PROCESS IN SE

Rohan Padhye and **Michael Hilton**

LEARNING GOALS

Understand process aspects of QA

Describe the tradeoffs of QA techniques

Select an appropriate QA technique for a given project and quality attribute

Decide the when and how much of QA

Overview of concepts how to enforce QA techniques in a process

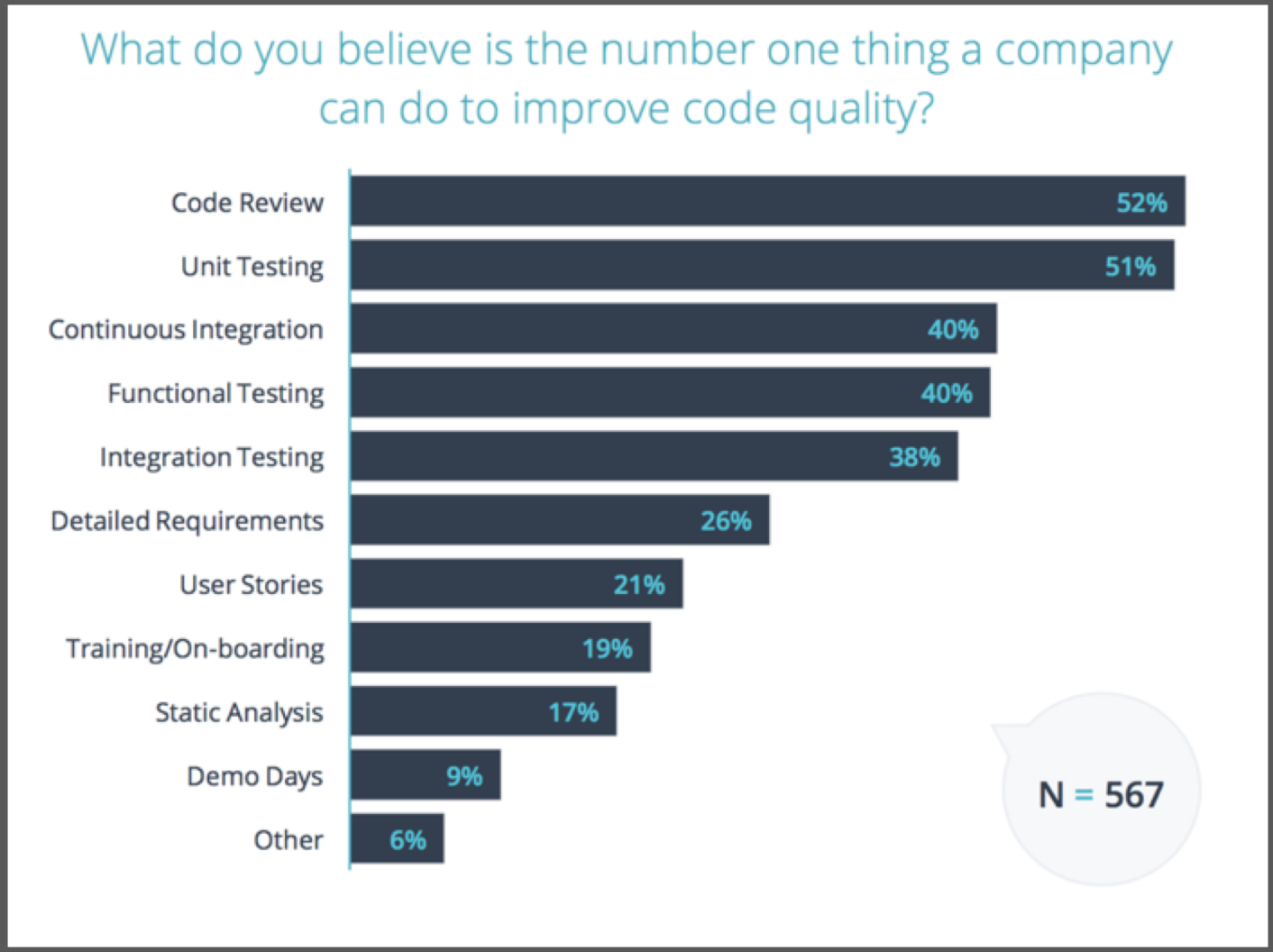
Understand human and social challenges of adopting QA techniques

Understand how process and tool improvement can solve the dilemma between features and quality

PROCESS IN SE

PROCESS ROUNDUP

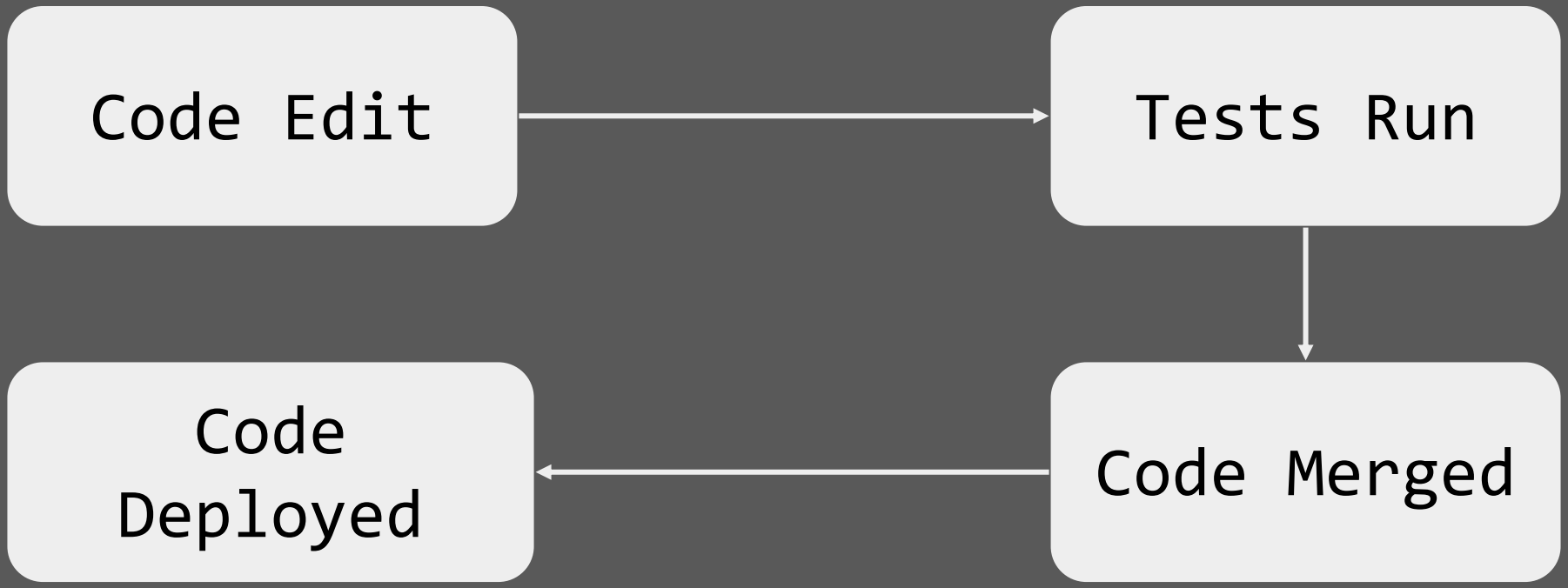
- Group Processes
  - Continuous Integration
  - Code Review
  - Pair/Mob Programming
- Individual Processes
  - Asking Questions
  - How to run a meeting



[State of Code Review 2017]

