Code Review

17-356/17-766
Software Engineering for Startups

https://cmu-17-356.github.io

Andrew Begel and Fraser Brown

Admin

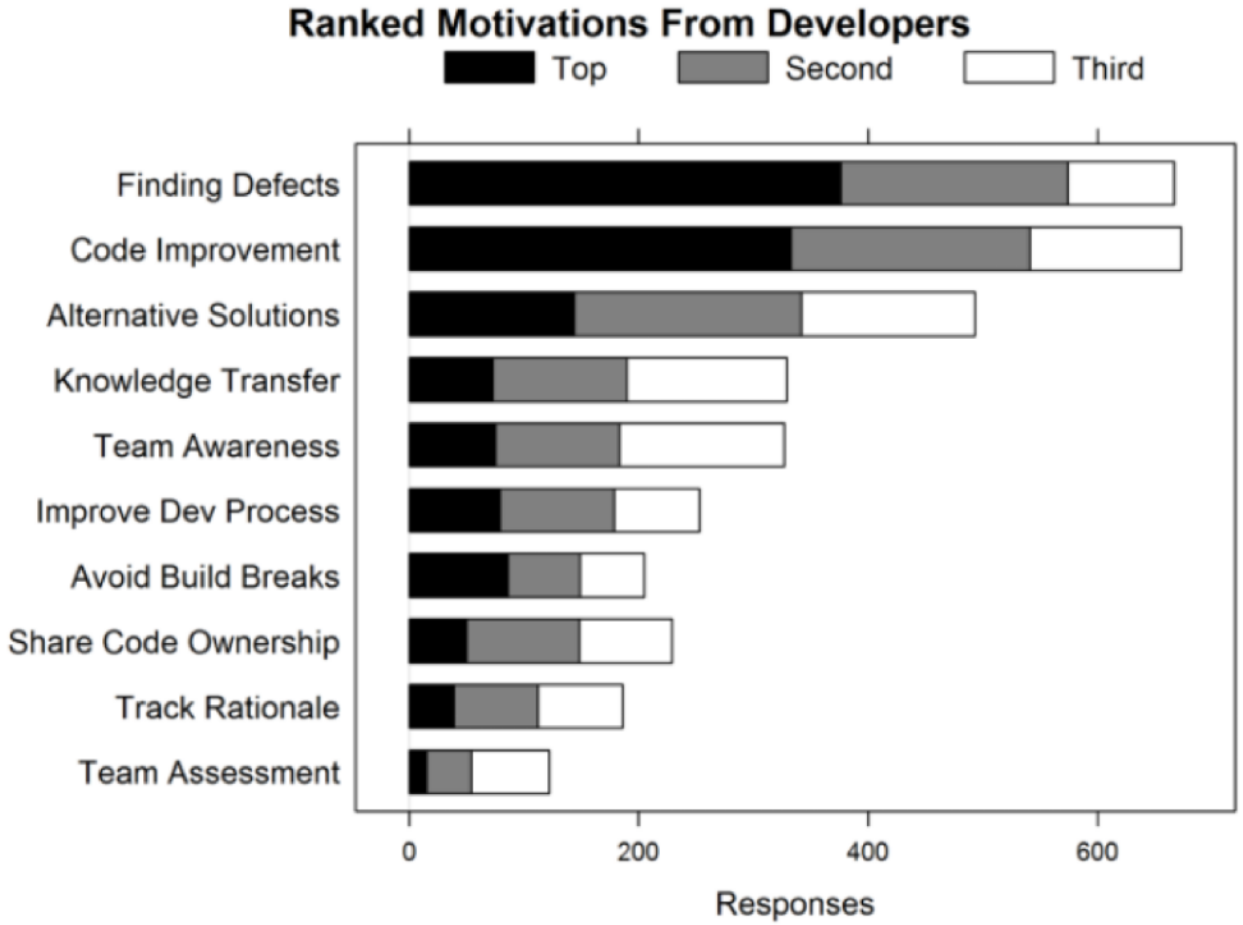
- HW3 due tonight
- HW4 released tonight
- Survey at the end of class
- Continue to work on P1

Code Review

- Does this code do what it claims?
  - Are there any programming bugs?
- Does this code clearly “explain” what it does?
- Why are we making this change?
  - Are there any design bugs?

