EFFECTIVE CODE REVIEW

EFFECTIVE CODE REVIEW

Who am I?

redhat.

@d0ugal

Raise your hand...

Not doing code review?

“the average defect detection rate is only 25 percent for unit testing, 35 percent for function testing, and 45 percent for integration testing. In contrast, the average effectiveness of design and code inspections are 55 and 60 percent”

Code Complete by Steve McConnell

“The only hurdle to a code review is finding a developer you respect to do it, and making the time to perform the review. Once you get started, I think you'll quickly find that every minute you spend in a code review is **paid back tenfold**.”

Jeff Atwood (Coding Horror)

“Formal design and code inspections […] often top 85 percent in defect removal efficiency and average about 65 percent”

Measuring Defect Potentials and Defect Removal Efficiency

Code Review Goals

Expectation vs Outcome

“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.”

Expectations, Outcomes, and Challenges Of Modern Code Review

Comment Outcomes

1. Code Improvements (29%)
2. Understanding
3. Social Communications
4. Defects (14%)
5. External Impact
6. Testing
7. Review Tool
8. Knowledge Transfer
9. Misc

Authors

Reviewers

Authors

VS

Reviewers

Code ~~Review~~

Code Discussion
Code Collaboration
...

Authors & Reviewers

Authoring Changes

Don’t start with code!

https://flic.kr/p/cexrh1

Adhere to Project Guidelines

Write test.
Write documentation.
Test the relevant platforms.
Follow the Style guide.

Provide Context

https://flic.kr/p/nZpgc6

Small & Contained

“code review:
10 LOC - 9 issues,
500 LOC - looks fine”

Mikhail Garber
(@mikhailgarber)

“Its regression coefficients are positive, indicating that larger patches lead to a higher likelihood of reviewers missing some bugs. Similarly, number of files has a good explanatory power in all four systems.”

Investigating Code Review Quality:
Do People and Participation Matter?

Opening a Review is the start

Start of the conversation

Don’t ask for it to be merged, ask for it to be reviewed

Relinquish Ownership

“0% thankfully. Coders act like they've painted a masterpiece and tend to debate every piece of feedback.”

Mark Litwintschik
(@marklit82)

</Authoring Changes>

Code Review is hard.

Reviewing Changes

Shared Responsibility

Contributions == Puppies

Everyone Reviews

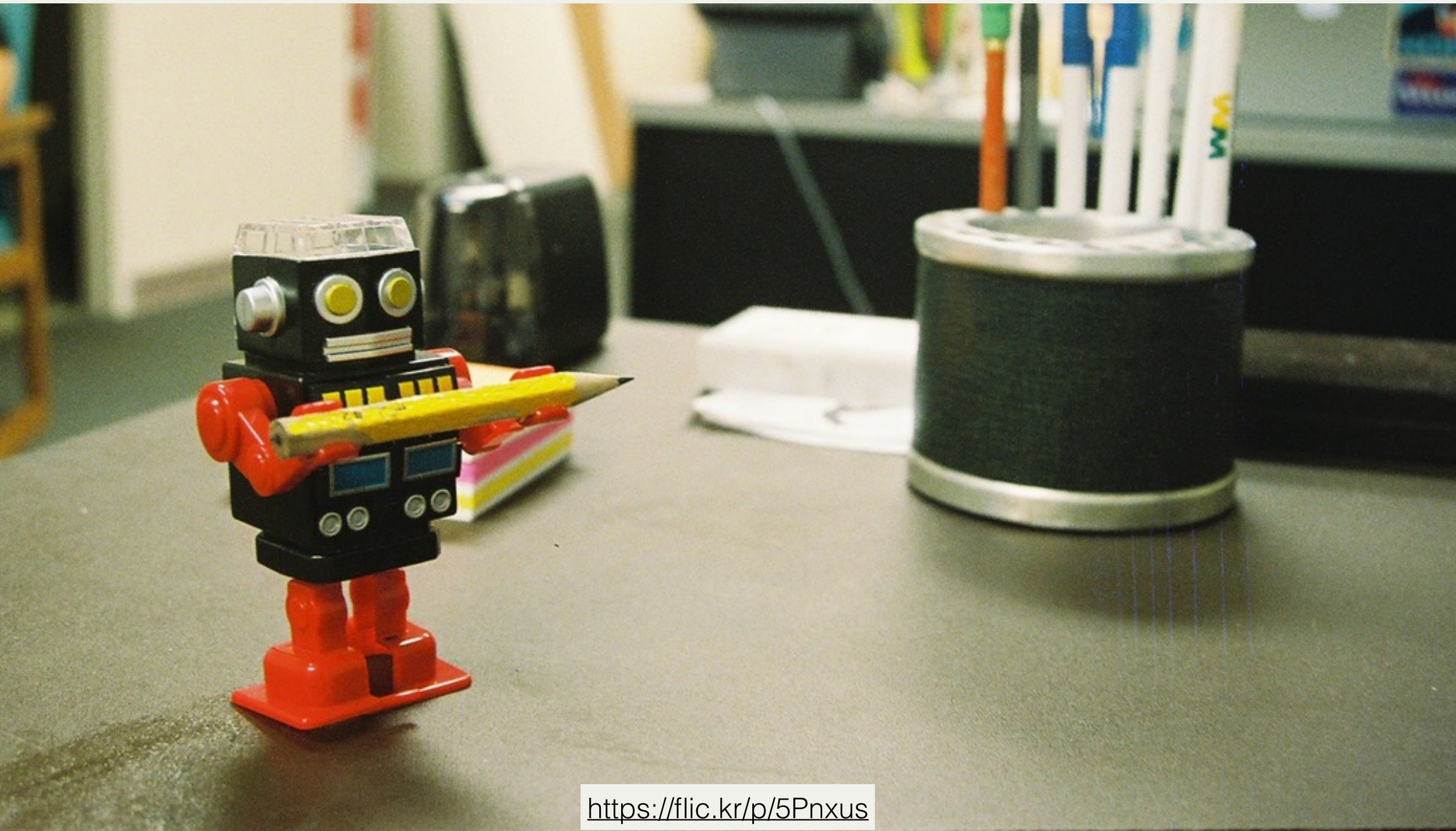
Juniors. Seniors.

Review to learn, verify and teach. Not necessarily in that order.

Keep reviewers on the same page

If they are all reviewing to different rules, it will never make sense

Automation



https://flic.kr/p/5Pnxus

Remove the Bikeshed

https://flic.kr/p/8qqMca

Multiple Reviewers

Frequent, Short Reviews

https://flic.kr/p/atDNLR

Constructive criticism and Praise

It’s easy to just point out the bad things, but when somebody teaches you something - “I didn’t know you could do that!” moments - let them know.

Be Polite and aware of tone

Some things can come across overly negative.

“Why didn’t you do …?”

Sounds more negative written than in person. Replace with

“Could we do this …?”

Never harsh. Never Personal

https://flic.kr/p/efcTcb

</Reviewing Changes>

Writing Code is hard.

Collaboration

Help each other.
Automate what you can.
Be kind to yourself.

Tooling

GitHub? Gerrit? Phabricator?
GitLab? Review Board?

Review Before The Merge

GitHub

Loose workflow. Labels are useful.

Simple UI.

Gerrit

Very defined. Multiple reviewers.

Code Review Data

Questions?

twitter.com/d0ugal

github.com/d0ugal

dougal@dougalmatthews.com

(Sort-of related; OpenStack Open Space tomorrow afternoon)