Making the Case for Review

Science & Practice

Austin Bingham

@austin_bingham
Please only put bread in the toaster.
If you require any other items heated please ask a host.
# Introduction

What is review, and what is it good for?

<table>
<thead>
<tr>
<th>What do we know about reviews?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A short review of some of the science about reviews</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The benefits of review</th>
</tr>
</thead>
<tbody>
<tr>
<td>How reviews can improve your software</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The perils of review</th>
</tr>
</thead>
<tbody>
<tr>
<td>When reviews attack!</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Introducing reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to start with reviews in your workflow</td>
</tr>
</tbody>
</table>
What is review?

“To view, look at, or look over again.”
- or -
“To inspect, especially formally or officially.”

dictionary.com
What is review?

Some act of looking over the work of yourself or another.
What is review?

Some act of looking over the work of yourself or another.

» Validation
What is review?

Some act of looking over the work of yourself or another.

- Validation
- Learning
What is review?

Some act of looking over the work of yourself or another.

- Validation
- Learning
- Quality check
What is review?

Some act of looking over the work of yourself or another.

- Validation
- Learning
- Quality check
- Whatever!
Improve quality
Improve productivity
Improve teams
Introduction
What is review, and what is it good for?

What do we know about reviews?
A short review of some of the science about reviews

The benefits of review
How reviews can improve your software

The perils of review
When reviews attack!

Introducing reviews
How to start with reviews in your workflow
Reviews can be roughly ordered from formal inspections to *ad hoc*

**Review formality spectrum**

Based on the original diagram by Karl E. Wiegers in *Peer Reviews in Software: A Practical Guide*
Michael Fagan, 1976, IBM

Formal reviews / inspections

Meetings
Roles
Process
Data collection
Metrics
Are meetings really necessary for design reviews?

<table>
<thead>
<tr>
<th>Synergy</th>
<th>Meetings: Teams find faults better than individuals</th>
<th>No Meetings: Meetings tend to find false-positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Education</td>
<td>Less-experienced learn from more-experienced</td>
<td>“Education by observation” is not very effective</td>
</tr>
<tr>
<td>Deadline</td>
<td>Meetings impose a schedule</td>
<td>Deadlines can be imposed without meetings</td>
</tr>
<tr>
<td>Competition</td>
<td>Egos give incentives to contribute/improve</td>
<td>Competition can be achieved without meetings</td>
</tr>
<tr>
<td>Process</td>
<td>“Inspections are part of official process.”</td>
<td>Facts, not tradition, should determine process</td>
</tr>
</tbody>
</table>
Lawrence Votta, 1993, Bell Labs

Are meetings really necessary for design reviews?

4% of defects found in meetings
Largely confirmed Votta’s findings.

<table>
<thead>
<tr>
<th>Reading</th>
<th>Meeting</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.7 defects/hr.</td>
<td>1.2 defects/hr.</td>
</tr>
</tbody>
</table>

Reading is 50% more efficient
Measure impact of reading techniques on UML inspections

Reidar Conradi, 2003, Ericsson Norway/NTNU/Agder Univ.

% Time spent
- Reading: 75%
- Meeting: 25%

% Defects found
- Reading: 80%
- Meeting: 20%
Smartbear, 2006, Cisco

Large study of use of lightweight, tool-driven code review
Smartbear, 2006, Cisco

Large study of use of lightweight, tool-driven code review

- Review size should be under 200, and no more than 400
Smartbear, 2006, Cisco

Large study of use of lightweight, tool-driven code review

- Review size should be under 200, and no more than 400
- Less than 300 LOC/hour for best detection rate
Smartbear, 2006, Cisco
Large study of use of lightweight, tool-driven code review

- Review size should be under 200, and no more than 400
- Less than 300 LOC/hour for best detection rate
- Author preparation/annotation results in far fewer defects
Smartbear, 2006, Cisco
Large study of use of lightweight, tool-driven code review

- Review size should be under 200, and no more than 400
- Less than 300 LOC/hour for best detection rate
- Author preparation/annotation results in far fewer defects
- Total review time should be less than 60 min., not to exceed 90 min.
Smartbear, 2006, Cisco
Large study of use of lightweight, tool-driven code review