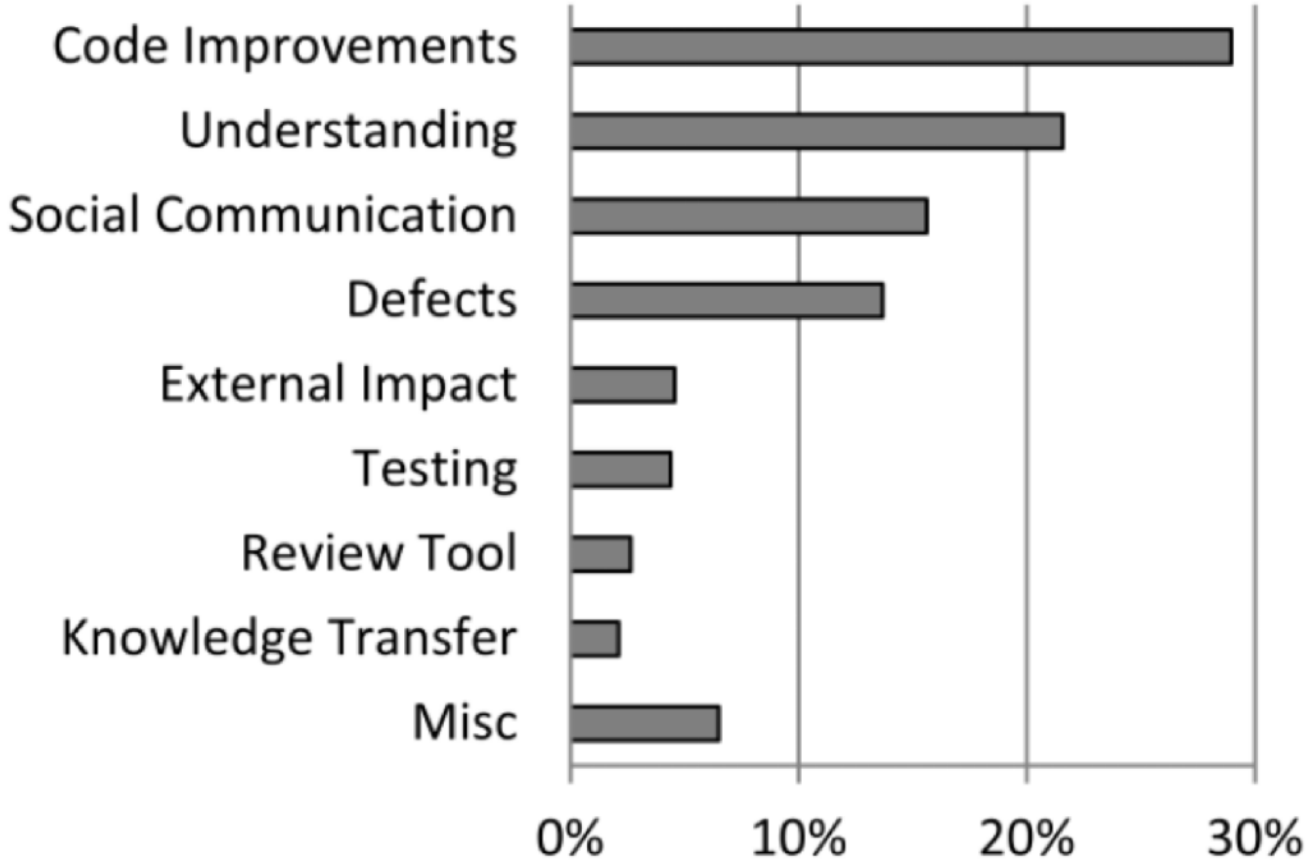
Expectations and Outcomes

Code Review at Microsoft

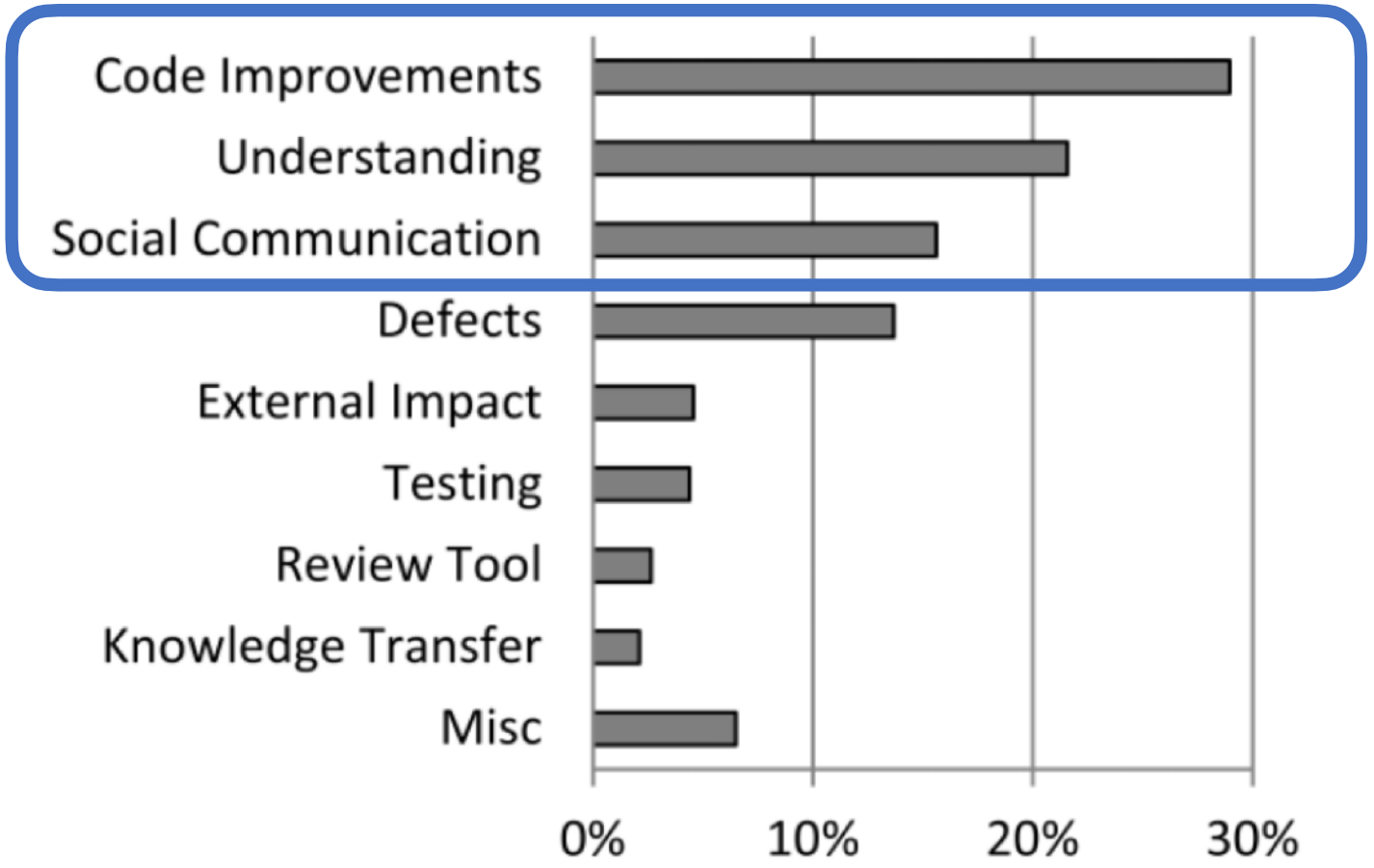


Bacchelli, Alberto and Christian Bird. "Expectations, outcomes, and challenges of modern code review." Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013.

Outcomes (Analyzing Reviews)



Outcomes (Analyzing Reviews)



Code Review at Google

- **Introduced to "force developers to write code that other developers could understand"**
- Three benefits:
  - checking the consistency of style and design
  - ensuring adequate tests
  - improving security by making sure no single developer could commit arbitrary code without oversight

Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. 2018. Modern Code Review: A Case Study at Google. International Conference on Software Engineering

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

What is frustrating about code review?

Code Review Principles (starting point)

- Start with the “big ideas”
- Automate the little things
- Focus on understanding
- Remember a person wrote the code
- Don’t overwhelm the person with feedback

3 Pillars of Social Interaction

- Humility
  - You are not the center of the universe (nor is your code!). You're neither omniscient, nor infallible. You are open to self-improvement.
- Respect
  - You genuinely care about the others you work with. You treat them kindly and appreciate their abilities and accomplishments.
- Trust
  - You believe others are competent and will do the right thing, and when appropriate, you are OK with letting them drive.

Code Review Principles (starting point)

- Start with the “big ideas”
- Automate the little things
- Focus on understanding
- Remember a person wrote the code
- Don’t overwhelm the person with feedback

With your teams, for your projects: what are high-level best practices you care about? What can you automate?

Checklists help manage complex processes

OFFICIAL A.A.F. PILOT'S CHECK LIST

PILOT

BEFORE STARTING

CO-PILOT

BEFORE TAKE OFF

AFTER TAKE OFF

