# Foundations of Software Engineering

Part 15: Inspections and Reviews

Christian Kästner



institute for SOFTWARE RESEARCH

# Administrivia

# Find the Bug(s)!

```
BlockingQueue queue = …

while (!queue.isEmpty() && ...) {
     CheaterFutureTask Task =
              queue.remove();
     incompleteTasks.add(Task);
     taskValues.add(
          Task.getRawCallable().
              call());
}
```

BatchCommitLogExecutorService.java using BlockingQueue in Cassandra, one bug injected

institute for SOFTWARE RESEARCH

# Software Peer Reviews

# Learning Goals

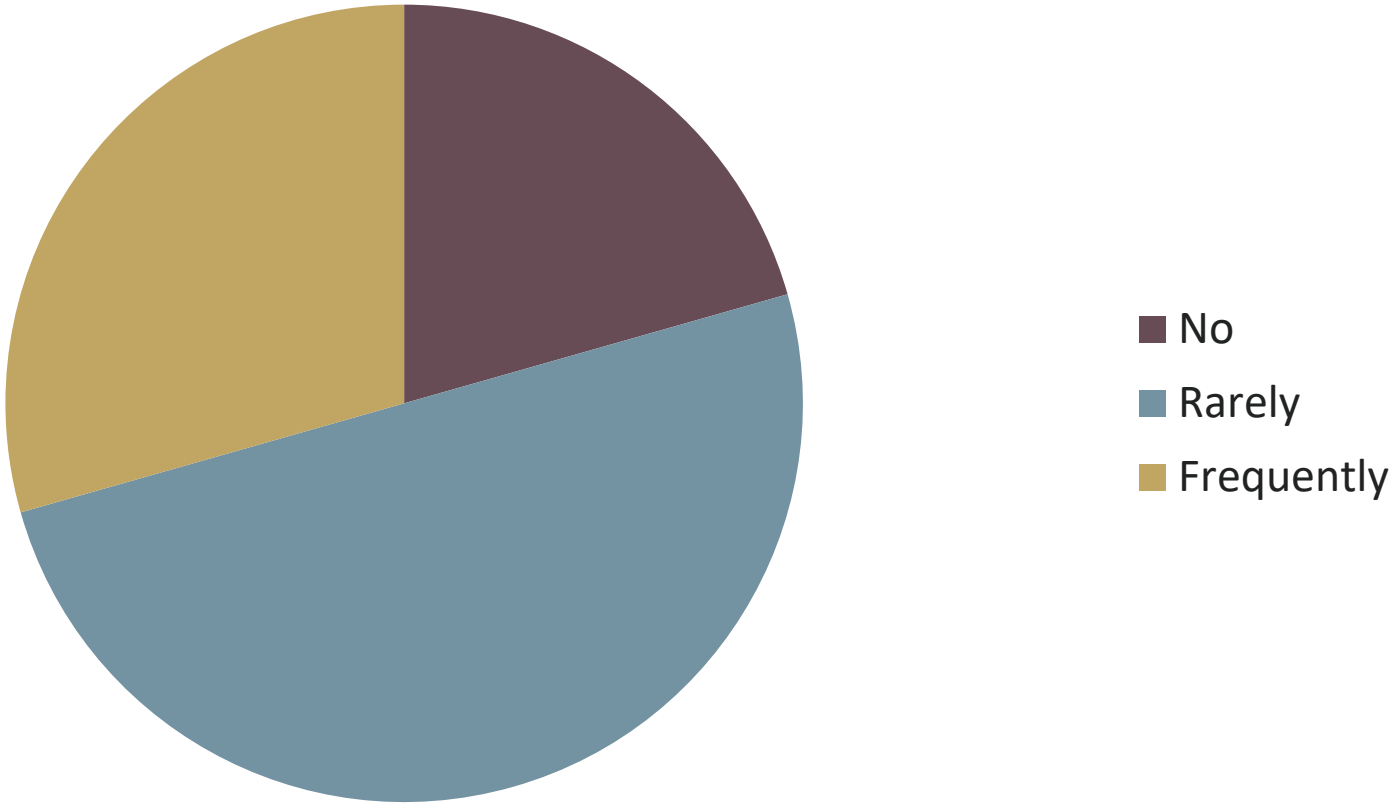
- Understand different forms of peer reviews with different formality levels
- Select appropriate review forms for a project
- Conduct an inspection session, aware of common pitfalls and social issues
- Perform code reviews with automated software tools
- Understand the expectations and outcomes of modern peer reviews

# Agenda

- Modern Code Reviews
  - Expectations and outcomes
- Formal Inspections
  - Roles and process
  - Social issues
  - Experience
- Other forms of code reviews