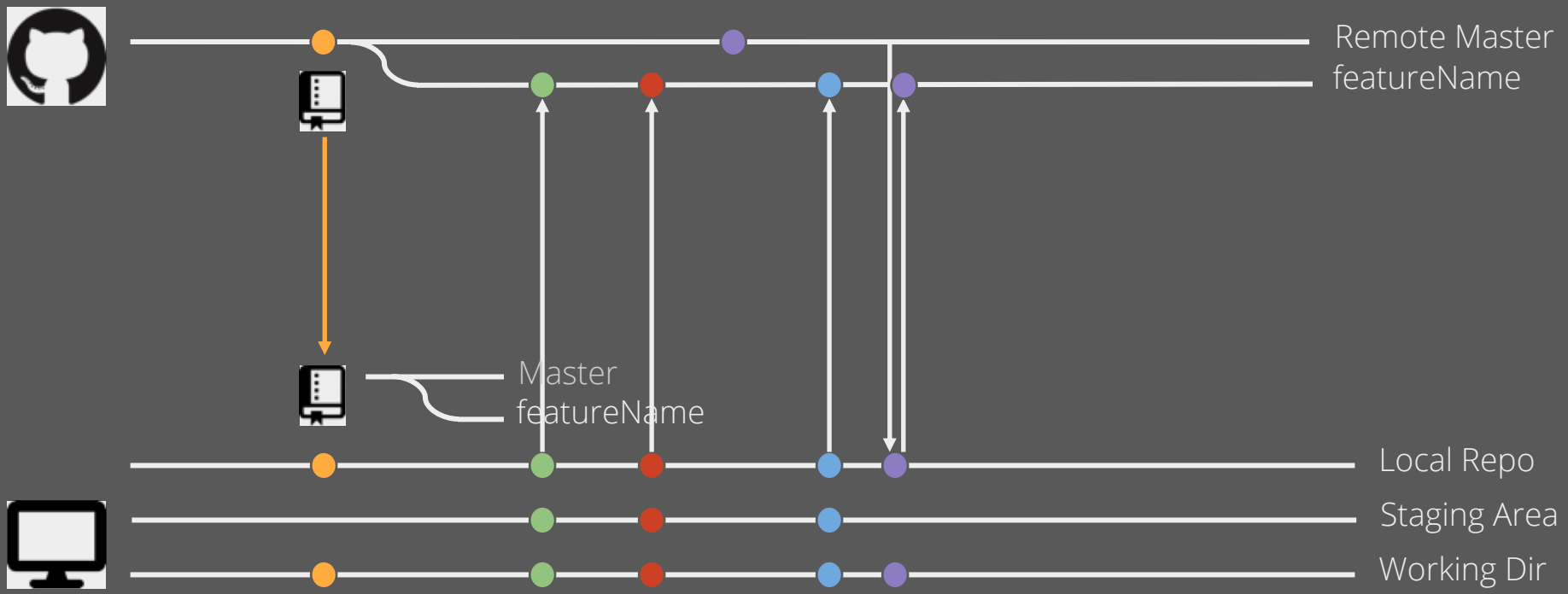
CONTINUOUS INTEGRATION

CI/CD PIPELINE OVERVIEW



GITHUB FLOW

REVIEW: GITHUB FLOW



CONTINUOUS INTEGRATION:

HISTORY OF CI

(1999) Extreme Programming (XP) rule: “Integrate Often”

(2000) Martin Fowler posts “Continuous Integration” blog

(2001) First CI tool

(2005) Hudson/Jenkins

(2011) Travis CI

(2019) GitHub Actions

SAMPLE CI WORKFLOW

Create Pull Request

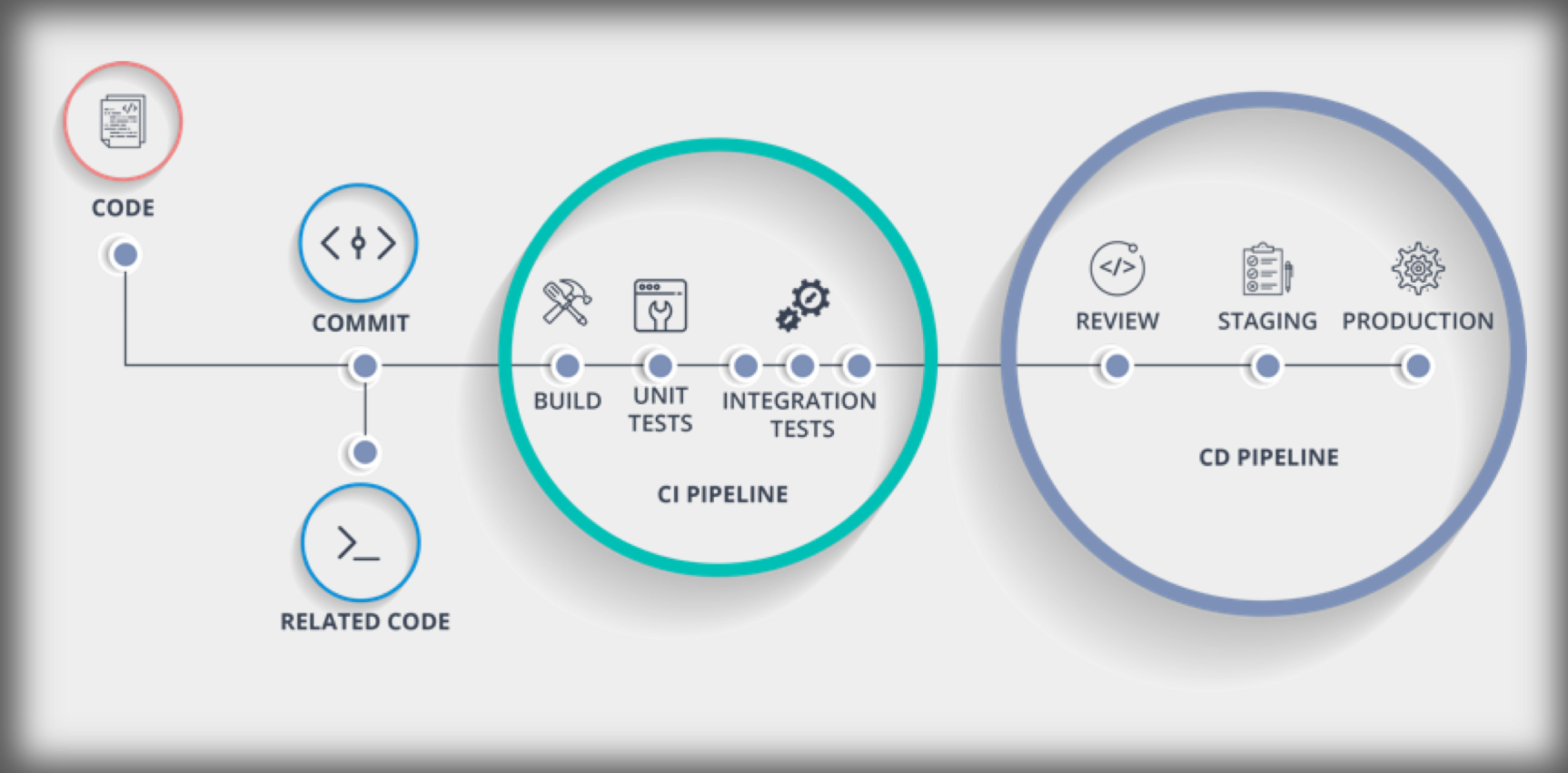
GitHub tells Travis CI build is mergeable

It builds and passes tests

Travis updates PR

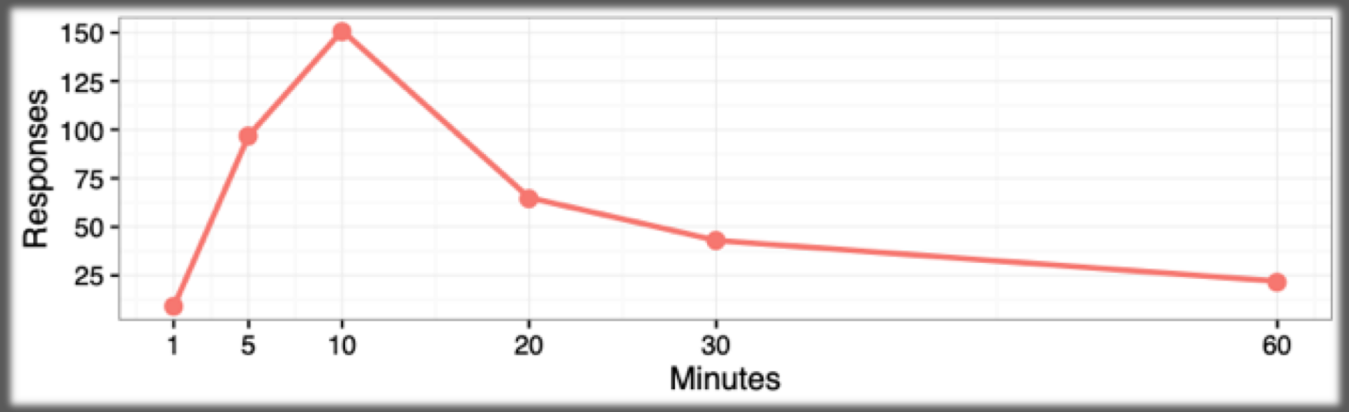
PR is merged

EXAMPLE CI/CD PIPELINE



CI RESEARCH

"My favorite way of thinking about build time is basically, you have tea time, lunch time, or bedtime..."