- Review size should be under 200, and no more than 400
- Less than 300 LOC/hour for best detection rate
- Author preparation/annotation results in far fewer defects
- Total review time should be less than 60 min., not to exceed 90 min.
- Expect around 15 defects per hour
Smartbear, 2006, Cisco

Large study of use of lightweight, tool-driven code review

- Review size should be under 200, and no more than 400
- Less than 300 LOC/hour for best detection rate
- Author preparation/annotation results in far fewer defects
- Total review time should be less than 60 min., not to exceed 90 min.
- Expect around 15 defects per hour
- Inspection rates can vary widely
Smartbear, 2006, Cisco
Large study of use of lightweight, tool-driven code review

- Review size should be under 200, and no more than 400
- Less than 300 LOC/hour for best detection rate
- Author preparation/annotation results in far fewer defects
- Total review time should be less than 60 min., not to exceed 90 min.
- Expect around 15 defects per hour
- Inspection rates can vary widely

Review between 100 and 300 LOC
Spend 30-60 minutes
Spend at least 5 minutes for even a single-line review
Meeting are good for finding false-positives so keep them short and small
### Cost saving from reviews

As reported in “Peer Reviews in Software”, Wiegers

<table>
<thead>
<tr>
<th>Company</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hewlett-Packard</td>
<td>10:1 ROI, saving $21.4 million per year.</td>
</tr>
<tr>
<td>AT&amp;T Bell Labs</td>
<td>Error-detection cost reduced by a factor of 10. 10-fold quality improvement. 14% productivity increase.</td>
</tr>
<tr>
<td>Bell Northern Research</td>
<td>Prevented 33 hours of maintenance per defect discovered. 2-4x speed detection-time improvement versus testing.</td>
</tr>
<tr>
<td>IBM</td>
<td>1 hour of inspection saved 20 hours of testing and 82 hours of rework (if defect had made it to customers.)</td>
</tr>
<tr>
<td>Imperial Chemical</td>
<td>Maintenance cost for inspected programs was 1/10th of that for uninspected programs.</td>
</tr>
<tr>
<td>Litton Data Systems</td>
<td>3% project effort in inspections reduced testing defects by 30%. Design and code inspections cut integration effort in half.</td>
</tr>
</tbody>
</table>
Upstream inspection is powerful
Finding defects in early phases avoids wasted work in later phases

"Bellcore found that 44 percent of all bugs were due to defects in requirements and design reaching the programmers."

Tom Gilb, Optimizing Software Inspections
Upstream inspection is powerful
Finding defects in early phases avoids wasted work in later phases

"Bellcore found that 44 percent of all bugs were due to defects in requirements and design reaching the programmers."

Tom Gilb, Optimizing Software Inspections

Programmers can do whatever you ask them to do!
"Research study after research study has shown that inspections can detect up to 90% of the errors in a software product before any test cases have been run. And that signifies an extremely effective process."

Robert Glass
"...the same studies show that the cost of inspections is less than the cost of the testing that would be necessary to find the same errors. What we have here is an effective process that is also cost-effective. And that’s a pretty nice combination."

Robert Glass
What about testing?
Frank Blakely et al., 1991, HP

Cost-effectiveness of inspection vs. testing

21 defects found in inspection
Frank Blakely et al., 1991, HP

Cost-effectiveness of inspection vs. testing

21 defects found in inspection

4 would have been found in testing
"Testing alone has never been sufficient to achieve high-quality software."

- Capers Jones
"...software, by its very nature is subject to unknown unknowns. No amount of functional or nonfunctional testing can be designed to detect and correct these problems."

- Capers Jones
Want to know more?
Want to know more?
Want to know more?
Introduction
What is review, and what is it good for?

What do we know about reviews?
A short review of some of the science about reviews

The benefits of review
How reviews can improve your software

The perils of review
When reviews attack!

Introducing reviews
How to start with reviews in your workflow
Defect prevention

Reviews reduce defect injection rates

Generate artifact

Learn

Review artifact
Reviews provide plenty of "teachable moments"

HANDED THE MOST DANGEROUS WEAPON IN THE GALAXY

IMMEDIATELY POINT AT FACE
Monitoring and learning
Reviews allow you to see what others are doing

- Code Quality
- Growth of junior members
- Habits of senior members
- New ideas and techniques
Team cohesion

Shared experience and group ownership
Confidence
Review tools can be helpful for recording decisions.
Defect reduction
Peer reviews are an excellent way to find defects early in your process.