# About you

**I have participated in code reviews (or pull requests)**






institute for SOFTWARE RESEARCH

# What are Code Reviews?

Hub, Inc. [US] 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 th

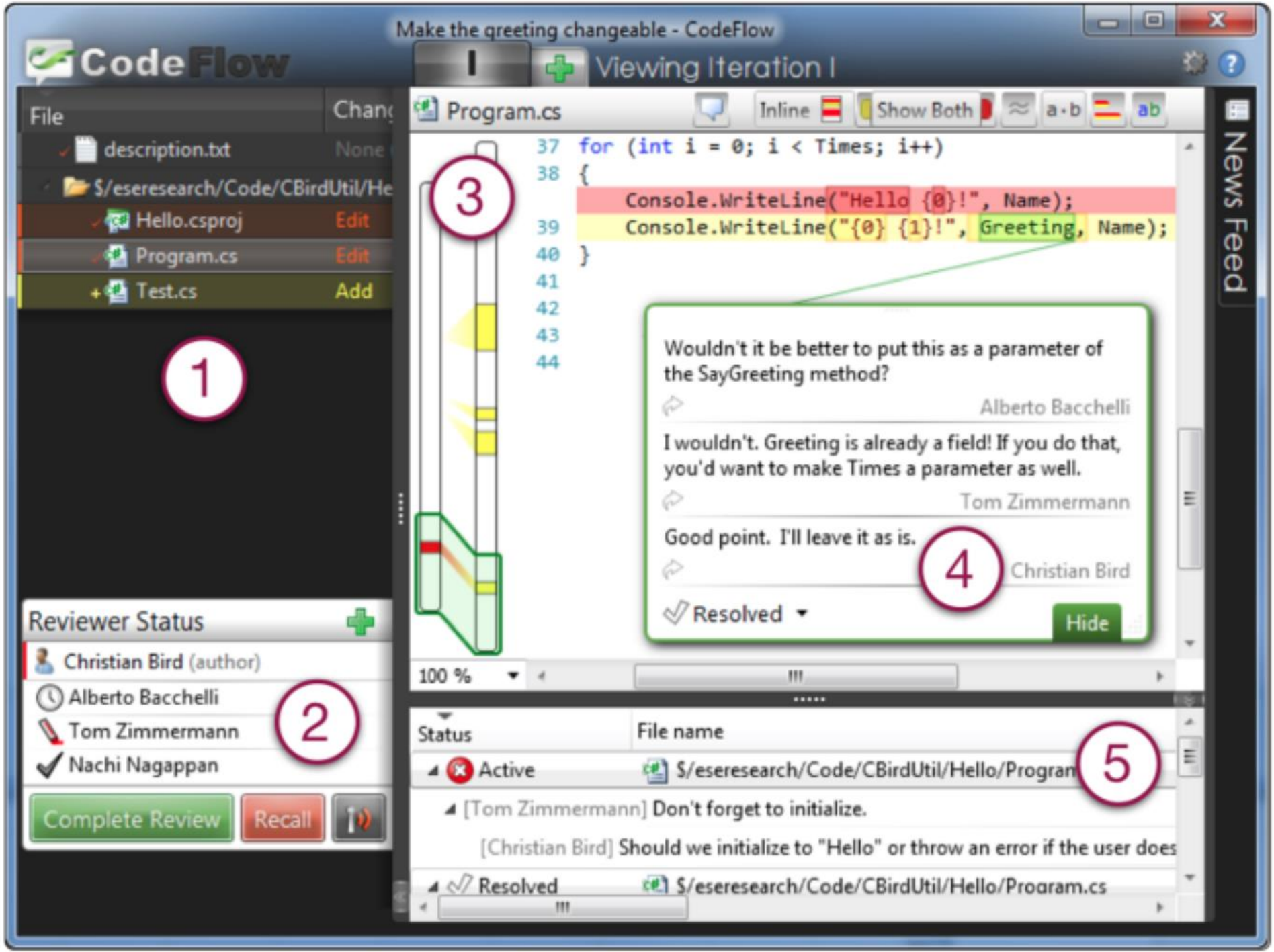
Labels
None yet

Milestone
No milestone

Assignee
No one assigned

2 participants

https://help.github.com/articles/using-pull-requests/


Make the greeting changeable - CodeFlow
CodeFlow
Viewing Iteration 1
File
Chang
description.txt
None
$/eseresearch/Code/CBirdUtil/He
Hello.csproj
Edit
Program.cs
Edit
Test.cs
Add
1
Reviewer Status
Christian Bird (author)
Alberto Bacchelli
Tom Zimmermann
Nachi Nagappan
2
Complete Review
Recall
Program.cs
Inline
Show Both
a·b
ab
3
37 for (int i = 0; i < Times; i++)
38 {
Console.WriteLine("Hello {0}!", Name);
39 Console.WriteLine("{0} {1}!", Greeting, Name);
40 }
41
42
43
44
Wouldn't it be better to put this as a parameter of the SayGreeting method?
Alberto Bacchelli
I wouldn't. Greeting is already a field! If you do that, you'd want to make Times a parameter as well.
Tom Zimmermann
Good point. I'll leave it as is.
Christian Bird
4
Resolved
Hide
100 %
News Feed
Status
File name
Active
$/eseresearch/Code/CBirdUtil/Hello/Program
5
[Tom Zimmermann] Don't forget to initialize.
[Christian Bird] Should we initialize to "Hello" or throw an error if the user does
Resolved
$/eseresearch/Code/CBirdUtil/Hello/Program.cs


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

# 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

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**

Press ? to show keyboard shortcuts.

- Subscribe
- Edit Dependencies
- Edit Maniphest Tasks
- Herald Transcripts
- Download Raw Diff
- Award Token
- Flag For Later

**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**

---

**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: I584255f80c7e2634617a803e0a387c1bfa6400ee

Added get version method to extension

Change-Id: I1f962956e9cf9c404c2fc685963964978ef52516
```

| 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 fee9e992a68d285a727e39608cb608e535bcc10a (gitweb)

▼ Patch Set 2 7b7840b470961405bf0560d645fe6b39d848601c (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.

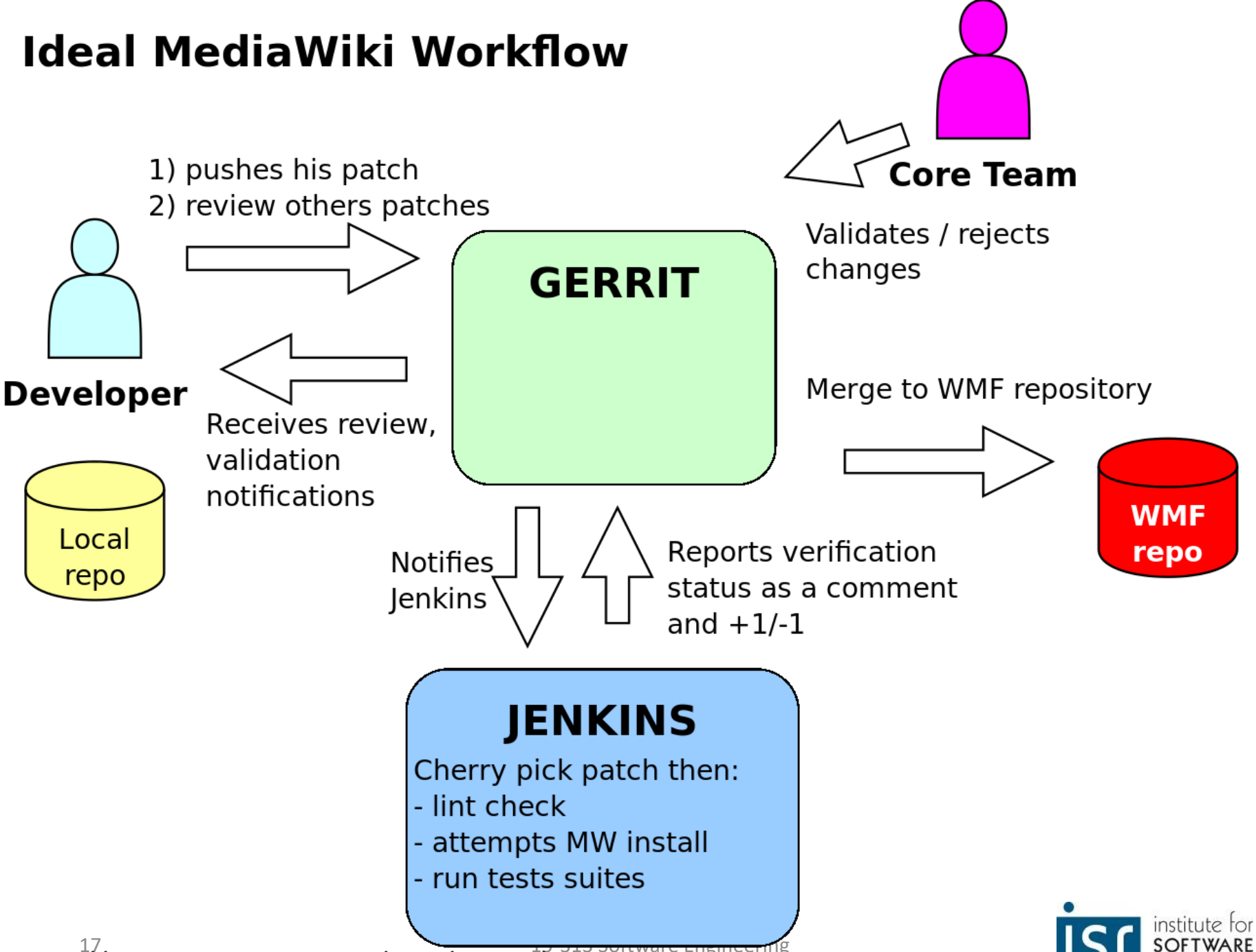
Press "?" to view keyboard shortcuts

Powered by Gerrit Code Review (2.3) | Report Bug

**Gerrit**

(open source)

# Ideal MediaWiki Workflow




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



institute for SOFTWARE RESEARCH

//lkml.org/lkml/2014/10/16/359

[lkml] [2014] [Oct] [16] [last100] RSS
Views: [wrap] [headers] [forward]

**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/{staging => }/android/binder.h (95%)
 rename drivers/{staging => }/android/binder_trace.h (100%)
 create mode 100
 rename {drivers
diff --git a/drivers/Kconfig b/drivers/Kconfig
index 1a693d3f9d51..569ff7886dc3 100644
--- a/drivers/Kconfig
```