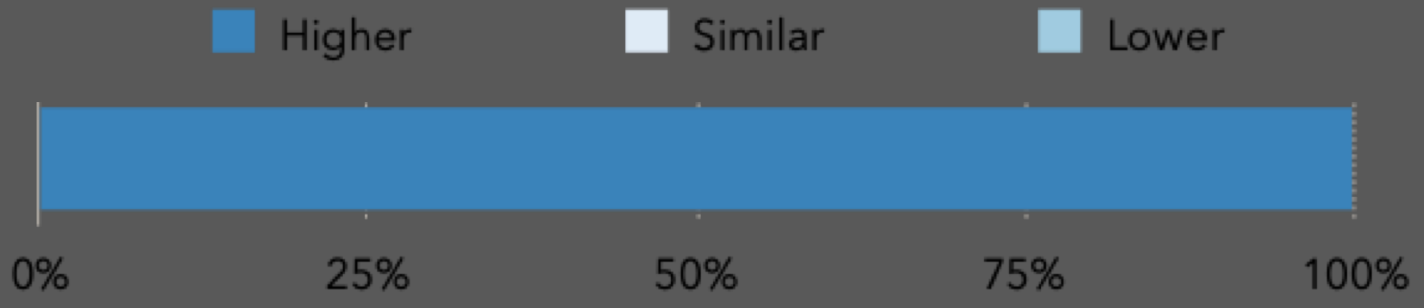
DEVELOPERS SAY:

CI helps us catch bugs earlier
CI makes us less worried about breaking our builds
CI lets us spend less time debugging

"[CI] does have a pretty big impact on [catching bugs]. It allows us to find issues even before they get into our main repo, … rather than letting bugs go unnoticed, for months, and letting users catch them."

DEVELOPERS REPORT:

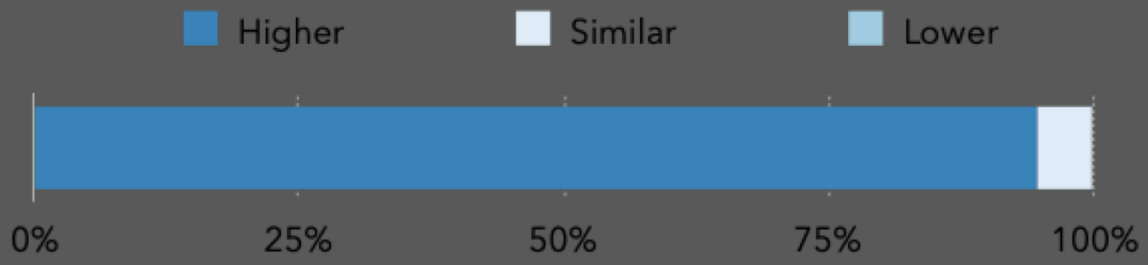
Do developers on projects with CI give (more/similar/less) value to automated tests?



DEVELOPERS REPORT:

Do developers on projects with CI give (more/similar/less) value to automated tests?

Do projects with CI have (higher/similar/lower) test quality?

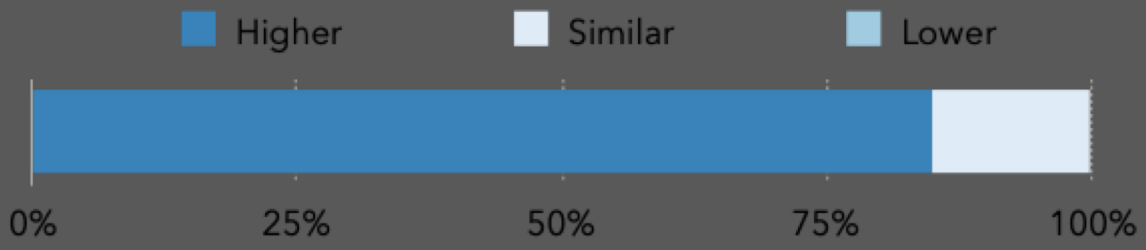


DEVELOPERS REPORT:

Do developers on projects with CI give (more/similar/less) value to automated tests?

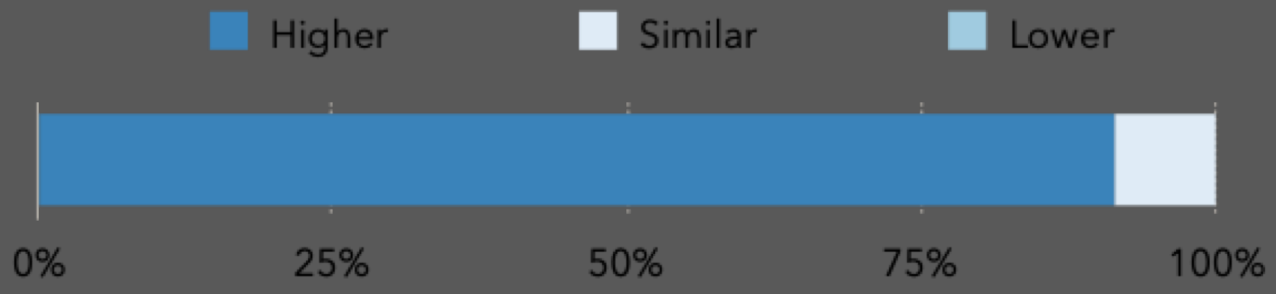
Do projects with CI have (higher/similar/lower) test quality?

Do projects with CI have (higher/similar/lower) code quality?

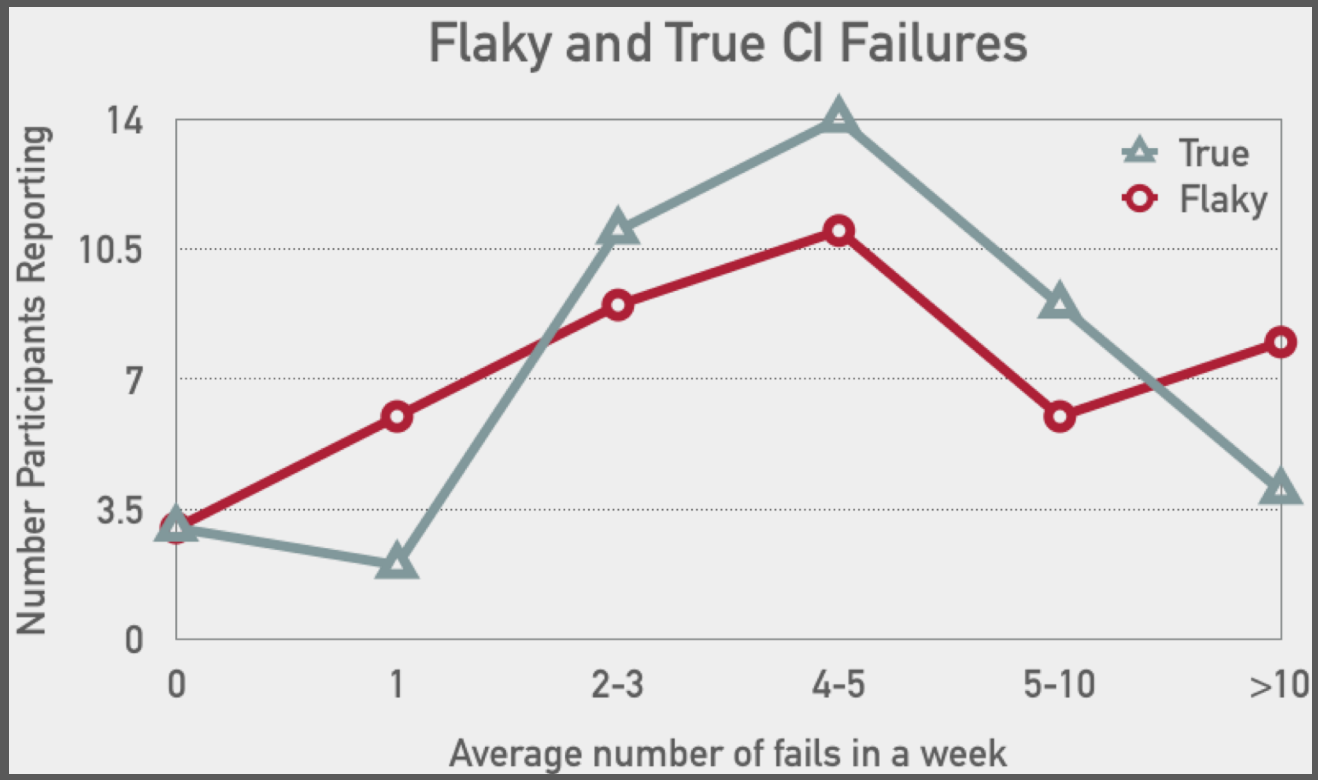


DEVELOPERS REPORT:

Do developers on projects with CI give (more/similar/less) value to automated tests?
Do projects with CI have (higher/similar/lower) test quality?
Do projects with CI have (higher/similar/lower) code quality?
Are developers on projects with CI (more/similar/less) productive?



