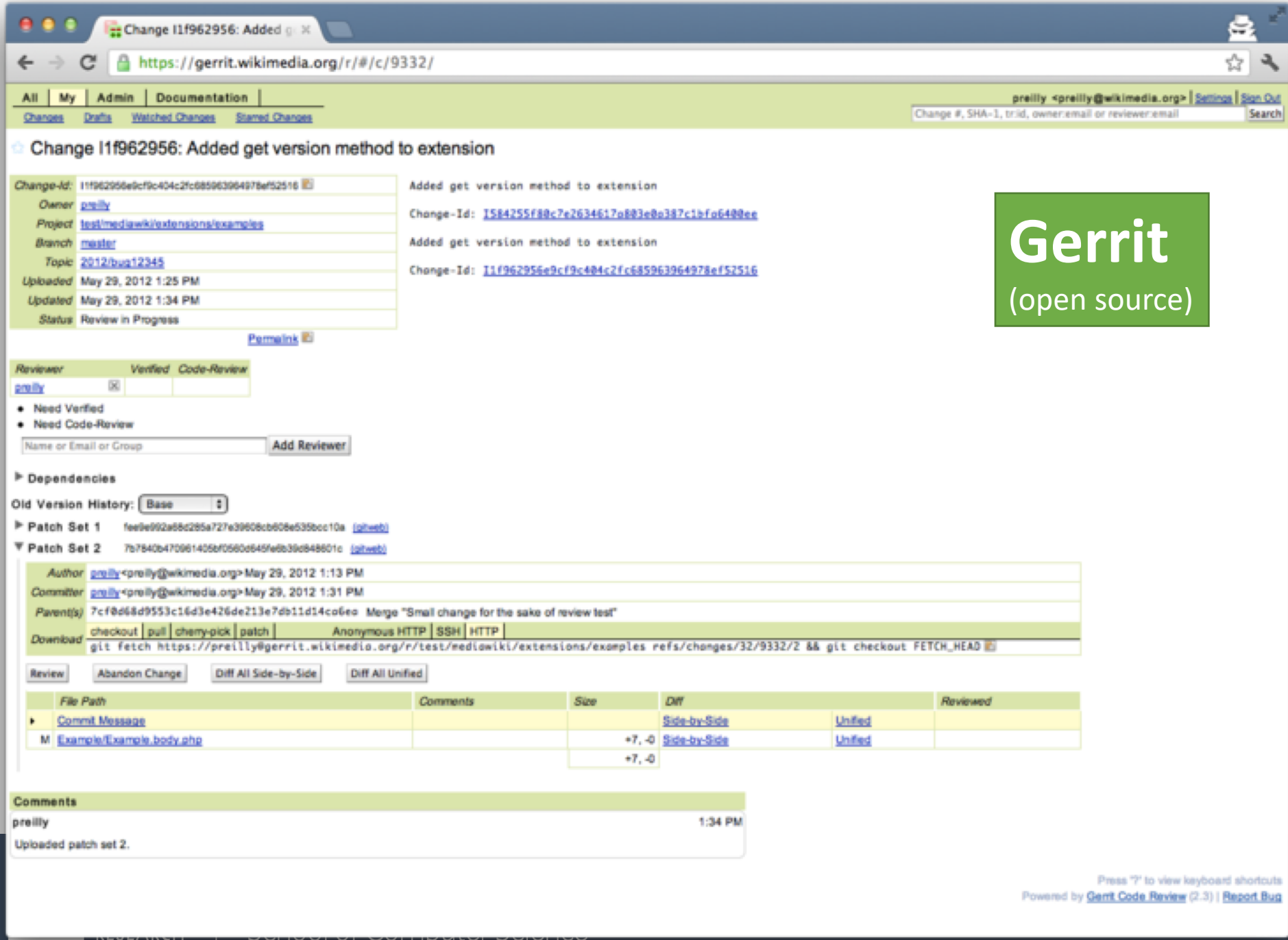
- Errors can mask other errors
- Only completed implementations can be tested (esp. scalability, performance)
- Design documents cannot be tested
- Tests don’t check code quality
- Many quality attributes (eg., security, compliance, scalability) are difficult to test