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

Hub, Inc. [US] 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 th

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”

# Expectations and Outcomes of Modern Code Reviews

# Reasons for Code Reviews

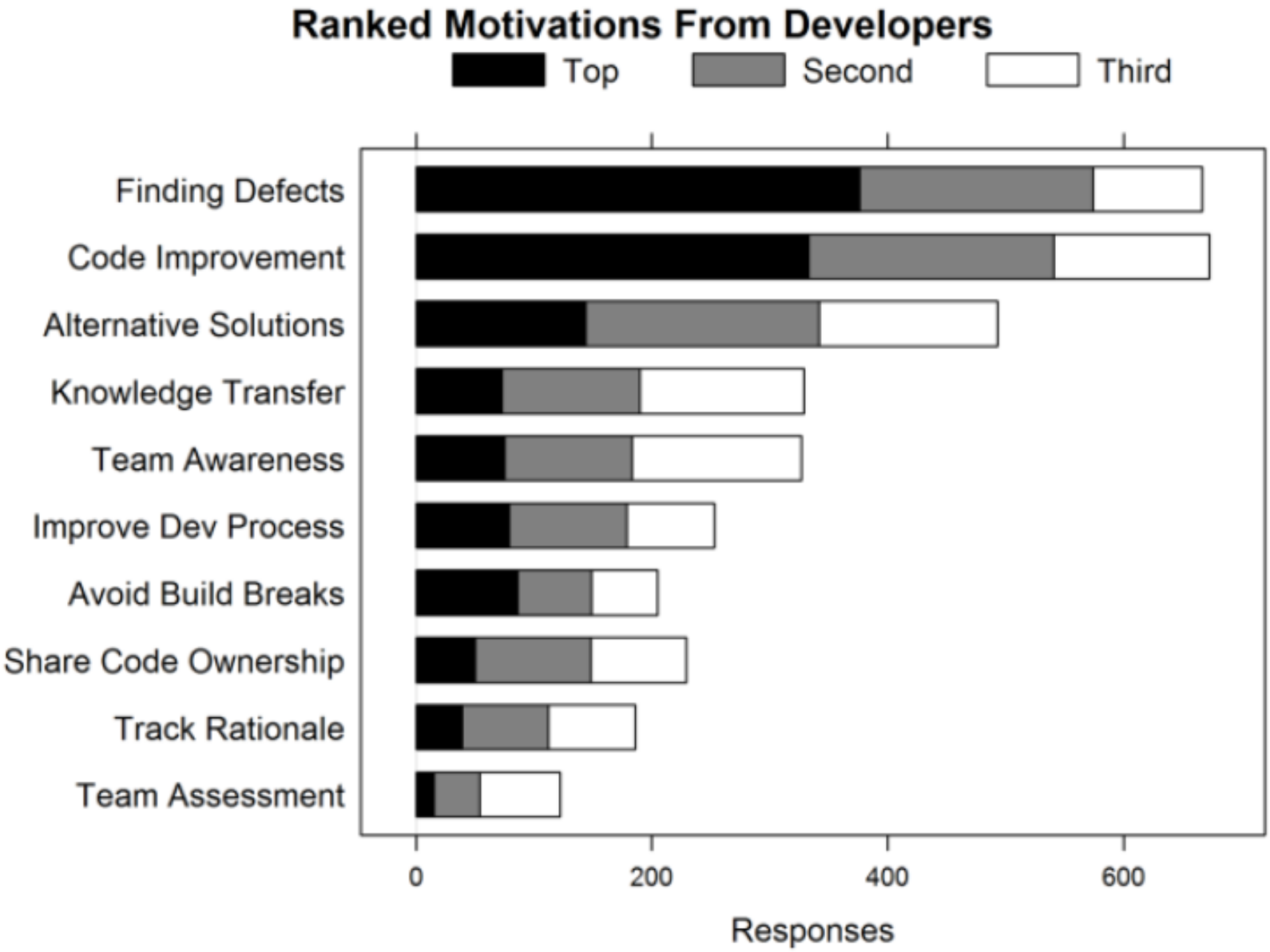
- Finding defects
  - both low-level and high-level issues
  - requirements/design/code issues
  - security/performance/… issues
- Code improvement
  - readability, formatting, commenting, consistency, dead code removal, naming
  - enforce to coding standards
- Identifying alternative solutions
- Knowledge 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.



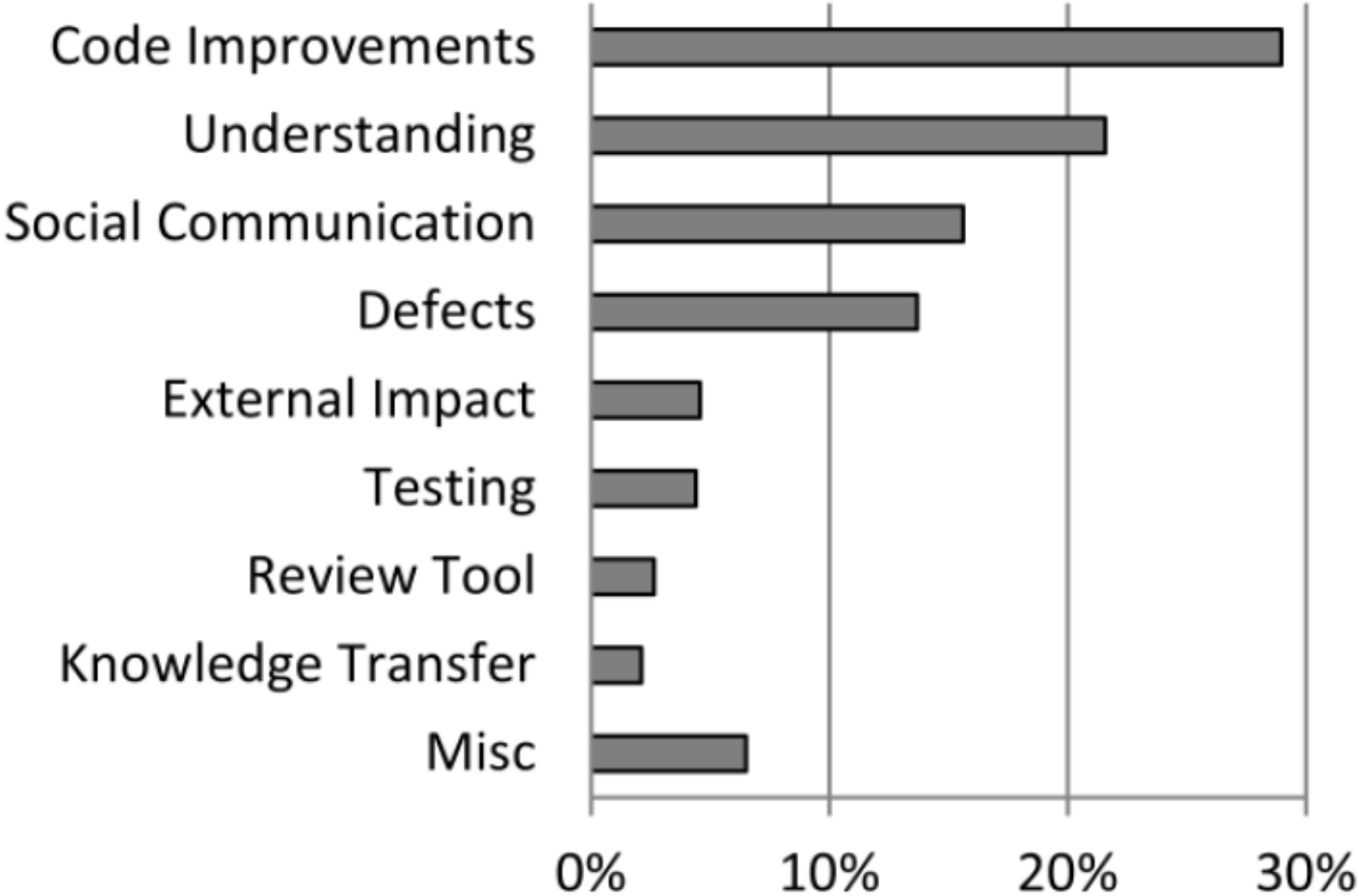

Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.