CHALLENGE: FLAKY TESTS



OBSERVATION

Most of the benefits of CI come from running tests

TESTS AND QA: COMING SOON!

CODE REVIEW

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

“Many eyes make all bugs shallow”

Standard Refrain in Open Source

“Have peers, rather than customers, find defects”

Karl Wiegers

EXPECTATIONS AND OUTCOMES OF MODERN CODE REVIEWS

institute for SOFTWARE RESEARCH | Carnegie Mellon University School of Computer Science

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

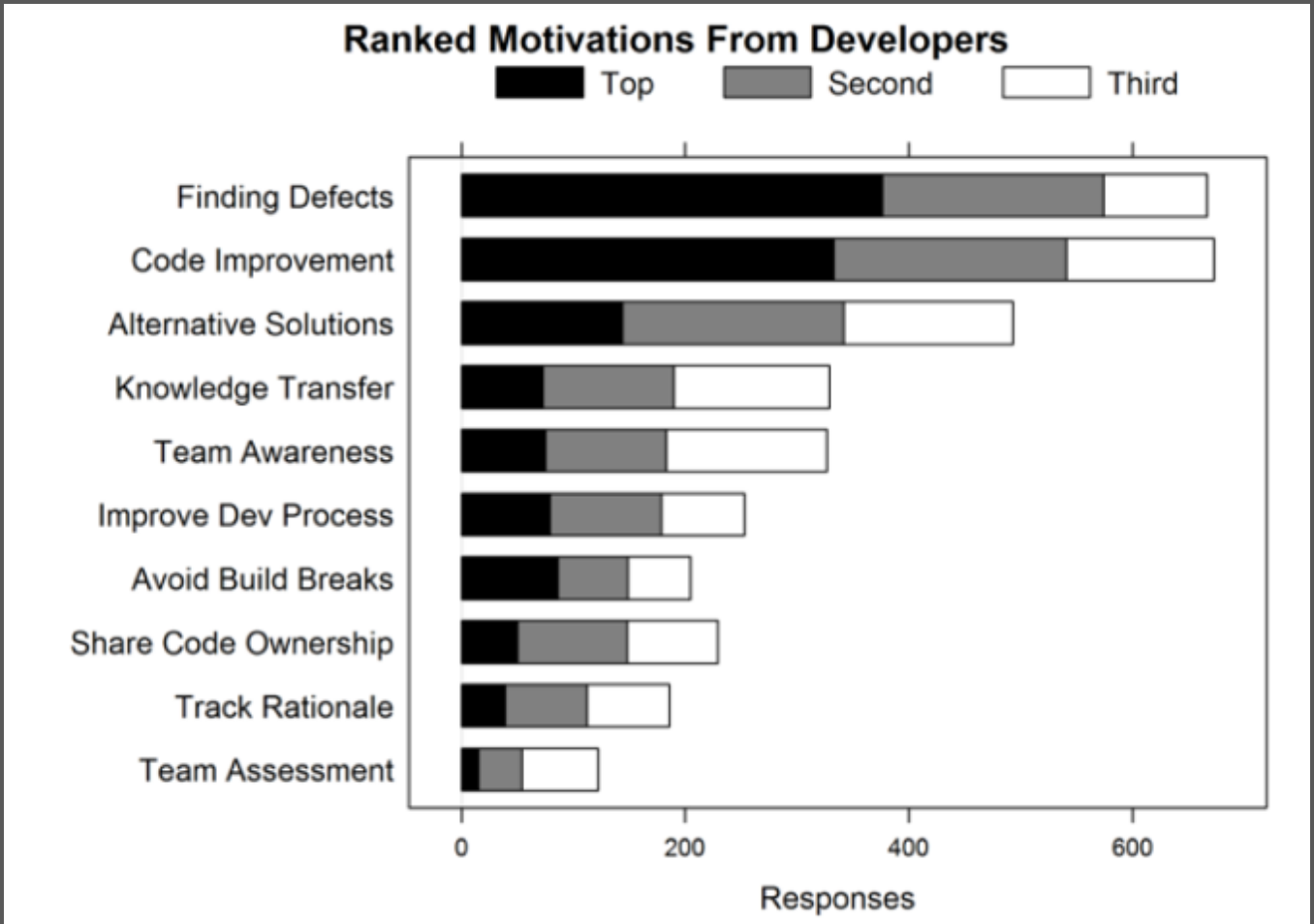
Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.

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

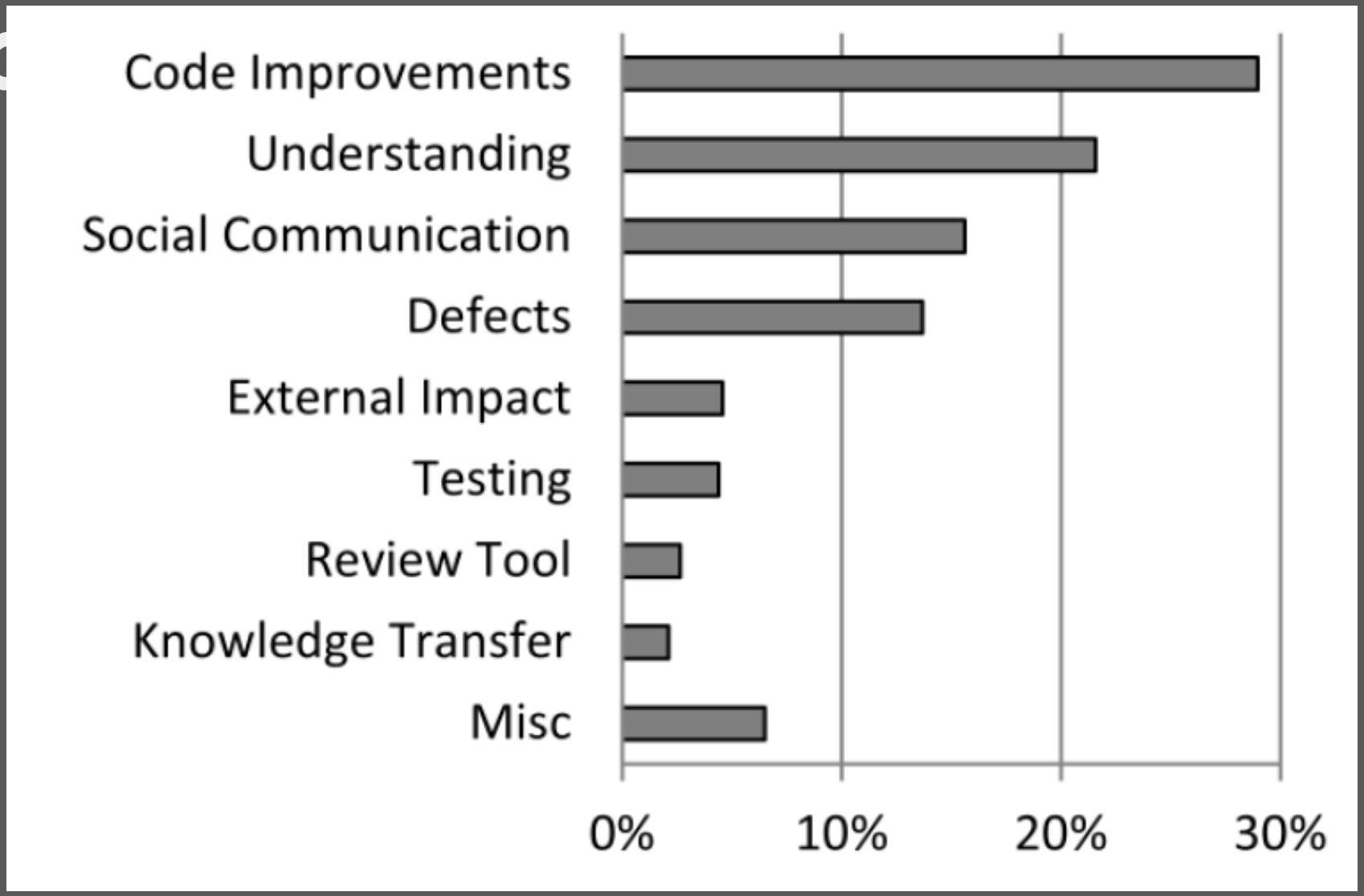
Bacchelli, Alberto, and Christian Bird. "Expectations, outcomes, and challenges of modern code review." *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013.