A second pair of eyes

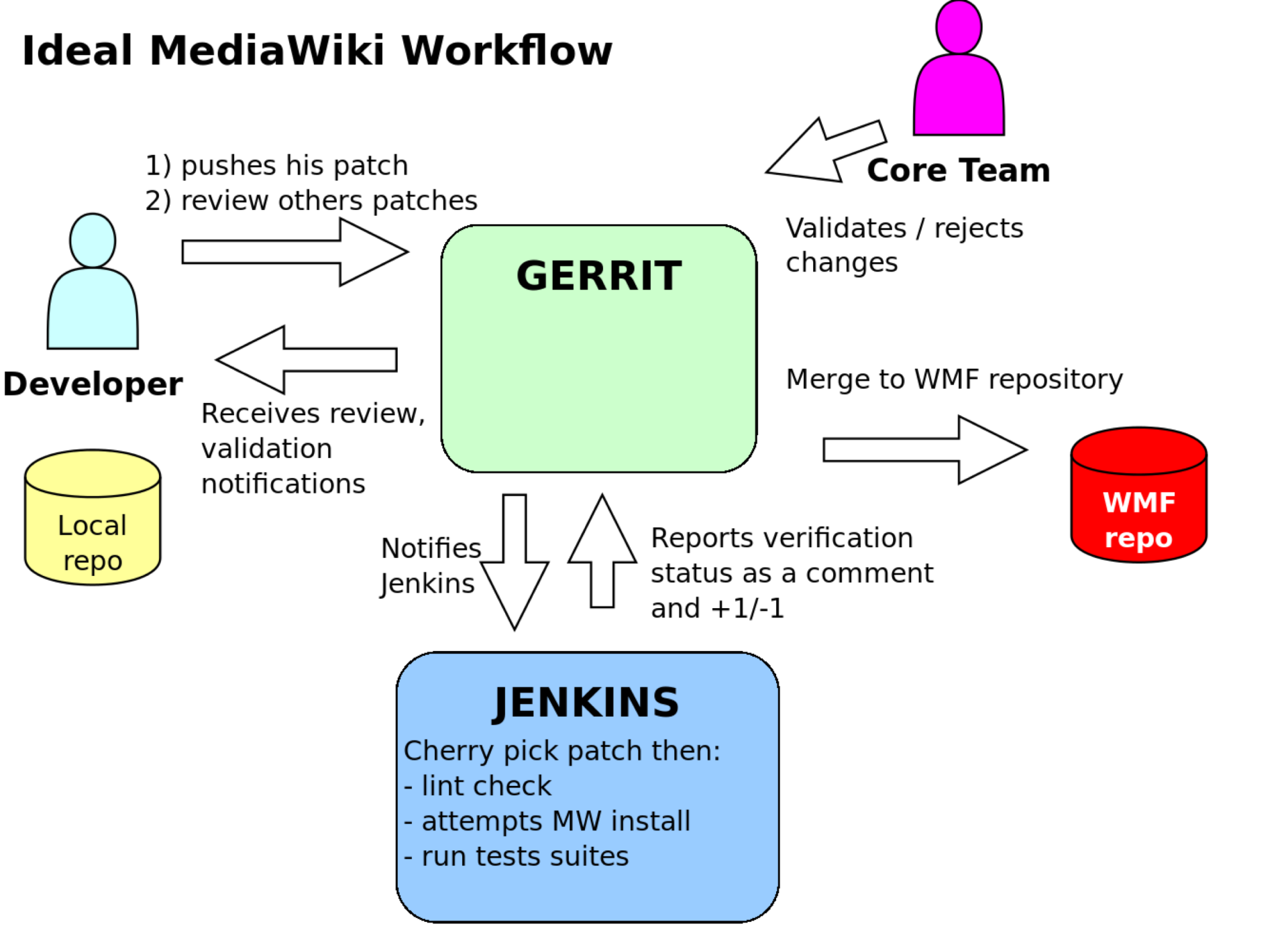
- Different background, different experience
- No preconceived idea of correctness
- Not biased by “what was intended”


or validation helpers */
dation-error {
#b94a48;
dation-valid {
: none;
-validation-error {
1px solid #ddd;
Team Explorer - Home
Search Work Items (Ctrl+')
Home | Fabrikam Fiber
Project
Web Portal | Task Board | Team Room
My Work
Pending Changes
Source Control Explorer
Work Items
Builds
Settings
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 - New Code Review
Search Work Items (Ctrl+')
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
Team Explorer - My Work
Search Work Items (Ctrl+')
My Work | Fabrikam Fiber
Streaming Video: How to multi-task with My Work
In Progress Work
Suspend
Request Review
Check In | Actions
1 edit(s) | View Changes
Suspended Work
Resume | Merge with In Progress
No suspended work.