# 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.

# Outcomes (Analyzing Reviews)




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.

# Formal Inspections

# Formal Inspections

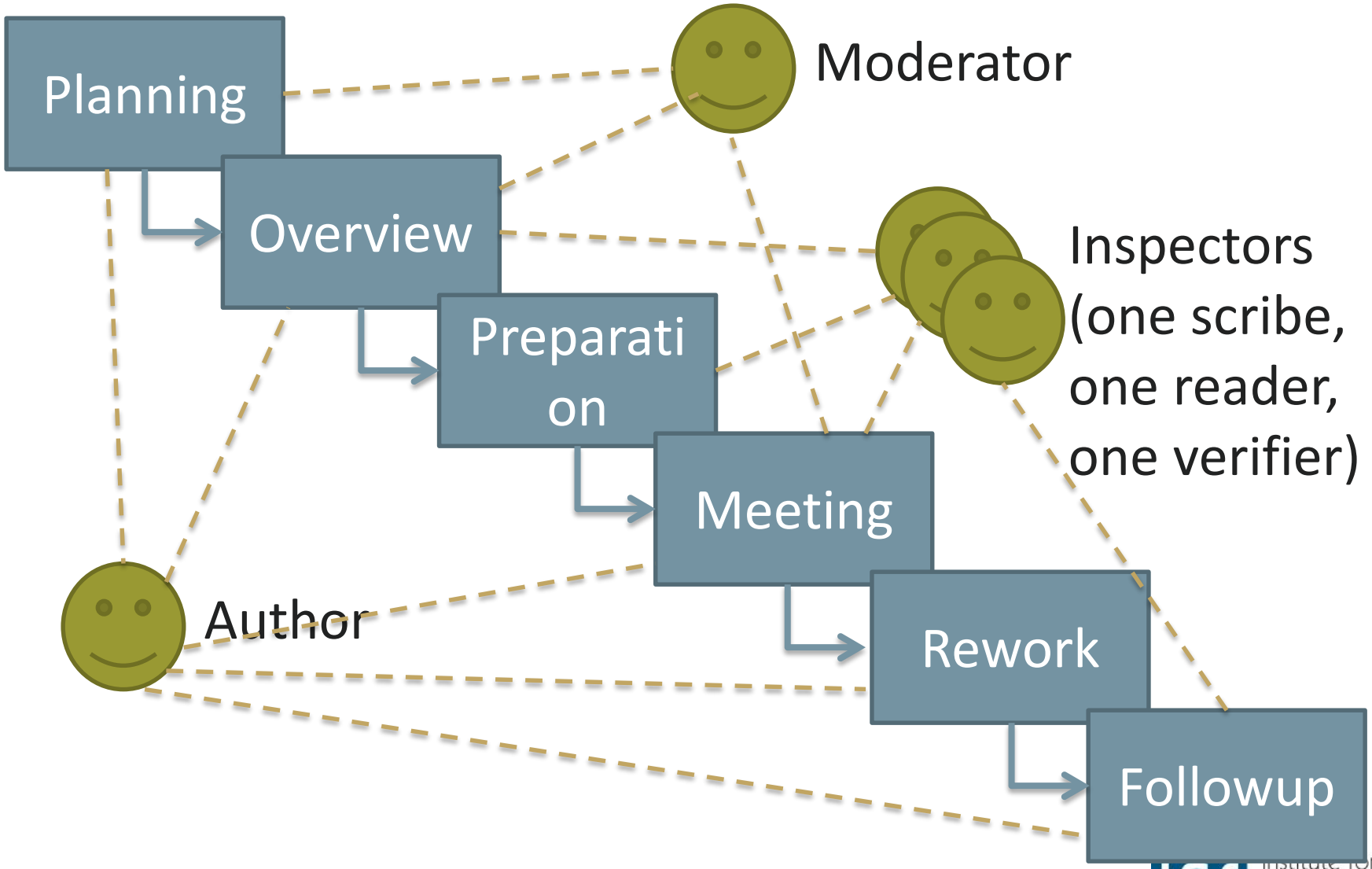
- Idea popularized in 70s at IBM
- Broadly adopted in 80s, much research
  - Sometimes replacing component testing
- Group of developers meets to formally review code or other artifacts
- Most effective approach to find bugs
  - Typically 60-90% of bugs found with inspections
- Expensive and labor-intensive

(see textbook Chapter 22.2)

# 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



# Inspection Process

- Planning
  - Select Moderator
- Overview (brief)
  - Author presents context in meeting
- Preparation (1-2h)
  - Every reviewer inspects the code separately
- Meeting (1h)
  - Moderator conducts meeting
  - Reader presents the code
  - All reviewers identify issues
  - Meetings only discover issues, do not discuss solution or whether it really is an issue
- Rework
  - Author corrects issues (fix code/documentation/…)
- Followup
  - Verifier checks changes
- Root cause analysis (optional) for process improvement

# 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?

# Perspective-based Inspections

- Have inspectors with different specialties or different focuses/checklists
  - Encourages alternative thinking patterns
- Have reviewers start in different places in the document
  - Avoid loosing focus at the same location
- Especially in preparation phase
- Little published data, but considered an effective practice