CODE REVIEW AT MICROSOFT



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


Code Improvements
Understanding
Social Communication
Defects
External Impact
Testing
Review Tool
Knowledge Transfer
Misc
0%
10%
20%
30%

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

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

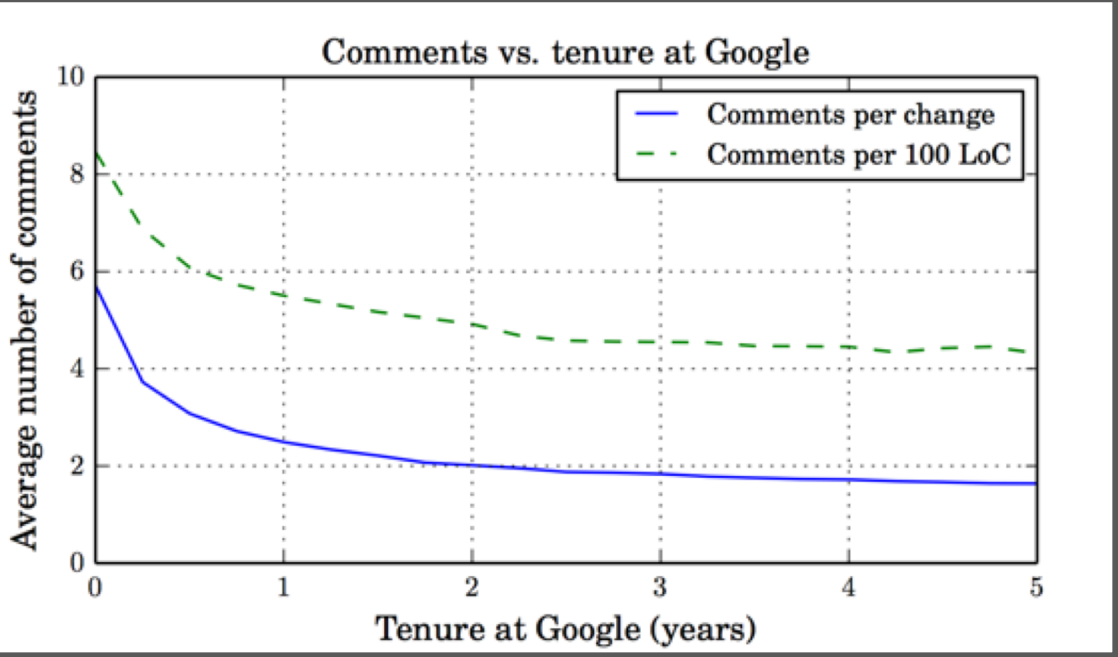
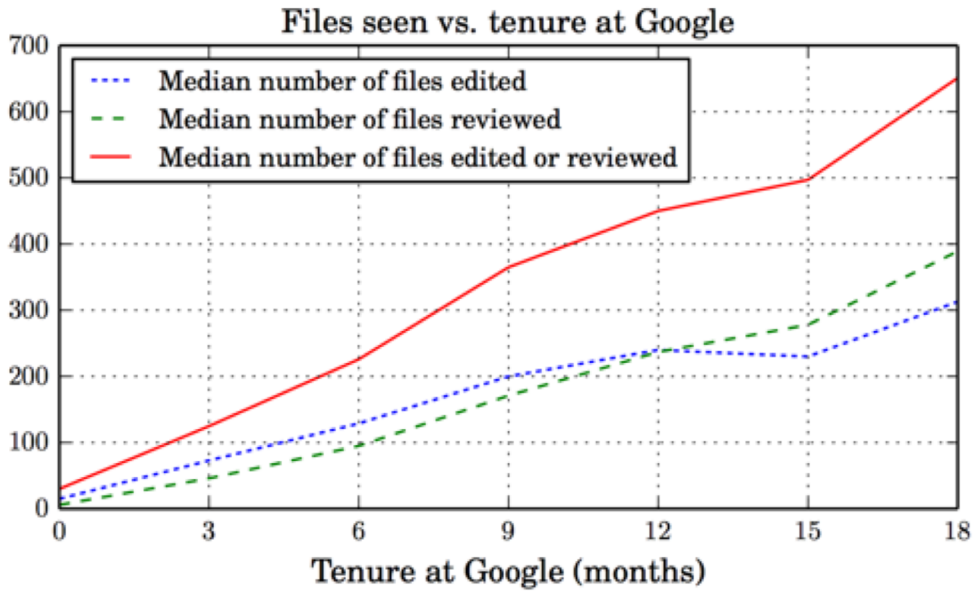
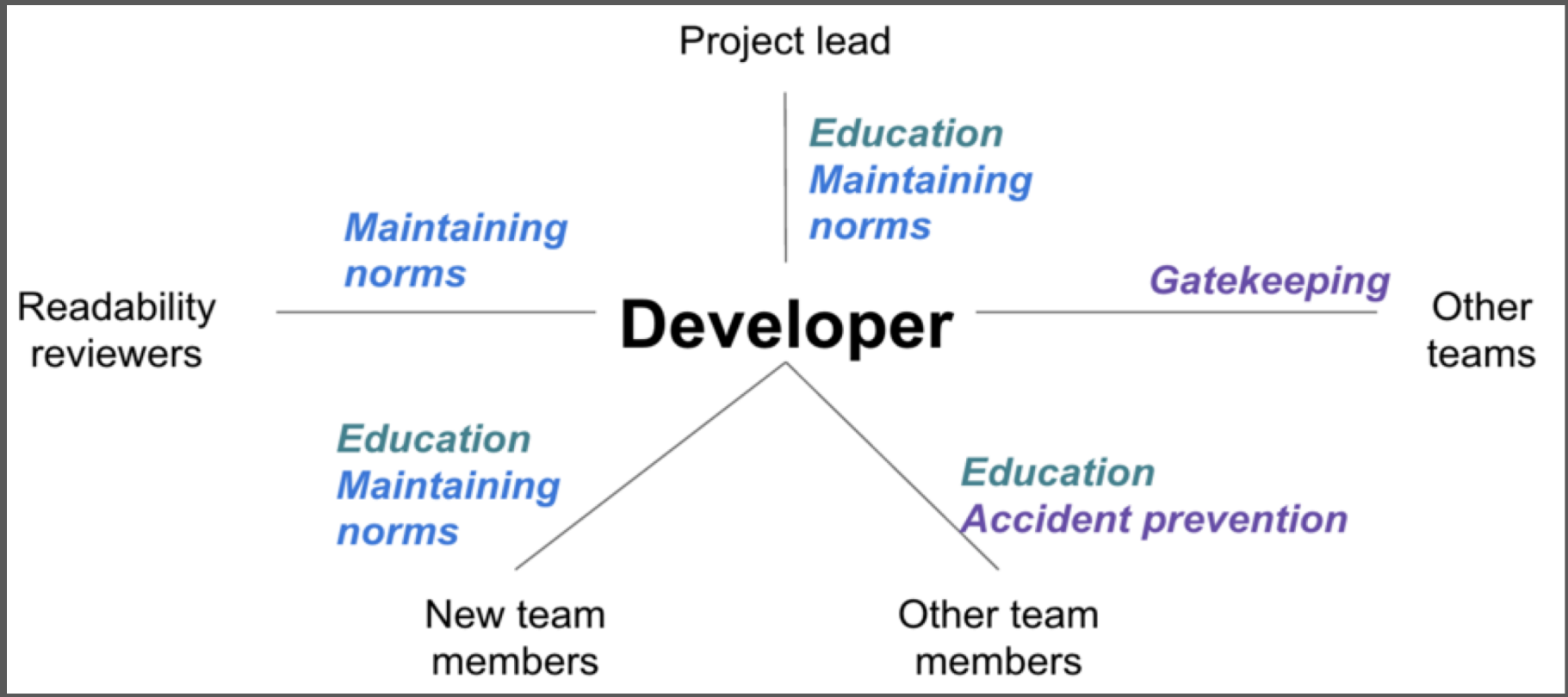


Figure 2: Reviewer comments vs. author's tenure at Google



REVIEWING RELATIONSHIPS



GOOGLE VS. MICROSOFT?