D212 Fix daemon issues
https://secure.phabricator.com/D212
To hatena
QRコード
ニコニコチャンネル
Press This
Game on HTML5
Pin It
日本語化
inky-linky
deCSS3
Shareist Bookmarklet
その他のブックマーク
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


Change I1f962956: Added g
https://gerrit.wikimedia.org/r/#/c/9332/
All My Admin Documentation
Changes Drafts Watched Changes Starred Changes
preilly <preilly@wikimedia.org> Settings Sign Out
Change #, SHA-1, tr:id, owner:email or reviewer:email Search
Change I1f962956: Added get version method to extension
Change-Id: I1f962956e9cf9c404c2fc685963964978ef52516
Owner preilly
Project test/mediawiki/extensions/examples
Branch master
Topic 2012/bug12345
Uploaded May 29, 2012 1:25 PM
Updated May 29, 2012 1:34 PM
Status Review in Progress
Permalink
Added get version method to extension
Change-Id: I584255f80c7e2634617a803e0a387c1bfa6480ee
Added get version method to extension
Change-Id: I1f962956e9cf9c404c2fc685963964978ef52516
Gerrit
(open source)
Reviewer Verified Code-Review
preilly
Need Verified
Need Code-Review
Name or Email or Group
Add Reviewer
Dependencies
Old Version History: Base
Patch Set 1 fee0e992a68d285a727e39608cb608e535bcc10a (gitweb)
Patch Set 2 7b7840b47096145bf0560d645fe8b39d848601c (gitweb)
Author preilly <preilly@wikimedia.org> May 29, 2012 1:13 PM
Committer preilly <preilly@wikimedia.org> May 29, 2012 1:31 PM
Parent(s) 7cf0d68d9553c16d3e426de213e7db11d14ca6ea Merge "Small change for the sake of review test"
Download checkout pull cherry-pick patch Anonymous HTTP SSH HTTP
git fetch https://preilly@gerrit.wikimedia.org/r/test/mediawiki/extensions/examples refs/changes/32/9332/2 && git checkout FETCH_HEAD
Review
Abandon Change
Diff All Side-by-Side
Diff All Unified
File Path Comments Size Diff Reviewed
Commit Message Side-by-Side Unified
M Example/Example.body.php +7, -0 Side-by-Side Unified
+7, -0
Comments
preilly 1:34 PM
Uploaded patch set 2.
Press "?" to view keyboard shortcuts
Powered by Gerrit Code Review (2.3) | Report Bug

Ideal MediaWiki Workflow



http://www.mediawiki.org/wiki/Gerrit/Advanced_usage

Date Thu, 16 Oct 2014 14:47:41 +0200
From Greg Kroah-Hartman <>
Subject [PATCH] staging: android: binder: move to the "real" part of the kernel

From: Greg Kroah-Hartman <gregkh@linuxfoundation.org>

The Android binder code has been "stable" for many years now. No matter
what comes in the future, we are going to have to support this API, so
might as well move it to the "real" part of the kernel as there's no
real work that needs to be done to the existing code.

Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>

This was discussed in the Android miniconf at the Plumbers conference.
If anyone has any objections to this, please let me know, otherwise I'm
queueing this up for 3.19-rc1

```
 drivers/Kconfig                                    |  2 ++
 drivers/Makefile                                   |  1 +
 drivers/android/Kconfig                            | 37 ++++++++++++++++++++++++
 drivers/android/Makefile                           |  3 ++
 drivers/{staging => }/android/binder.c             |  0
 drivers/{staging => }/android/binder.h             |  2 +-
 drivers/{staging => }/android/binder_trace.h       |  0
 drivers/staging/android/Kconfig                    | 30 ----------------------
 drivers/staging/android/Makefile                   |  1 -
 include/uapi/linux/Kbuild                          |  1 +
 include/uapi/linux/android/Kbuild                  |  2 ++
 .../uapi => include/uapi/linux/android}/binder.h   |  0
 12 files changed, 47 insertions(+), 32 deletions(-)
 create mode 100644 drivers/android/Kconfig
 create mode 100644 drivers/android/Makefile
 rename drivers/{staging => }/android/binder.c (100%)
 rename drivers
 rename drivers
 create mode 100644 include/uapi/linux/android/Kbuild
 rename {drivers/staging/android/uapi => include/uapi/linux/android}/binder.h (100%)
```