# 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

# 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

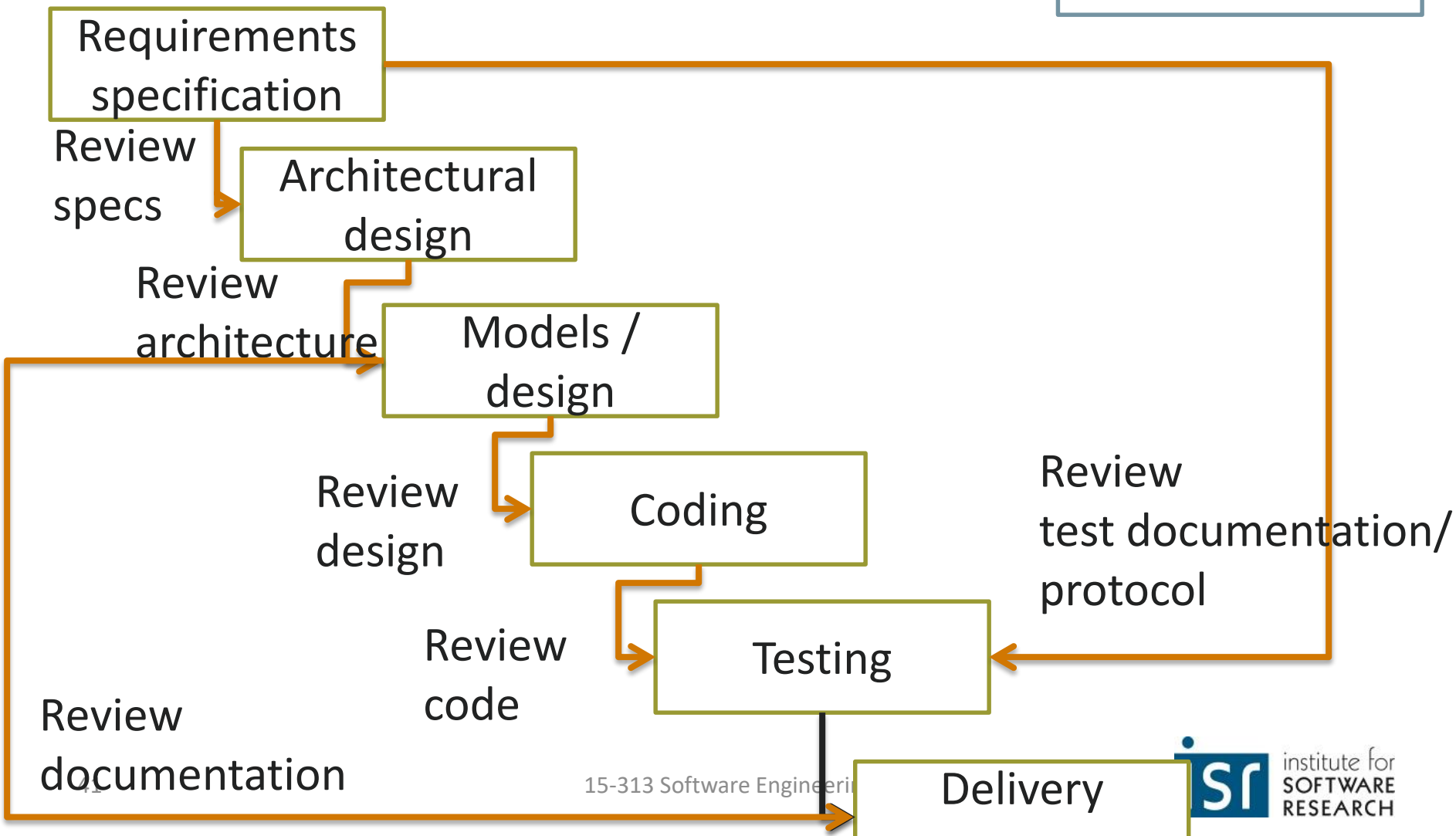
- Moderator must move discussion along, resolve conflicts
- Meetings should not include management
- Do not use for HR evaluation
  - “finding more than 5 bugs during inspection counts against the author”
  - Leads to avoidance, fragmented submission, not pointing out defects, holding pre-reviews
- Responsibility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

# Root Cause Analysis

- Beyond the immediate puzzle
- How to improve the development process to avoid this problem
  - Restructure development process
  - New policies
  - New development tools, new languages, new analysis tools

