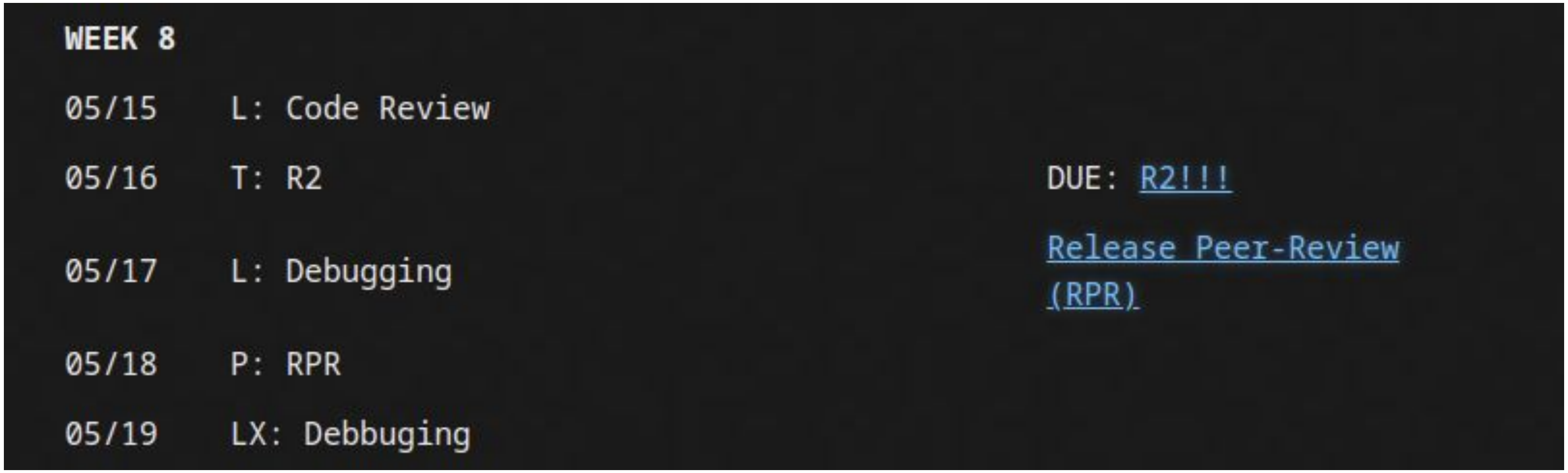
CSE 403

Software Engineering

Spring 2023

#22: Code Review

Logistics



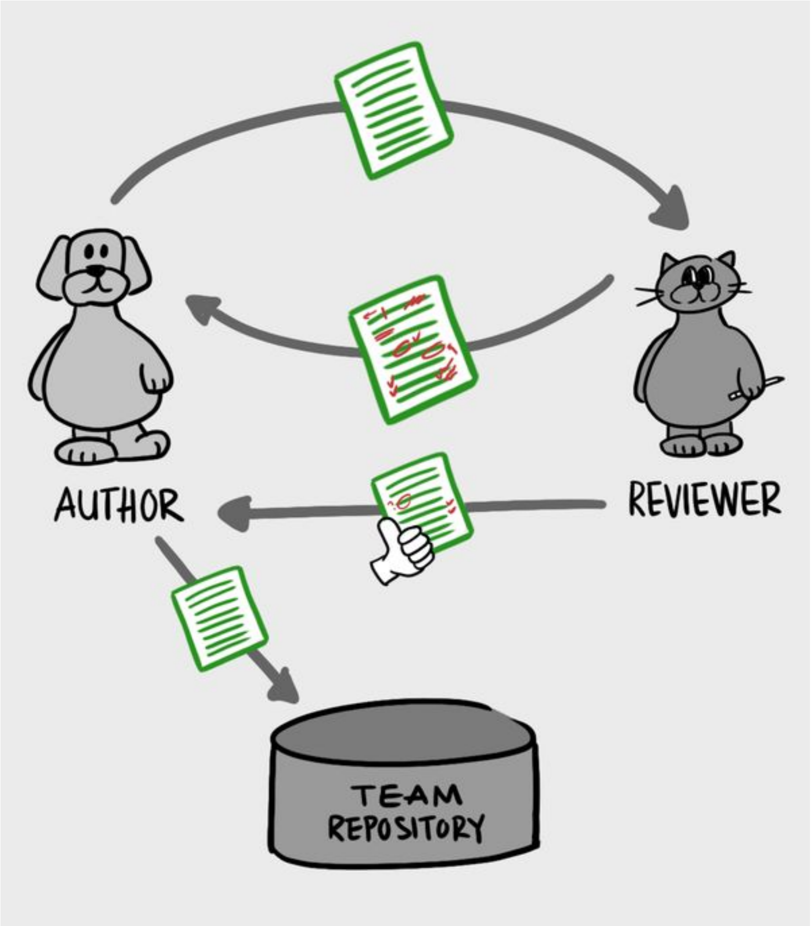
WEEK 8

05/15	L: Code Review	
05/16	T: R2	DUE: R2!!!
05/17	L: Debugging	Release Peer-Review (RPR)
05/18	P: RPR	
05/19	LX: Debbuging	

Code Review: What?