“Peer review catches 60% of the defects.”

Diminish effects of ego

Everybody screws up sometimes!

Spain spent $680 million on submarine that ‘can’t resurface’

Published time: May 22, 2013 16:59
Edited time: May 24, 2013 22:27

Navantia demonstrates the main electric motor for the first S-80 class submarine (Photo: navantia.es)

Personal growth

Review results can reveal patterns and bad practices that you can then fix.
Introduction
What is review, and what is it good for?

What do we know about reviews?
A short review of some of the science about reviews

The benefits of review
How reviews can improve your software

The perils of review
When reviews attack!

Introducing reviews
How to start with reviews in your workflow
Egos

Reviews can also inflame egos if they're perceived as attacks
Developer alienation

Developers need to buy into the review process
Wasted time

Uncritical or shallow reviews cost time and don't improve quality.
Wasted time
Uncritical or shallow reviews cost time and don't improve quality

...it is the rigor (focused attention) with which the inspection team members approach the inspection process that determines how successful the inspection will be, not the use of formality.

Robert Glass
Big Brother effect
It is dangerous to tie review data to employee evaluation

“Tell me how you will measure me, and I will tell you how I behave.”

- Eli Goldratt, “The Goal”
Flow disruption
Reviews can become a distraction
Introduction
What is review, and what is it good for?

What do we know about reviews?
A short review of some of the science about reviews

The benefits of review
How reviews can improve your software

The perils of review
When reviews attack!

Introducing reviews
How to start with reviews in your workflow
Management support

Old status quo → Chaos → Practice & Integration → New status quo

Foreign element
Transforming idea

Satir Change Model
Selling reviews to management

Speak their language
This would be easier with emacs.
Make results tangible
Don't be too disruptive

“People hate change...
and that’s because people hate change...
I want to be sure that you get my point.
People really hate change.
They really, really do.”

But be disruptive enough!
Get everyone involved quickly
In Practice: Where to start?
In Practice: Where to start?

- Code reviews are the most obvious
In Practice: Where to start?

- Code reviews are the most obvious
- But start with what makes sense for you!
In Practice: Where to start?

- Code reviews are the most obvious
- But start with what makes sense for you!
- Increase coverage organically
In Practice: Maintenance
In Practice: Maintenance

› Vigilance!
In Practice: Maintenance

‣ Vigilance!
‣ Emphasize the benefits
In Practice: Maintenance

- Vigilance!
- Emphasize the benefits
- Avoid excessive ceremony
In Practice: What's in a review?

The only valid measurement of code quality: WTFs/minute

good code

code review

bad code

code review

dude, WTF

(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics
In Practice: What's in a review?

- The only valid measurement of code quality: WTFs/minute
- At least one competent reviewer

(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics
In Practice: What's in a review?

The only valid measurement of code quality: WTFs/minute

- At least one competent reviewer
- Early feedback with opportunity for followup
In Practice: What's in a review?

- At least one competent reviewer
- Early feedback with opportunity for followup
- Review before “committing”
In Practice: What's in a review?

- At least one competent reviewer
- Early feedback with opportunity for followup
- Review before “committing”
- Reviewer can block commit

The only valid measurement of code quality: WTFs/minute

(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics
In Practice: What's in a review?

- At least one competent reviewer
- Early feedback with opportunity for followup
- Review before “committing”
- Reviewer can block commit
- Author has final say on commit
References


“Peer Reviews in Software: A Practical Guide”, Karl E. Wiegers

"Facts & Fallacies of Software Engineering", Robert Glass

“Peopleware”, Tom DeMarco, Tim Lister

“The Goal”, Eli Goldratt

“Software Defect Reduction Top 10 List”, Barry Boehm, Victor R. Basili


http://svenpet.com/2014/01/07/better-code-reviews/

http://phinze.github.io/2013/12/08/pairing-vs-code-review.html
Thank you!

Austin Bingham
@austin_bingham

SixtyNORTH
@sixty_north