# Review Checkpoints during Lifecycle

Also reviewable:
Business plan
Marketing documents
Project plans
Documentation



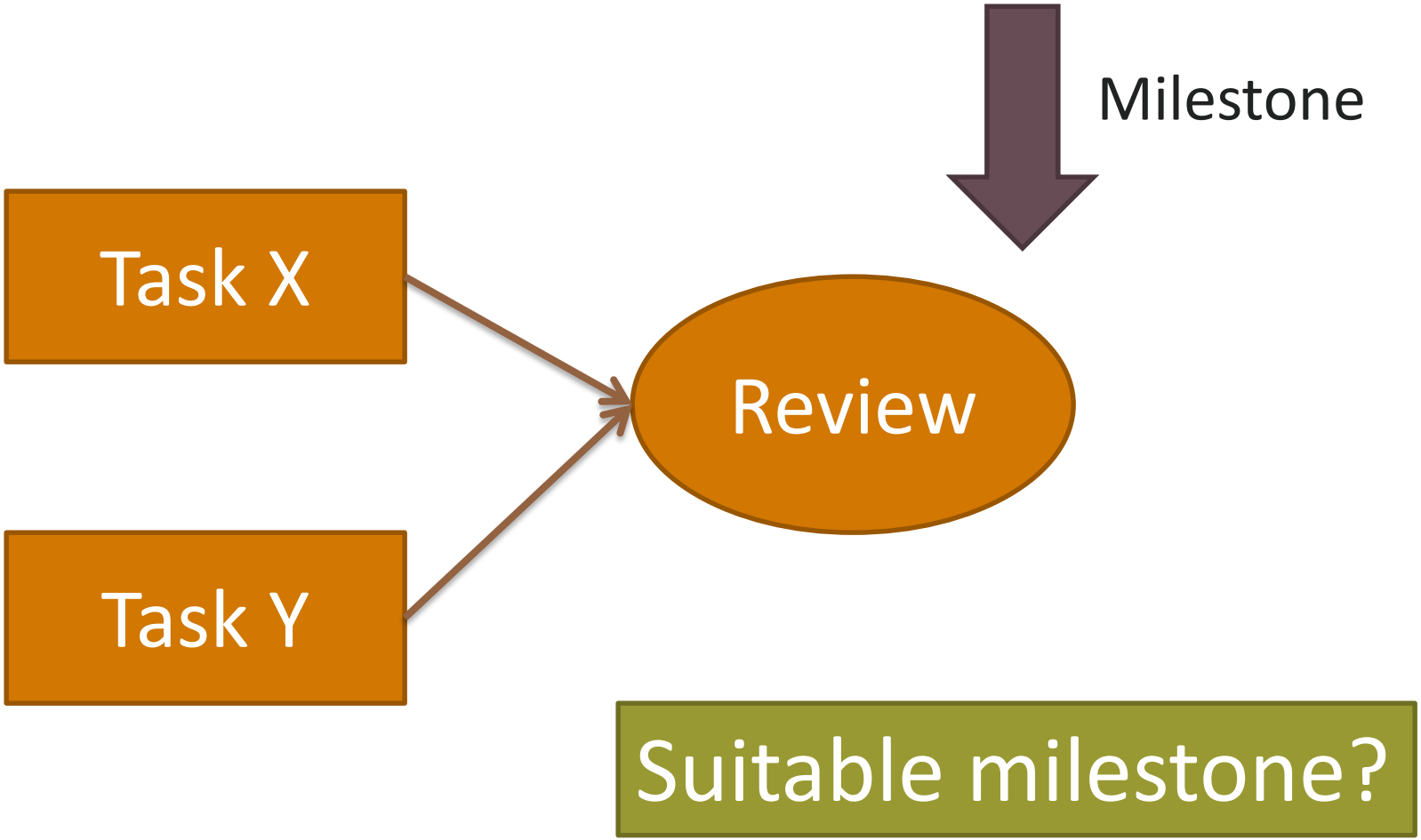



institute for SOFTWARE RESEARCH

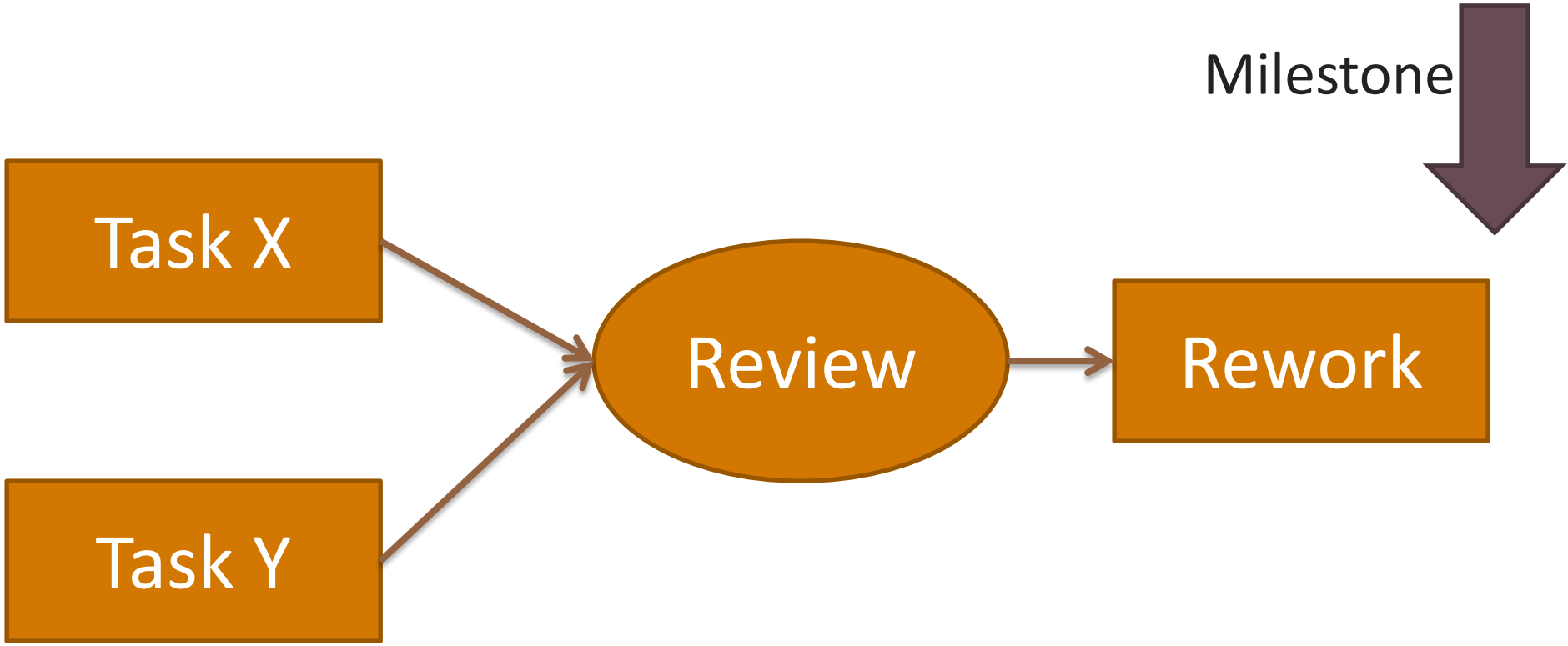
# When to inspect

- 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

# Reviews as part of a Milestone



# Reviews as part of a Milestone

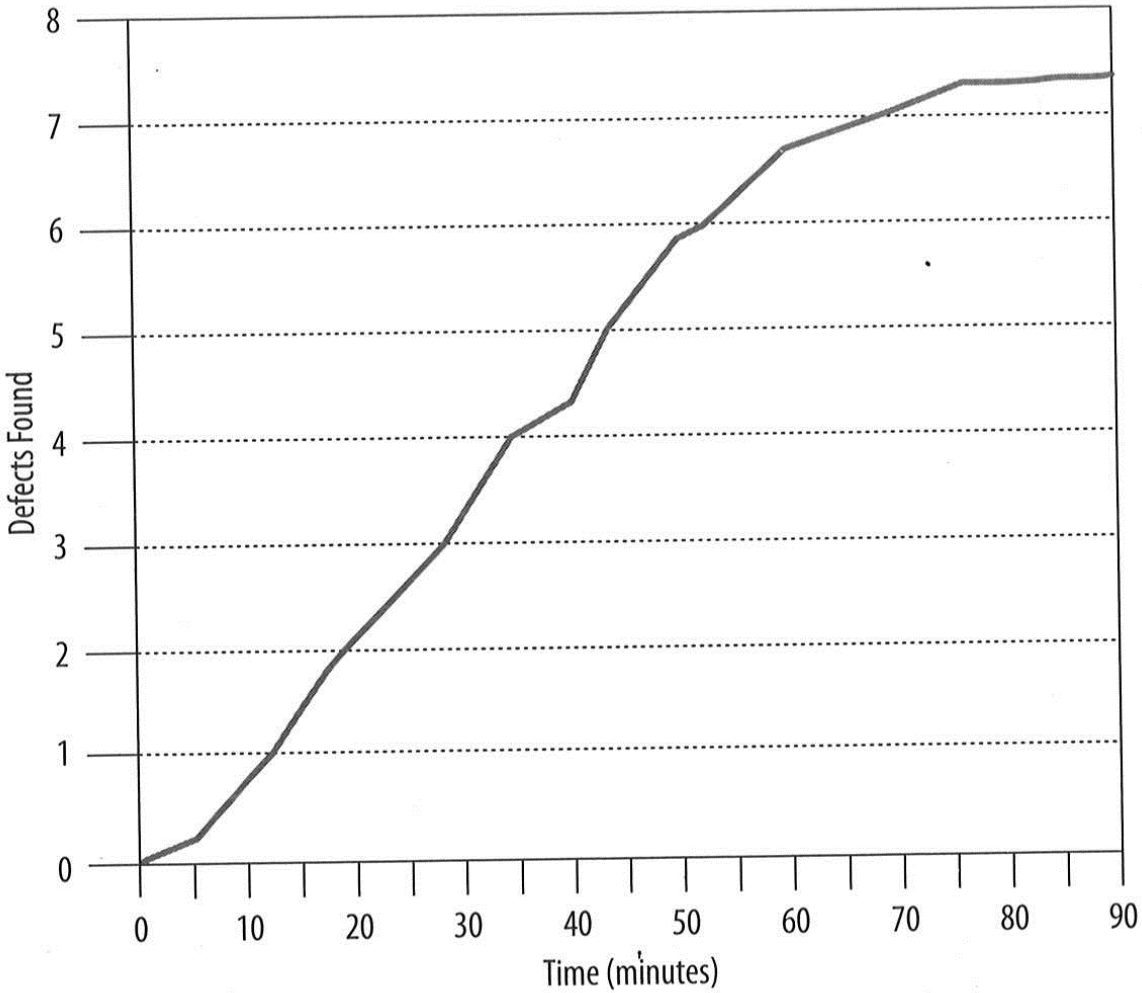


# Guidelines for Inspections

- Collected over many companies in many projects and experiments
- Several metrics easily measureable (effort, issues found, lines of code inspected) …