"A code review is a process where someone other than the author(s) of a piece of code examines that code."

https://google.github.io/eng-practices/review/



Code Review: Why?

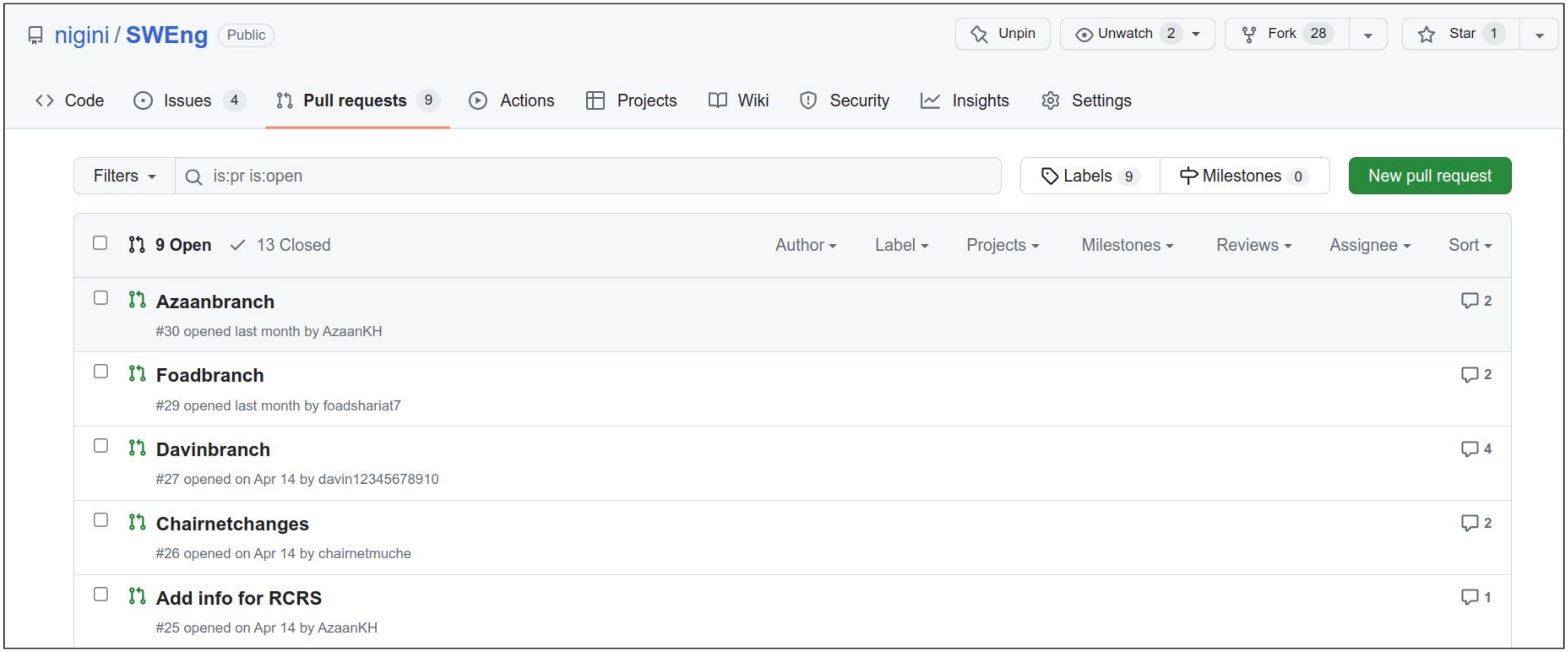
"The primary purpose of code review is to make sure that the overall code health of Google’s code base is improving over time."