https://www.kernel.org/doc/Documentation/SubmittingPatches

Process: Checklists!

https://en.wikipedia.org/wiki/File:B17_-_Chino_Airshow_2014_(framed).jpg

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

B-17F AND B-17G

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

PILOT

BEFORE STARTING

1. Pilot's Pre-flight — Complete.
2. Form 1A, 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 Selector — 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 $1/3$ — $1/3$ Down

FINAL APPROACH

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

The Checklist: https://www.newyorker.com/magazine/2007/12/10/the-checklist

Activity

DEVELOP CHECKLIST FOR CODE REVIEW

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

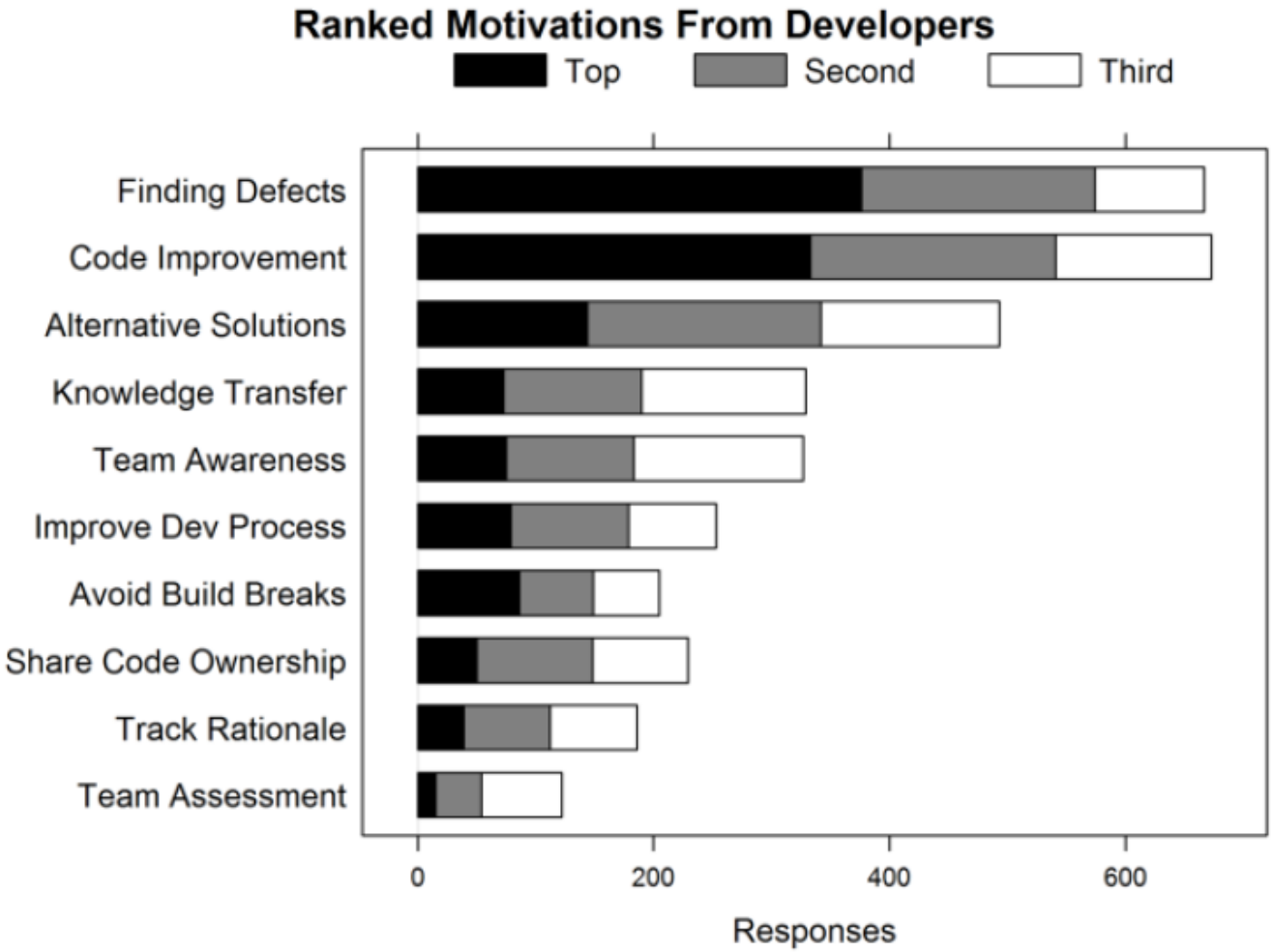
Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.