Observations?

DON'T FORGET DEVS ARE HUMANS TOO

- Author's self-worth in artifacts
- CI can avoid embarrassment
- 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

CODE REVIEW

Let computers do the parts they are good at,

Let the humans focus on the parts they are good at.

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-1 or AN 01-20EG-1 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

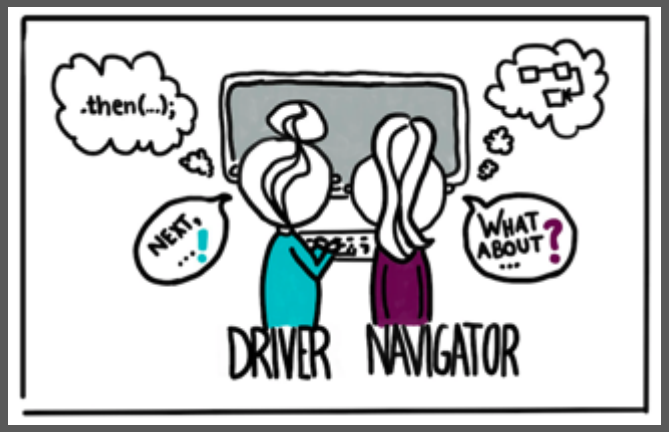
DEVELOP CHECKLIST FOR CODE REVIEW

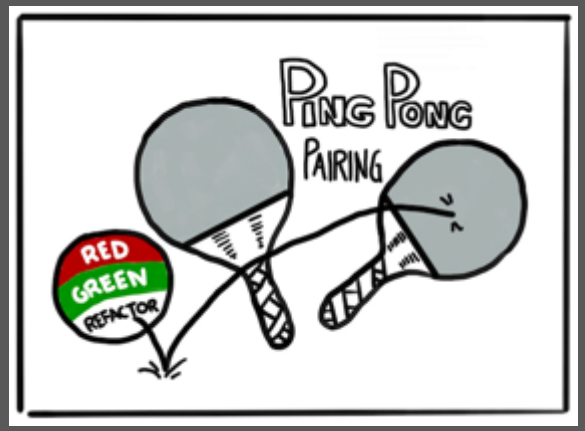
institute for SOFTWARE RESEARCH | Carnegie Mellon University School of Computer Science

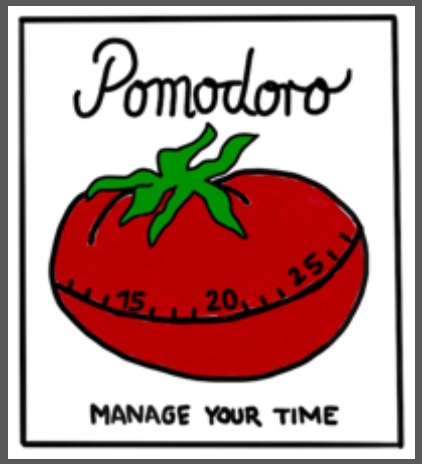
PAIR/MOB PROGRAMMING

institute for SOFTWARE RESEARCH | Carnegie Mellon University School of Computer Science

PAIR PROGRAMMING

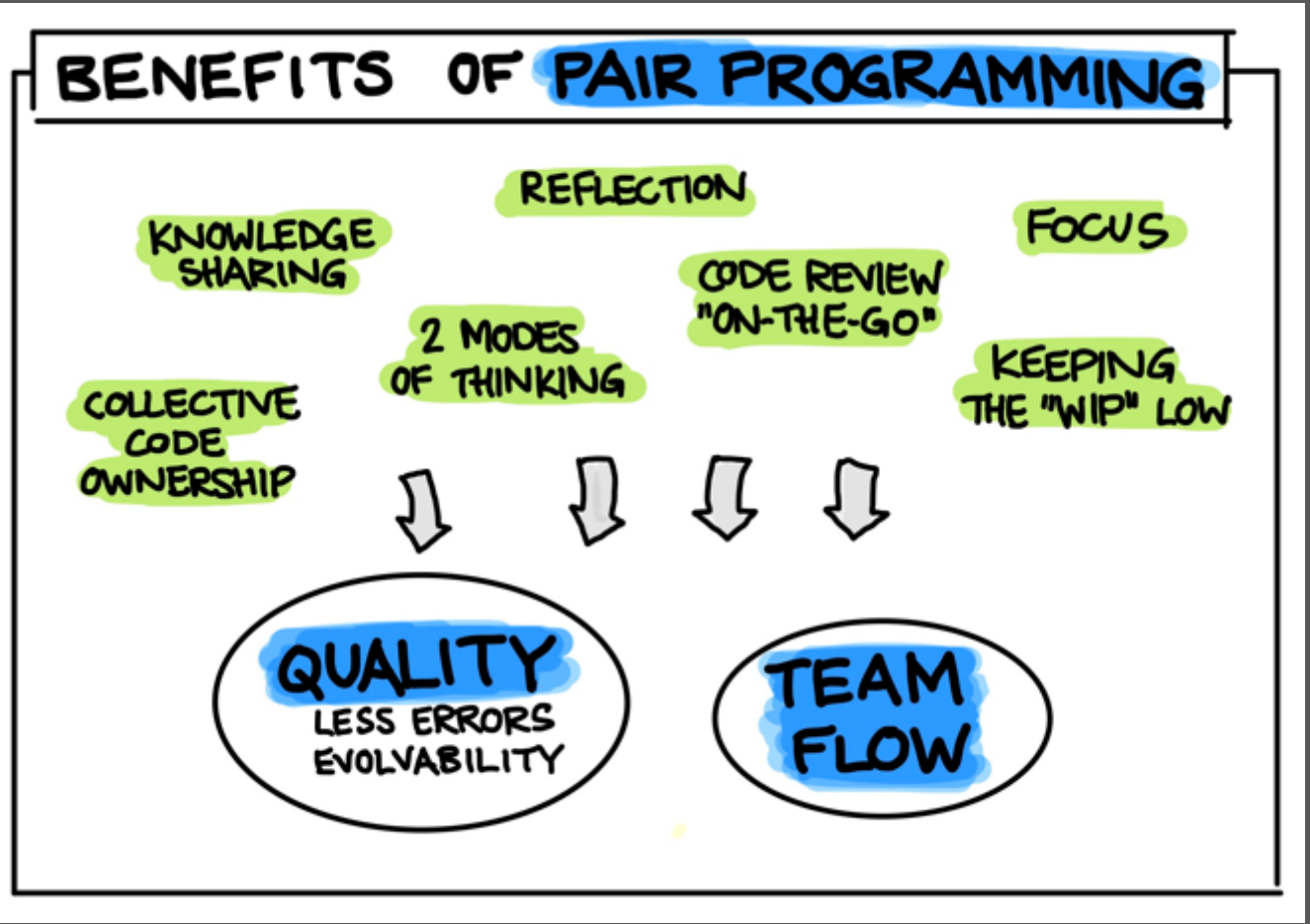






https://martinfowler.com/articles/on-pair-programming.html

BENEFITS


BENEFITS OF PAIR PROGRAMMING
REFLECTION
KNOWLEDGE SHARING
FOCUS
CODE REVIEW "ON-THE-GO"
2 MODES OF THINKING
KEEPING THE "WIP" LOW
COLLECTIVE CODE OWNERSHIP
QUALITY
LESS ERRORS
EVOLVABILITY
TEAM FLOW

MOB PROGRAMMING

All the brilliant people working on the same thing, at the same time, in the same space, on the same computer.