BEFORE LANDING

STARTING ENGINES

FINAL APPROACH

The Checklist: https://www.newyorker.com/magazine/2007/12/10/the-checklist

Activity: Review code

Adds accounts #1

Open **mlfbrown** wants to merge 1 commit into `main` from `mlf/example1`

Conversation 0 · Commits 1 · Checks 0 · Files changed 2

mlfbrown commented now

No description provided.

Viewed

```
1  + const Account = function(name, nickname, date) {
2  +   this.name = name;
3  +         this.nickname = nickname;
4  +         this.date = date;
5  + }
6  +
7  + function makeDefaultAccount() {
8  +         return {
9  +                 name: "Default",
10 +                 nickname: "Default",
11 +                 date: "Jan 1 2025"
12 +         }
13 + }
14 +
15 + function updateAccount(account, nickname) {
16 +         account.nickname = nickname;
17 + }
```

That was an easy one.

Think for a few minutes about things that have tripped you up in the past.

What do you keep an eye out for now?

Sample Low-Level Coding Checklist

- General
  - Are all changes relevant?
  - Do the classes and methods fulfill their purpose?
  - Are the messages and texts for the user correct?
- Classes
  - Are all assignments of attributes correct?
  - Are the classes implemented correctly?
- Methods
  - Do methods always return a valid value?
  - Do methods check parameters for validity (if needed)?
  - Are all parameters used?
- Arguments
  - Are the correct arguments used in all method calls?
- Variables
  - Are all variables, counters, and accumulators initialized properly and, if necessary, re-initialized every time they are used?
  - Are all declared variables being used?
- If-Then Statements
  - Do the if-else statements fit the intended purpose?
  - Are all edge cases handled?
- Loops
  - Do the loops end under all possible conditions?
  - Are the break and continue statements used properly?
- Recursion
  - Does recursion terminate properly?
- Errors
  - Are exceptions handled correctly?
- Final Check
  - Are all changes consistent with one another?