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

Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.

Code Review at Microsoft

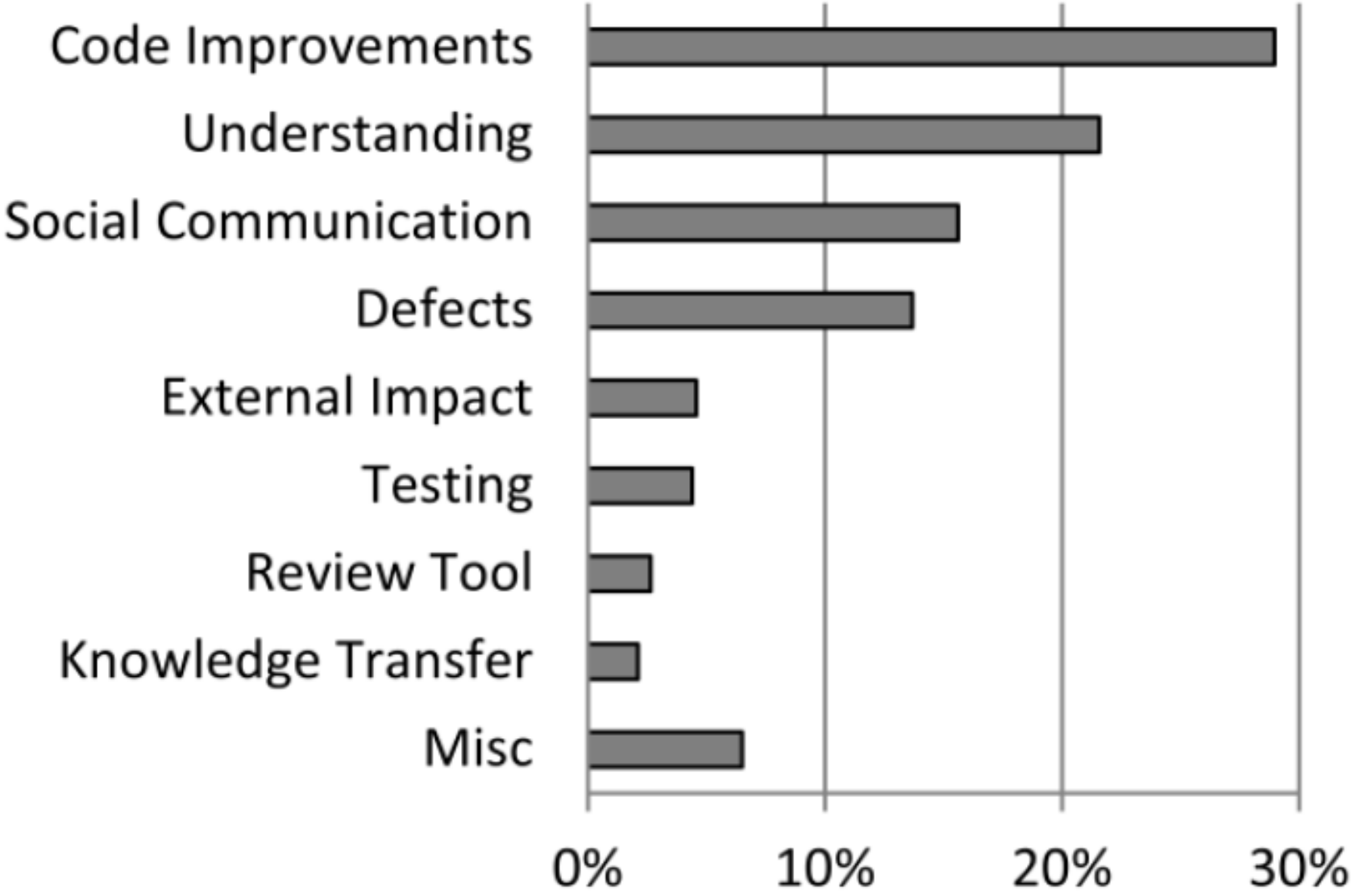


Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.

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

Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.



Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.