– Woody Zuill (the discoverer of Mob Programming)

https://dev.to/albertowar/mob-programming-revisited-2fo4

ITS NOT ABOUT GETTING THE MOST OUT OF YOUR TEAM, ITS ABOUT GETTING THE BEST OUT OF YOUR TEAM

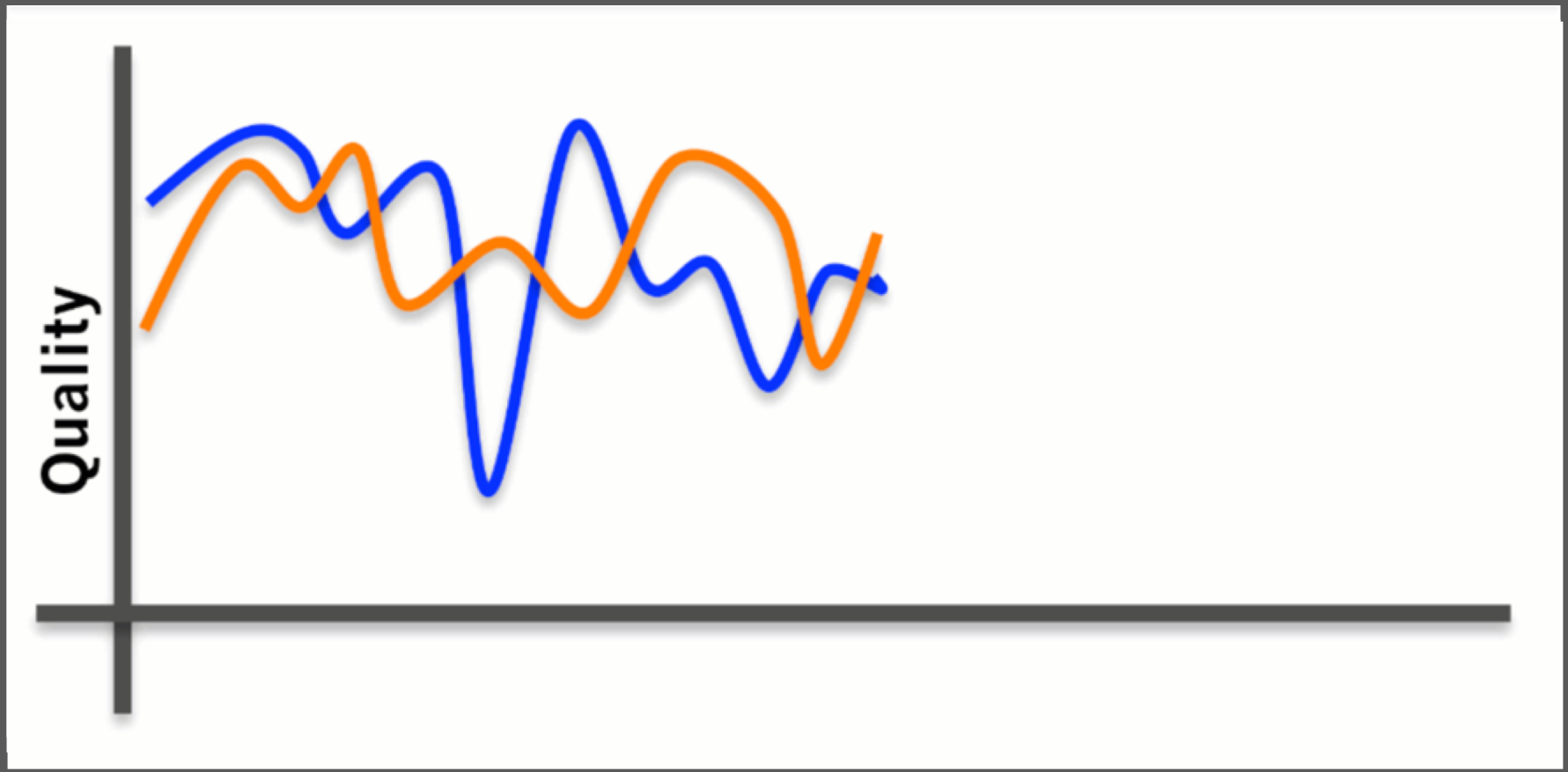
SOLO PROGRAMMING

source: http://imgur.com/fGlgTyg.gif

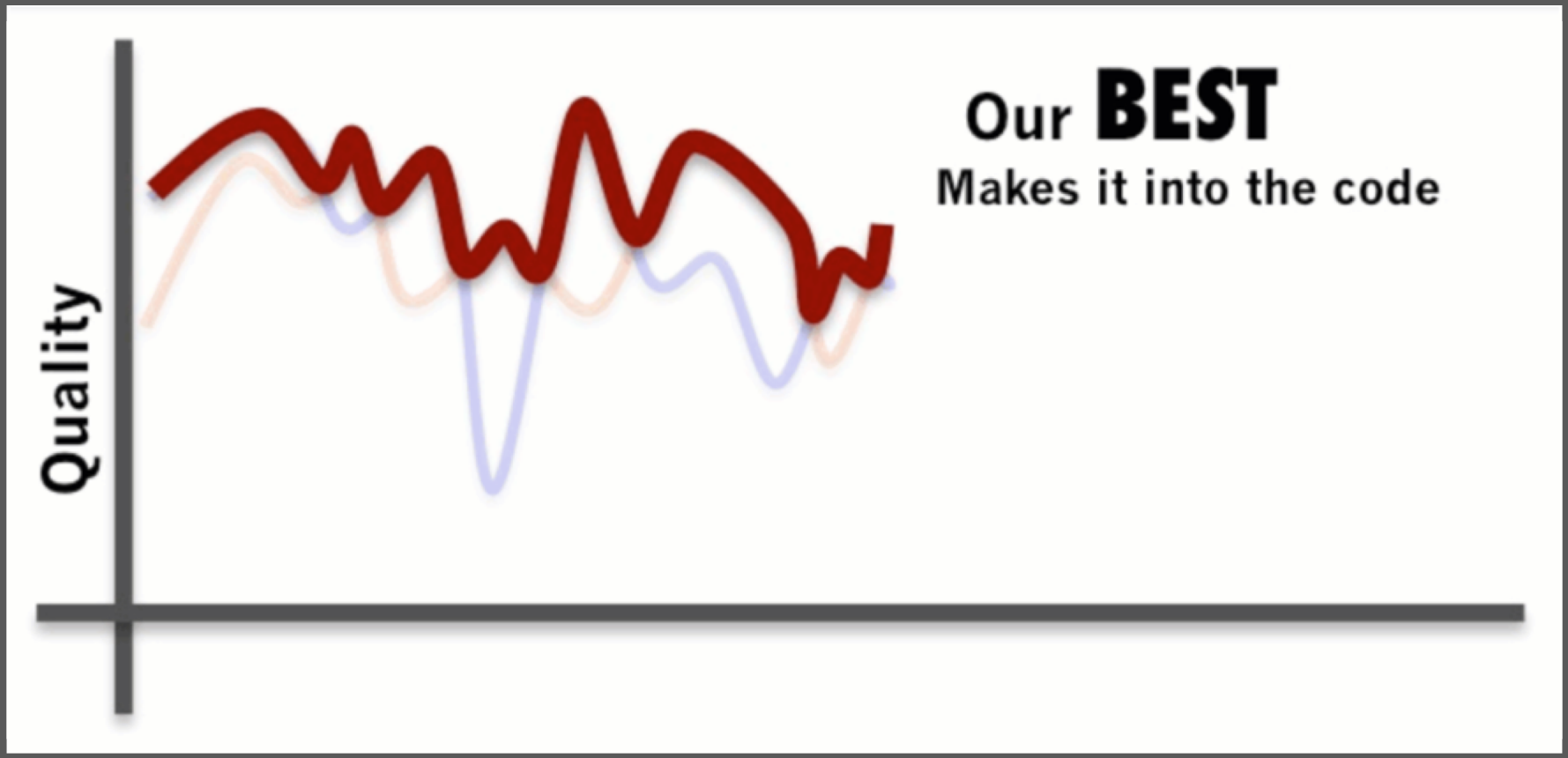
SOLO PROGRAMMING

source: http://i.imgur.com/fGlgTyg.gif

PAIR PROGRAMMING

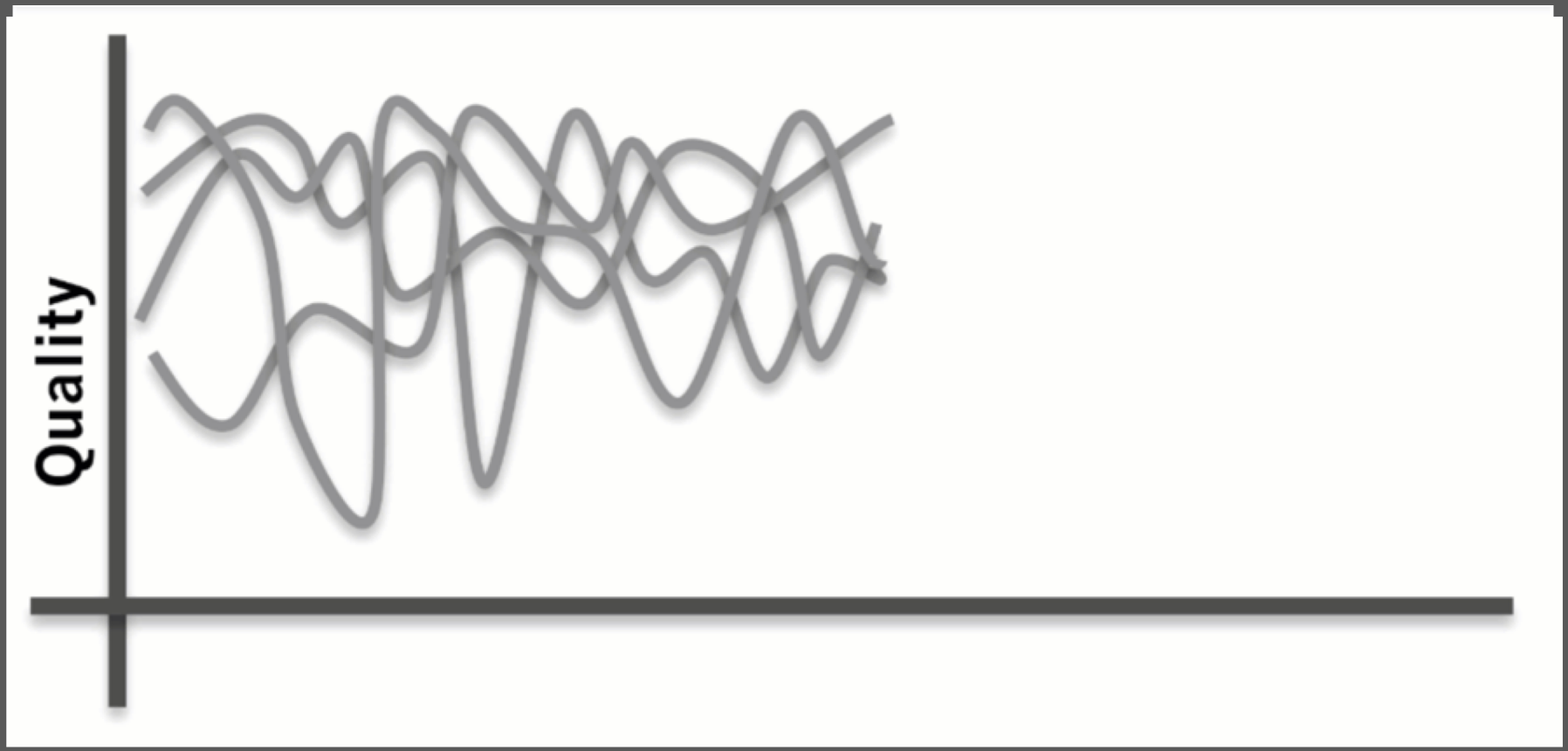


source: http://imgur.com/fGlgTyg.gif



source: http://imgur.com/fGlgTyg.gif

MOB PROGRAMMING

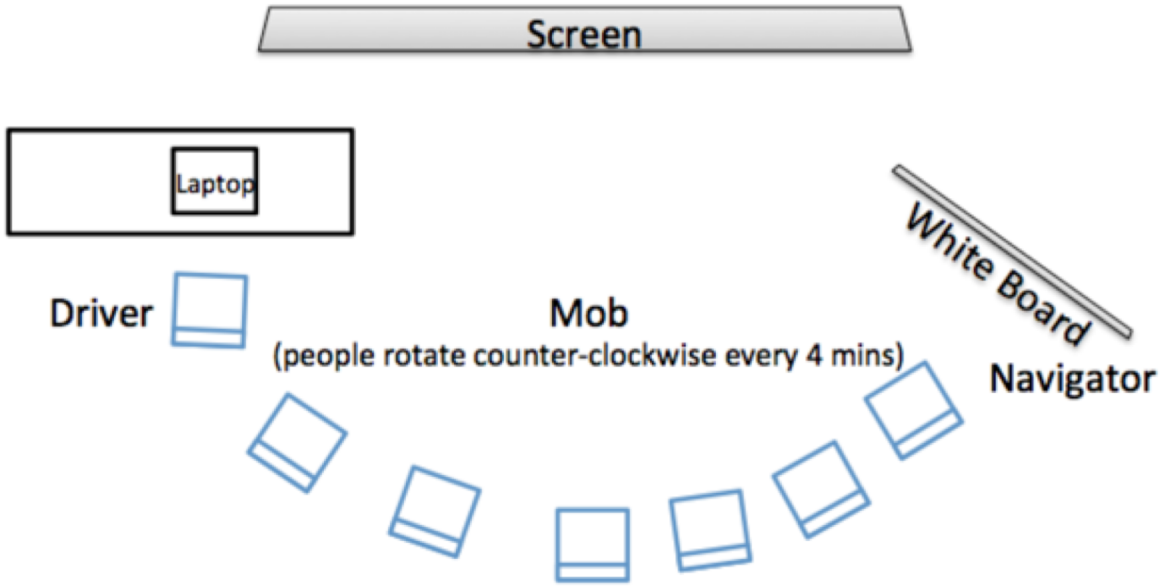


source: http://imgur.com/fGlgTyg.gif

MOB PROGRAMMING

source: http://imgur.com/fGlgTyg.gif

Mob Programming Setup



MOB PROGRAMMING ROLES

- The Driver: “no thinking, just typing”
- The Navigator: the main person programming
- The Mob: Checking the navigator, Contributing insights, Getting ready to rotate
- The Facilitator: Help guide the mob (Instructor)

PERSONAL PROCESSES

ASKING QUESTIONS

institute for SOFTWARE RESEARCH | Carnegie Mellon University School of Computer Science

HOW TO GET GOOD ANSWERS

Ask good questions:

"I am trying to ___, so that I can ___.
I'm running into ___.
I've looked at ___ and tried ___."

http://kwugirl.blogspot.com/2014/04/how-to-be-better-junior-developer_25.html

GOOD QUESTIONS

Keep a log of questions and answers (Make new mistakes! Ask new questions!)

Try to find answers first (timebox search)

Keep mental model of who knows what

Help others learn how to ask good questions too

RUNNING A MEETING

institute for SOFTWARE RESEARCH | Carnegie Mellon University School of Computer Science

HOW TO RUN A MEETING

- The Three Rules of Running a Meeting
  - Set the Agenda
  - Start on Time. End on Time.