https://google.github.io/eng-practices/review/

"It is a learning, social, and bonding activity among teammates. Focusing on bug finding is missing the point!"

https://mtlynch.io/human-code-reviews-1/

Code Review: What - Example



□ 9 Open ✓ 13 Closed

□ **Azaanbranch** 2
#30 opened last month by AzaanKH

□ **Foadbranch** 2
#29 opened last month by foadshariat7

□ **Davinbranch** 4
#27 opened on Apr 14 by davin12345678910

□ **Chairnetchanges** 2
#26 opened on Apr 14 by chairnetmuche

□ **Add info for RCRS** 1
#25 opened on Apr 14 by AzaanKH

https://github.com/nigini/SWEng/pulls

Code Review: Why (expanded)

- evolve code quality
- team building
- meet standards
- find bugs
- sharing knowledge
- check code understandability
- ensure the code does what it supposes to do
- collaborate on design
- …

Code Review: Practice

- Go to your team's repository commit list
- Find a commit made by someone else
  - with substantial change (50-100 lines?)
  - to a "component" you know less about
- Go to your Slack team channel and start a thread called **"I am reviewing commit XYZ"** (where XYZ is a link to the commit)

Answer the following questions…

Be careful: The commit author is also a human!!!

Code Review: Practice

- Go to your team's repository commit list
- Find a commit made by someone else
  - with substantial change (50-100 lines?)
  - to a "component" you know less about
- Go to your Slack team channel and start a thread called **"I am reviewing commit XYZ"** (where XYZ is a link to the commit)

Answer the following questions…

Quick Example!

Code Review: Practice

Answer some of these questions…

- What functionality does it provide to the user?
- Is this commit easy to understand? Why?
- Any part of the code that doesn't follow the code style?
- What is something positive about this code?
- Can the code be simplified?
- What have you learned about your project by studying this commit?
- Another positive thing about this code?

Code Review: When?

What do you think?

- during coding?
- after coding?
- after deployment?

Code Review: When?

- during coding? (ex.: Is this a good design?)
- after coding? (ex.: Does this code deliver?)
- after deployment? (ex.: Can I learn/improve something here?)

Depends on your review goals!

Code Review: When (does it end?)

What do you think?

- gatekeeper thumbs-up?
- reach agreement?
- good enough?

Code Review: When (does it end?)

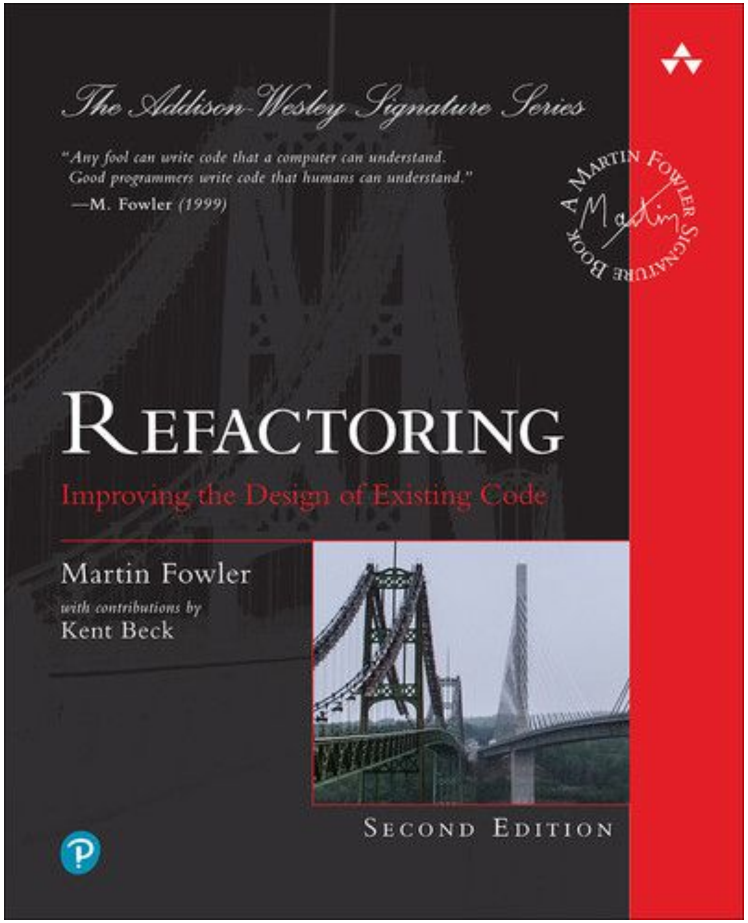
What do you think?