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

Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.

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

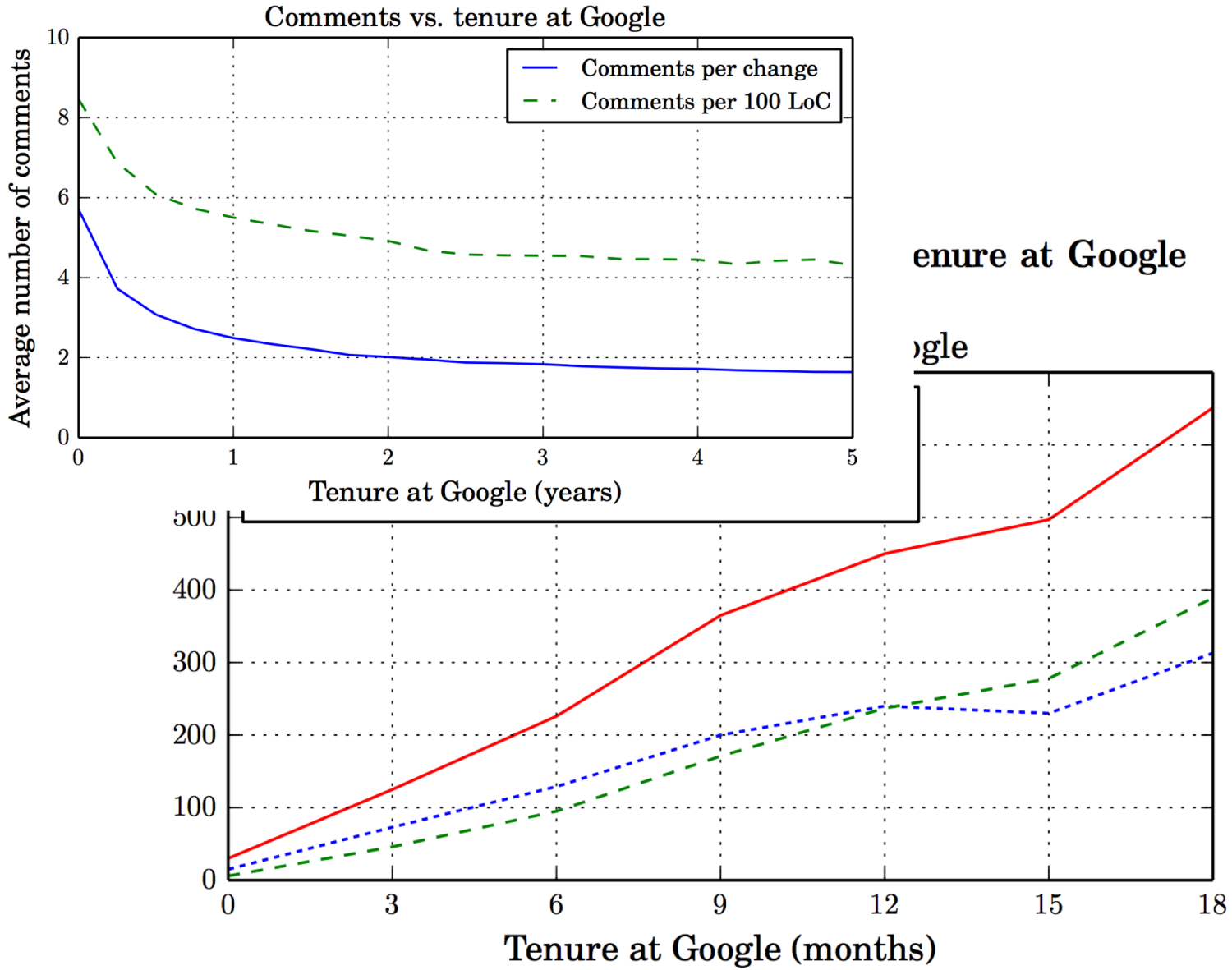
Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko and Alberto Bacchelli. 2018. Modern Code Review: A Case Study at Google. International Conference on Software Engineering