  - End with an Action Plan
- Give Everyone a Role
  - Establish Ground Rules
  - Decision, or Consensus?

https://www.nytimes.com/guides/business/how-to-run-an-effective-meeting

HOW TO RUN A MEETING

- Control the Meeting, Not the Conversation
  - Let Them Speak
- Make Everyone Contribute
  - Manage Personalities
  - Be Vulnerable
  - Make Everyone a Judge
- Make Meetings Essential
  - Do a Meeting Audit

https://www.nytimes.com/guides/business/how-to-run-an-effective-meeting


WHY DO YOU WANT TO CALL A MEETING?
DON'T KNOW OTHER WAYS TO COLLABORATE
SHARE INFORMATION
BUILD RELATIONSHIPS
CAN'T SELF-MOTIVATE
MAKE A DECISION
SOLVE A PROBLEM
GET FEEDBACK
GENERATE IDEAS
MAKE A PLAN
REFLECT & IMPROVE
PROVE HOW IMPORTANT I AM
Would email or a Confluence page do the trick?
Like a manager 1-1 thing?
Does this need to happen in real time?
Seriously?
GREAT. NO MEETING!
Can you find a time that works for everyone fairly soon?
BOOM. SCHEDULE IT!
Does everyone have enough background info to get it done?
VERY FUNNY. NO MEETING!
OK, FINE. BUT THIS BETTER BE IMPORTANT.
Do you just want an excuse to eat donuts?
FIX THAT FIRST. NO MEETING.
Has this been discussed before?
SO EAT A DONUT. NO MEETING!
You're procrastinating, aren't you?
Should we believe you?..
You really think this time will be different?
BOOM. SCHEDULE IT!
DON'T BE LAME. NO MEETING!
LIAR. NO MEETING!
BOOM. SCHEDULE IT!
GET REAL. NO MEETING!
atlassian.com/effectivemeetings
ATLASSIAN

RANDOM ADVICE

- Note takers have a lot of power to steer the meeting
  - Collaborative notes are even better!
- Different meeting types have different best practices
  - Regular team meeting
  - Decision-making meeting
  - Brainstorming meeting
  - Retrospective meeting
  - One-on-one meeting