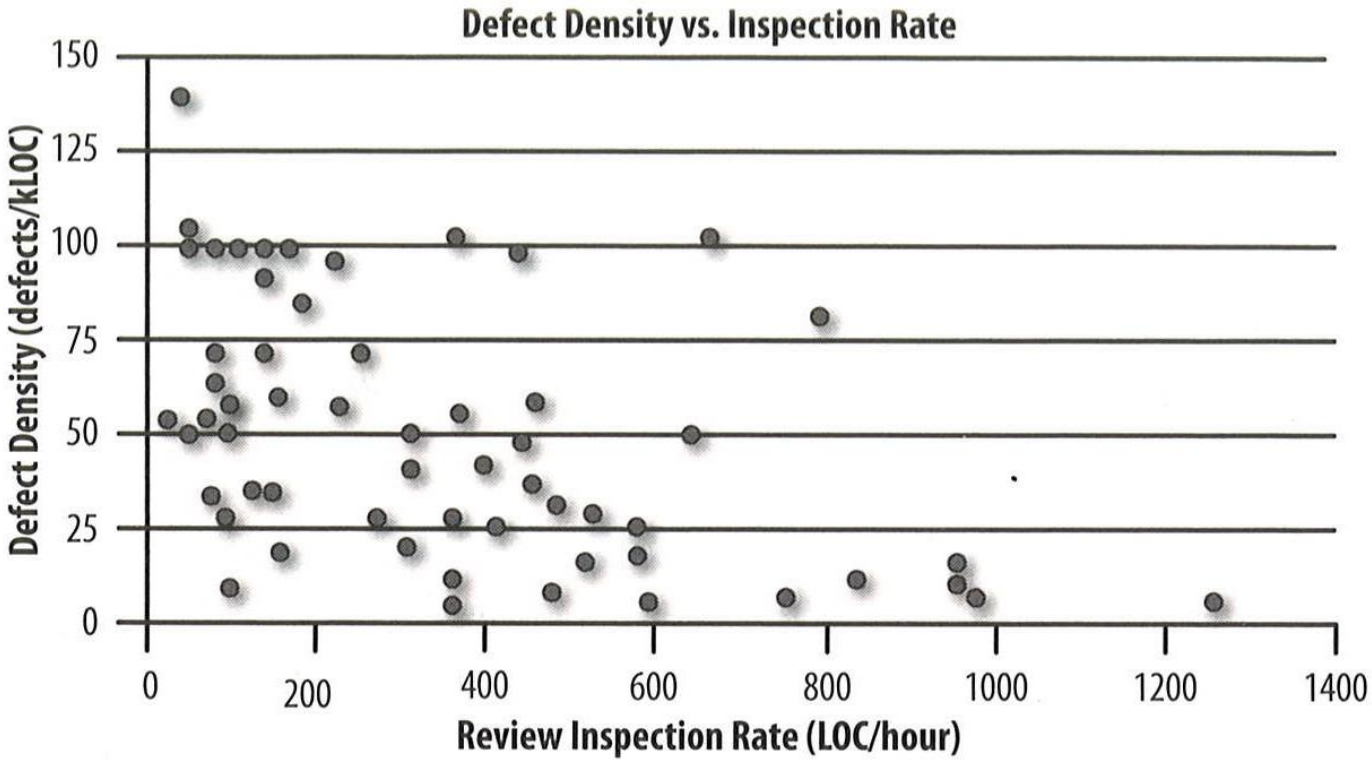
Source: Oram and Wilson (ed.). Making Software. O'Reilly 2010. Chapter 18 and papers reviewed therein

# Focus Fatigue




Recommendation:
Do not exceed
60 minute session

# Inspection speed



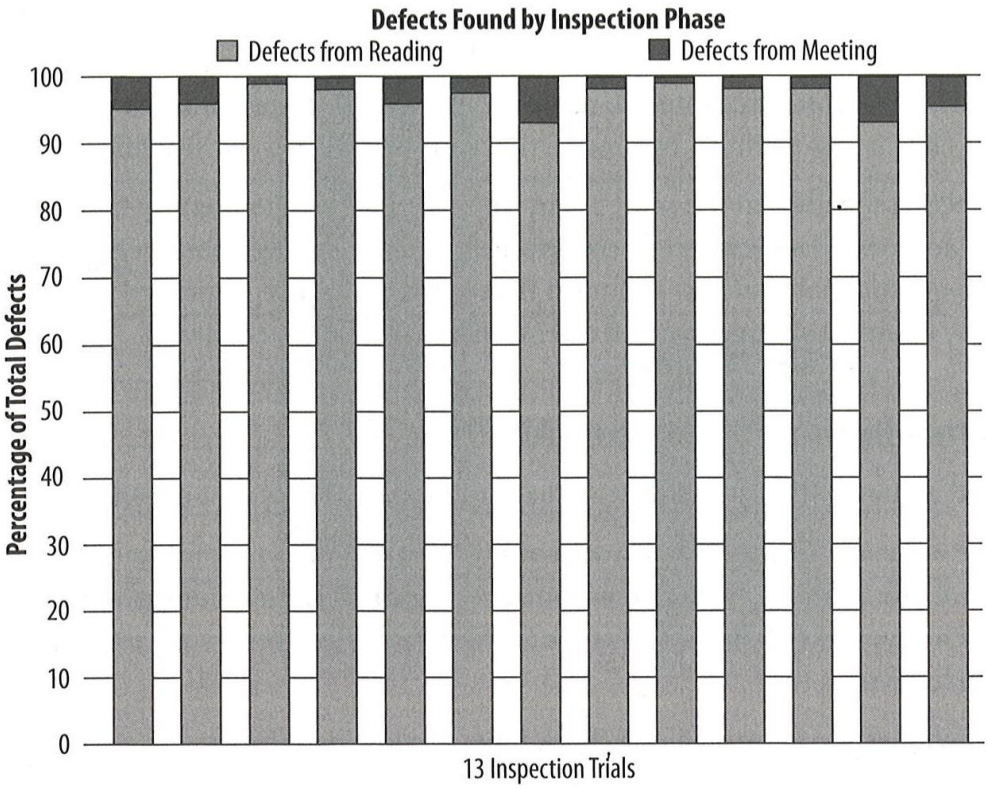

Above 400 LOC/h reviews get shallow

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

# Importance of Context

- Code with fewer context dependencies is easier to review
- Reviewers need to look at related files
- -> Modularity (small interfaces, high cohesion, low coupling, …)