- gatekeeper thumbs-up?
- reach agreement?
- **good enough?**

"In general, reviewers should favor approving a CL once it is in a state where it definitely improves the overall code health of the system being worked on, even if the CL isn’t perfect." - Google's Standard

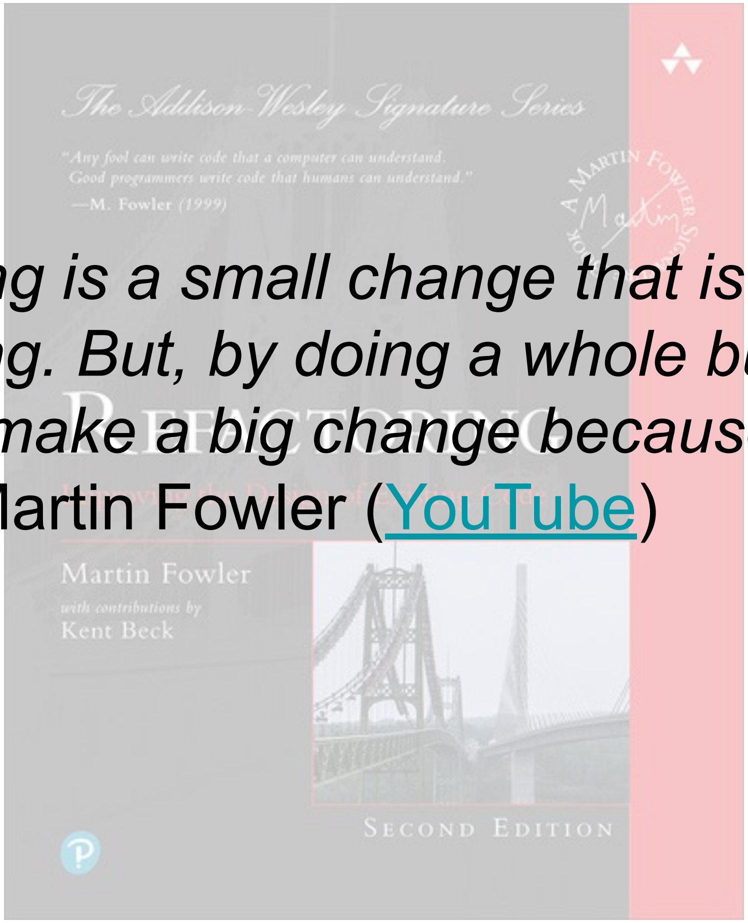
Depends on your review goals! But…

Code Review: Design & Refactoring


The Addison-Wesley Signature Series
"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."
—M. Fowler (1999)
A Martin Fowler Signature Book
Martin
Refactoring
Improving the Design of Existing Code
Martin Fowler
with contributions by
Kent Beck
Second Edition

Chap 3: Code Smells

Code Review: Design & Refactoring


The Addison-Wesley Signature Series
"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."
—M. Fowler (1999)
A Martin Fowler Signature Book
Martin
Refactoring
Martin Fowler
with contributions by
Kent Beck
Second Edition

"Every refactoring is a small change that is so small that it is not worth doing. But, by doing a whole bunch of them, you are able to make a big change because they compose really well." — Martin Fowler (YouTube)

Chap 3: Code Smells

Code Review: Summary - Take #1

Make sure that:

- The code is well-designed.
- The functionality is good for the users of the code.
- Any UI changes are sensible and look good.
- Any parallel programming is done safely.
- The code isn't more complex than it needs to be.
- The developer isn't implementing things they *might* need in the future but don't know they need now.
- Code has appropriate unit tests.
- Tests are well-designed.
- The developer used clear names for everything.
- Comments are clear and useful, and mostly explain *why* instead of *what*.
- Code is appropriately documented (generally in g3doc).
- The code conforms to our style guides.