Google's Code Review Policy

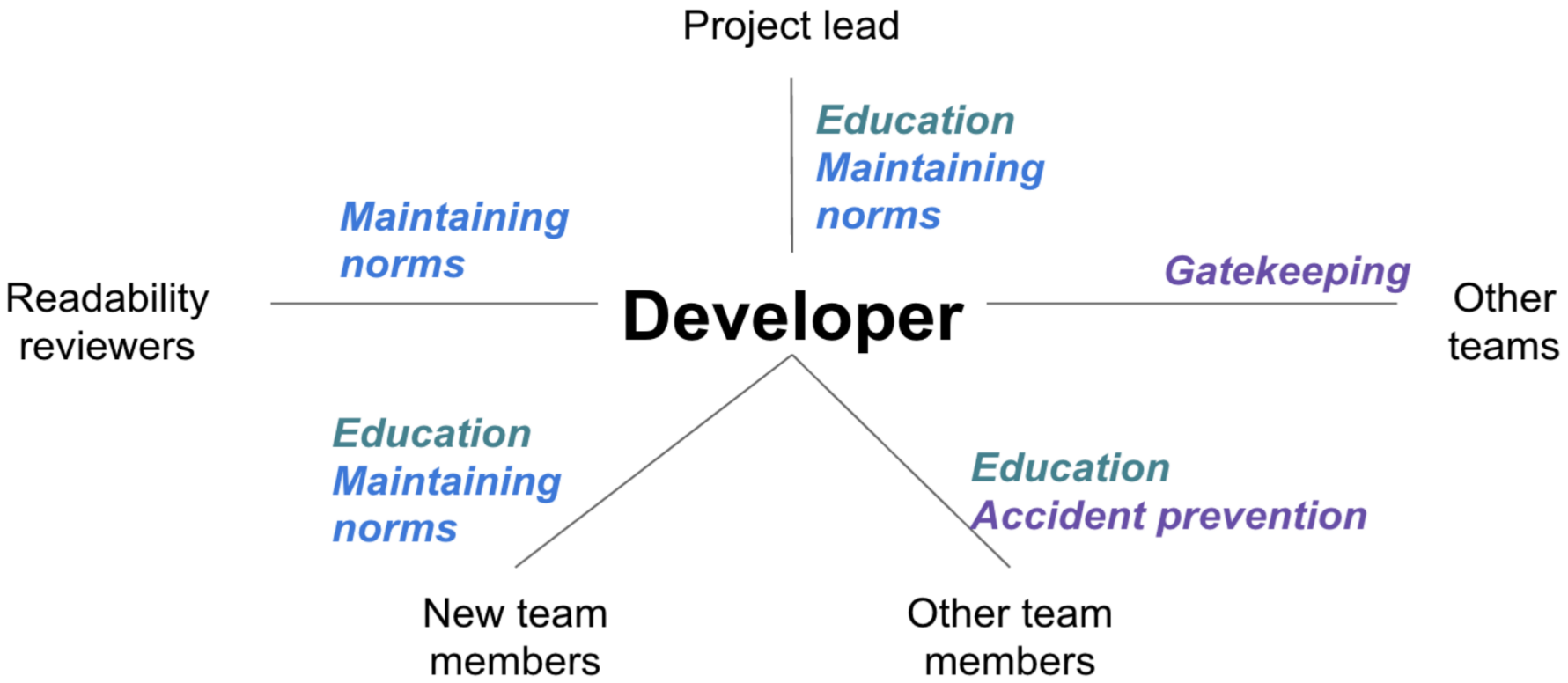
- All change lists must be reviewed. Period.
- Any CL can be reviewed by any engineer at Google.
- Each directory has a list of owners. At least one reviewer or the author must be an owner for each file that was touched in the commit. If the author is not in the owners file, the reviewer is expected 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


Code Review Dashboard for Guido van Rossum
Changes Awaiting Your Review
Your Changes Awaiting Review


Comments vs. tenure at Google
Comments per change
Comments per 100 LoC
Average number of comments
Tenure at Google (years)
enure at Google
ogle
Tenure at Google (months)

Reviewing relationships



Google vs. Microsoft?