# Are meetings required?

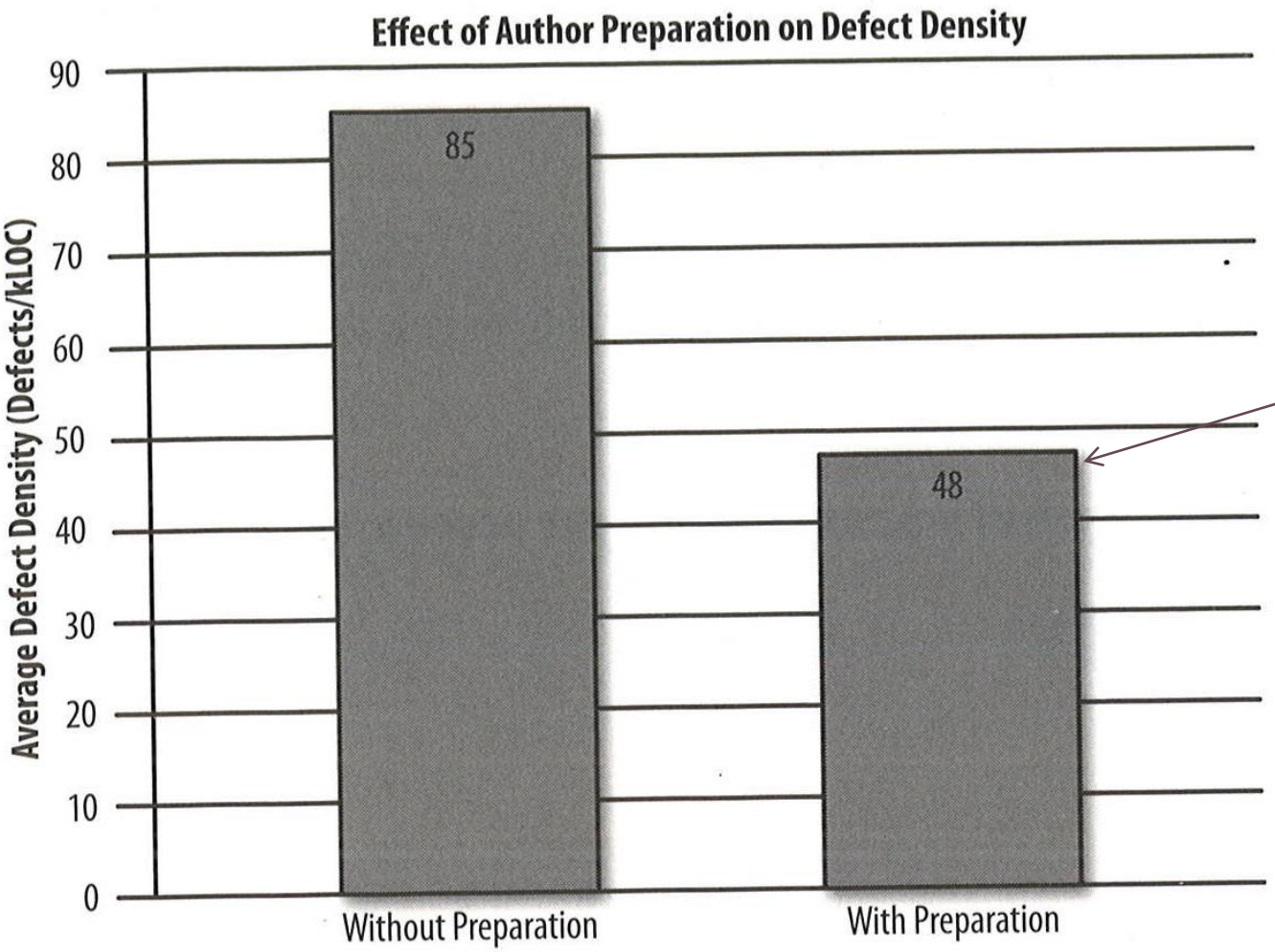



Most issues found during preparation, not in meeting.
Suggested synergy seems to have only low impact
Claim: Defects found in meetings often more subtle

# False positives

- About 25% of found issues are false positives
- Avoid discussing during meeting
- Confusion during meeting is indicator that document could be clearer

# Self-checks can find half the issues




Authors have self-checked their document before inspection



institute for SOFTWARE RESEARCH

# Arguments against Reviews?

# Arguments against Reviews

- Costs, Time, Disruptions
- Misunderstandings
- Reliance on testing
- Overconfidence in own ability
- Unpleasant experiences
  - Management retribution, public ridicule
  - Social conflicts
  - (Criticizing authors not their work)
  - “Who am I to criticize code/look for errors”

# Cost Discussion in Context

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

# 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

# Types of Code Reviews by Formality

- Ad hoc review
- Passaround ("modern code reviews")
- Pair programming
- Wa
- Insp

When to use reviews?
Which formality?

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

# Walkthroughs

- No advance preparation
- Author leads the discussion, presents code
- No formal follow-up
- Low costs
- Valuable for education

# Experience (studies/claims)

- Raytheon
  - 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

# Security Audits

# IsTrueCryptAuditedYet?
# Yes!

**Update Apr 2, 2015:** Phase II complete. TrueCrypt has been audited.

**Update Feb 18, 2015:** Matthew posted an update on the Phase II cryptanalysis today. The Phase I audit report is available on the Open Crypto Audit Project site, and a verified source and download archive for TrueCrypt v. 7.1a can be found on our GitHub mirror. We'll be posting further news @opencryptoaudit on Twitter in the months ahead.

TrueCrypt (TC) is an open source file and disk encryption software package used by people all over the world, but a complete cryptanalysis has not been performed on the software, and questions remain about differences between Windows, Linux and Mac OS X versions. In addition, there has been no legal review on the current TrueCrypt v. 3.0 open source license - preventing inclusion in most of the free operating systems, including Ubuntu, Debian, RedHat, CentOS and Fedora. We want to be able to trust it, but a fully audited, independently verified repository and software distribution would make us feel better about trusting our security to this software. We're pledging this money to sponsor a comprehensive public audit of TrueCrypt.

## Support the Project

You can help support the Project on our FundFill site, or our new IndieGoGo site *(note: both funds accept credit cards; FundFill also accepts Bitcoin, while IndieGoGo also takes PayPal & eChecks)*.

## Goals

- Resolve license status on the current (v. 7.1a) TrueCrypt source code (license v. 3.0 ) copyright & distribution, in order to create a verified, independent version control history repository (signed source and binary)
- Perform and document repeatable, deterministic builds of TC 7.1a from source code for current major operating systems:
  - Windows 7
  - Mac OS X (Lion 10.7 and Mountain Lion 10.8)
  - Ubuntu 12.04 LTS and 13.04, RedHat 6.4, CentOS 6.4, Debian 7.1, Fedora 19
- Conduct a public cryptanalysis and security audit of the TC 7.1a

## Rules

# “Many eyes make all bugs shallow”

Standard Refrain in Open Source

Firefox

src/usr.bin/head/head.c - view - 1.18

cvsweb.openbsd.org/cgi-bin/cvsweb/src/usr.bin/head/head.c?rev=1.18&content-type=text/x-cvsweb-markup

37 year old bug

**Return to head.c CVS log**

File: [local] / src / usr.bin / head / head.c (**download**)

Revision **1.18**, *Wed Oct 8 08:31:53 2014 UTC* (13 days, 4 hours ago) by *schwarze*
Branch: **MAIN**
CVS Tags: **HEAD**
Changes since **1.17: +7 -5 lines**

```
Fix a 37 year old bug introduced by Bill Joy on August 24, 1977
that was already present in the 1BSD release on March 9, 1978
by merging Keith Bostic's 22 year old fix from 4.4BSD (not kidding).

Original CSRG SCCS commit message:

  ^As 00009/00006/00145
  ^Ad D 5.7 92/03/04 14:35:42 bostic 9 8
  ^Ac can't use freopen; example is "date | head file1 /dev/stdin"

ok deraadt@ tedu@, also checked by Martin <Natano dot net>
```

```
/*      $OpenBSD: head.c,v 1.18 2014/10/08 08:31:53 schwarze Exp $      */

/*
 * Copyright (c) 1980, 1987 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
```

The Shellshock vulnerabilities affect **Bash**, a program that various Unix-based systems use to execute command lines and command scripts. Bash is free software, developed collaboratively and overseen since 1992 on a volunteer basis by Chet Ramey, a professional software architect.

Analysis of the source code history of Bash shows the vulnerabilities **had existed undiscovered since version 1.03 in 1989.**

# Inspection vs Static Analysis

# Static Analysis as "Automated Reviews"

- Low-level issues often checked by compiler or static analysis tool
  - Initializing variables; providing correct number of parameters
  - Closing file handles; freeing memory
  - Code style issues
- Root cause analysis -> Build new static checkers
- Enables inspections to focus on important issues

Which rules should ALWAYS be enforced?

# Find the Bug(s)!

```
BlockingQueue queue = …

while (!queue.isEmpty() && ...) {
    CheaterFutureTask Task =
            queue.remove();
    incompleteTasks.add(Task);
    taskValues.add(
        Task.getRawCallable().
            call());
}
```

BatchCommitLogExecutorService.java using BlockingQueue in Cassandra, one bug injected

# 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

# Learning Goals

- Understand different forms of peer reviews with different formality levels
- Select appropriate review forms for a project
- Conduct an inspection session, aware of common pitfalls and social issues
- Perform code reviews with automated software tools
- Understand the expectations and outcomes of modern peer reviews

# 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