Make sure to **review every line of code** you've been asked to review, look at the context, make sure you're **improving code health**, and **compliment developers** on good things that they do.

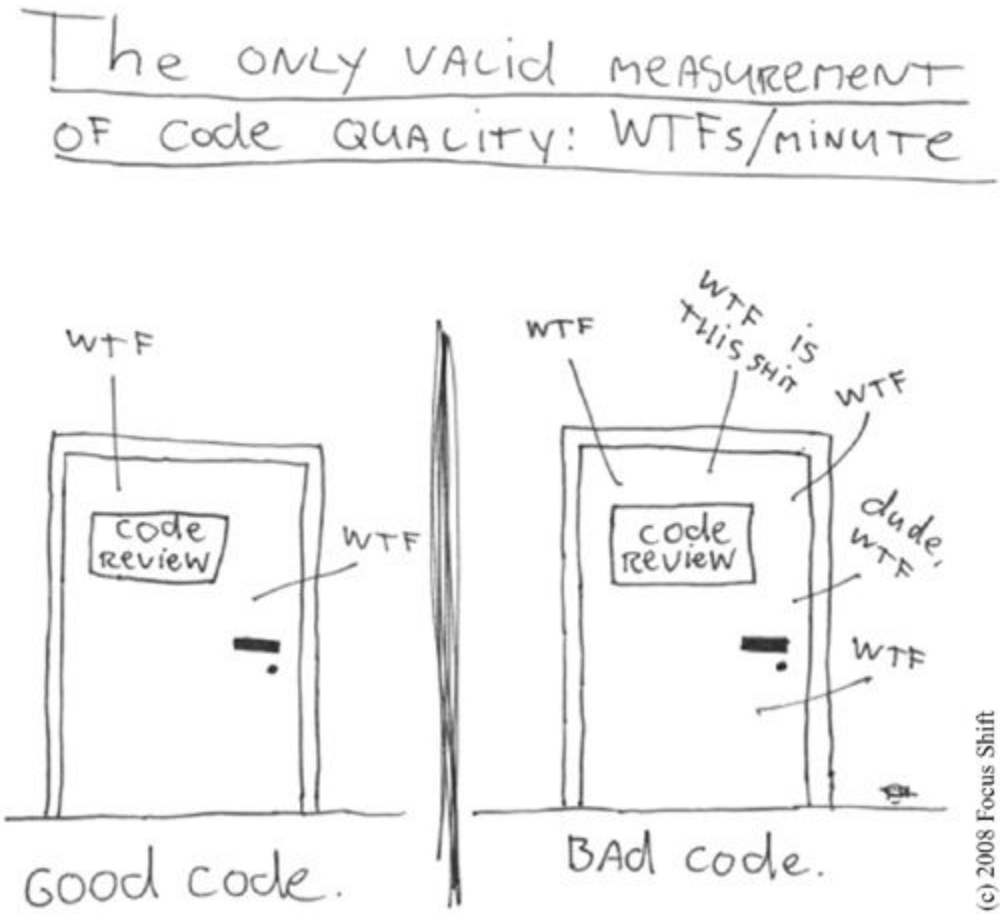
https://google.github.io/eng-practices/review/reviewer/looking-for.html

Code Review: Summary - Take #2

- Define a style guide as a team
- Let computers do the boring parts: linters/formatters (and CI)
- Give code examples instead of “possible change requests” (build trust)
- Never say “YOU” (focus on the code, not the coder!): we == team ownership
- Requests not command… frame it as an in-person conversation
- Add sincere positive praises
- Incremental improvements instead of perfection
- Handle stalemates proactively: talk it out, design review?, concede or escalate

How to Do Code Reviews Like a Human (video)

Code Review…



Reproduced with the kind permission of Thom Holwerda.
http://www.osnews.com/story/19266/WTFs_m

Questions, please!