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: Wiegers. 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. 8th 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

Quality Assurance Process

Claire Le Goues

November 5, 2019

How to get developers to [write tests | use static analysis | appreciate testers]

Claire Le Goues

November 5, 2019

QA Process Considerations

- We covered several QA techniques:
  - Formal verification (15-112)
  - Unit testing, test driven development
  - Various forms of advanced testing for quality attributes (GUI testing, fuzz testing, …)
  - Static analysis
  - Dynamic analysis
  - Formal inspections and other forms of code reviews
- But: When to use? Which techniques? How much? How to introduce? Automation? How to establish a quality culture? How to ensure compliance? Social issues? What about external components?

Jaspan, Ciera, I. Chen, and Anoop Sharma. "Understanding the value of program analysis tools." *Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion*. ACM, 2007.

CASE STUDY: INTRODUCING STATIC ANALYSIS AT EBAY

Findbugs in 214

- We forced everybody to use Findbugs
- Has it found bugs?
- Who is still using Findbugs?
- Why not?

Ebay: Prior Evaluations

- Individual teams tried tools
  - On snapshots
  - No tool customization
  - Overall negative results
  - Developers were not impressed: many minor issues (2 checkers reported half the issues, all irrelevant for Ebay)
- Would this change when integrated into process? i.e. incremental checking
- Which bugs to look at?

Ebay: Goals

- Find defects earlier in the lifecycle
  - Allow quality engineers to focus on different issues
- Find defects that are difficult to find through other QA techniques
  - security, performance, concurrency
- As early as feasible: Run on developer machines and in nightly builds
- No resources to build own tool
  - But few people for dedicated team (customization, policies, creating project-specific analyses etc) possible
- Continuous evaluation

Ebay: Customization

- Customization dropped false positives from 50% to 10%
- Separate checkers evaluated separately
  - By number of issues
  - By severity as judged by developers; iteratively with several groups
- Some low-priority checkers (e.g., dead store to local) was assigned high priority – performance impact important for Ebay

Ebay: Enforcement policy

- High priority: All these issues must be fixed (e.g. null pointer exceptions)
  - Potentially very costly given the huge existing code base
- Medium priority: May not be added to the code base. Old issues won't be fixed unless refactored anyway (e.g., high cyclomatic complexity)
- Low priority: At most X issues may be added between releases (usually stylistic)
- Tossed: Turned off entirely

Ebay: Cost estimation

- Free tool
- 2 developers full time for customization and extension
- A typical tester at ebay finds 10 bugs/week, 10% high priority
- Sample bugs found with Findbugs for a comparison

Aside: Cost/benefit analysis

- Cost/Benefit tradeoff
  - Benefit: How valuable is the bug?
    - How much does it cost if not found?
    - How expensive to find using testing/inspection?
  - Cost: How much did the analysis cost?
    - Effort spent running analysis, interpreting results – includes false positives
    - Effort spent finding remaining bugs (for unsound analysis)
- Rule of thumb
  - For critical bugs that testing/inspection can't find, a sound analysis is worth it, as long as false positive rate is acceptable.
  - For other bugs, maximize engineer productivity

Ebay: Combining tools

- Program analysis coverage
  - Performance – High importance
  - Security – High
  - Global quality – High
  - Local quality – medium
  - API/framework compliance – medium
  - Concurrency – low
  - Style and readability – low
- Select appropriate tools and detectors

Ebay: Enforcement

- Enforcement at dev/QA handoff:
- Developers run FindBugs on desktop
- QA runs FindBugs on receipt of code, posts results, require high-priority fixes.

Ebay: Continuous evaluation

- Gather data on detected bugs and false positives
- Present to developers, make case for tool

Incremental introduction

- Begin with early adopters in small team
- Use these as champions in organization

- Support team: answer questions, help with tool.