Code Review

Overview

- Today we'll learn about reviews, walkthroughs and inspections in software development
- Specifically, we'll focus on the actual practice of code reviews
  - checklist-directed code reviews, code paraphrasing, structured code walkthroughs, etc.
- We'll conclude with a case study on bias in code reviews

Software Development Artifacts

Artifacts of Software Development

- What do we produce when making software?
    - Plans, requirements specifications, design specifications, source code, comments, test cases, test reports, user documentation, technical documentation
- Of all these, we can only actually test one of them (code), and only when we are far along (at least partially runnable)
- So how can we address quality and detect faults earlier?

REVIEWS!

Reviews, Walkthroughs and Inspections

- How do we differentiate between the terminology including reviews, walkthroughs and inspections?

Design Reviews

- Refer to the management practice of meetings to informally consider state of the project at certain stages, in order to gain confidence in project direction
- Used to provide confidence that the design is sound
- Often attended by management and customers

Reviews, Walkthroughs and Inspections

Code Walkthroughs/Code Reviews

- Refer to an informal technical review, normally carried out by developers
- Used by development teams to improve product quality by involving whole team in quality assurance at each stage
- Focus is on critical analysis of artifacts with a goal of finding defects
- This is a decision process where a reviewer answers the question – “is the code acceptable?” If the answer is no then feedback is provided.

Reviews, Walkthroughs and Inspections

Inspections

- Refer to a completely formal process of review - also known as formal technical reviews
- A formal system used to identify and remove defects, and improve the overall quality of the development process
  - Includes formal written reports, defect data collection and analysis, required standards and measures
- Emphasis on documenting process progress and defects
- First introduced by Fagan (IBM)

Code Review Practices

- There are a range of actual practices that can be applied to implement the actual review of the artifact. The practices often vary depending on the company, industry or open source project.
- Practices for code review include:
    - Checklists
    - Paraphrasing
    - Structured walkthroughs
    - Axiomatic formal proofs

Code Checklists

Checklists

- Checklists may include general properties for any program or specific properties for a specific program
- Both desired properties (ones we want the code to have) and undesired properties (ones the code should not have) may appear in the list
- Checklist items can range from simple shallow properties (e.g., format) to deep semantic properties (e.g., termination)
- The idea is that the reviewer should look through the code to check for the presence or absence of each individual item, and check it off the list
  - In addition to checking items on the list, the reviewer must also check the correctness of the code

Code Checklists

An Example: Generic Java Code Inspection Checklist

1. **Variable and Constant Declaration Defects**
   - 1.1 Are descriptive variable and constant names used in accord with naming conventions?
   - 1.2 Are there variables with confusingly similar names?
   - 1.3 Is every variable properly initialized?
   - 1.4 Can any non-local variables be made local?
   - 1.5 Are there literal constants that should be named constants?
   - 1.6 Are there variables that should be constants?
2. **Method Definition Defects**
   - 2.1 Are descriptive method names used in accord with naming conventions?
   - 2.2 Is every parameter value checked before being used?
   - 2.3 Does every method return a correct value at every return point?

. . .

Code Checklists

An Example: Generic Java Code Inspection Checklist (continued)

. . .

4. **Computation Defects**

4.1 Is underflow or overflow possible in any computation?

4.2 Does any expression depend on order of evaluation of operators? Are parentheses used to
avoid ambiguity?

. . .

6. **Control Flow Defects**

6.1 Will all loops terminate in all cases?

. . .

Code Paraphrasing

Reading the Code in a Natural Language

- Code paraphrasing is the original method of review described by Fagan for use in code inspections
- Consists of reading the lines of code for their meaning in the problem domain, not in the programming language
  - should avoid mentioning variables or control flow
  - should be phrased in terms of the concepts and processes being implemented
  - should be seeded by scenarios
- The object is to ensure that the code really does implement what we want to have done
- Paraphrasing is often coupled with checklists – the checklist addresses low level properties of the code itself, while the paraphrasing addresses its high level meaning

```
//Paraphrase the below code
static int binarySearch (int[] list, int key, int start, int end) {
   int middle = (start + end) / 2;

   if (list[middle] < key)
      return binarySearch (list, key, middle, end);
   else if (list[middle] > key)
      return binarySearch (list, key, start, middle);
   else
      return middle;
}

static int find (int[] list, int key) {
   return binarySearch (list, key, 0, list.length);
}
```

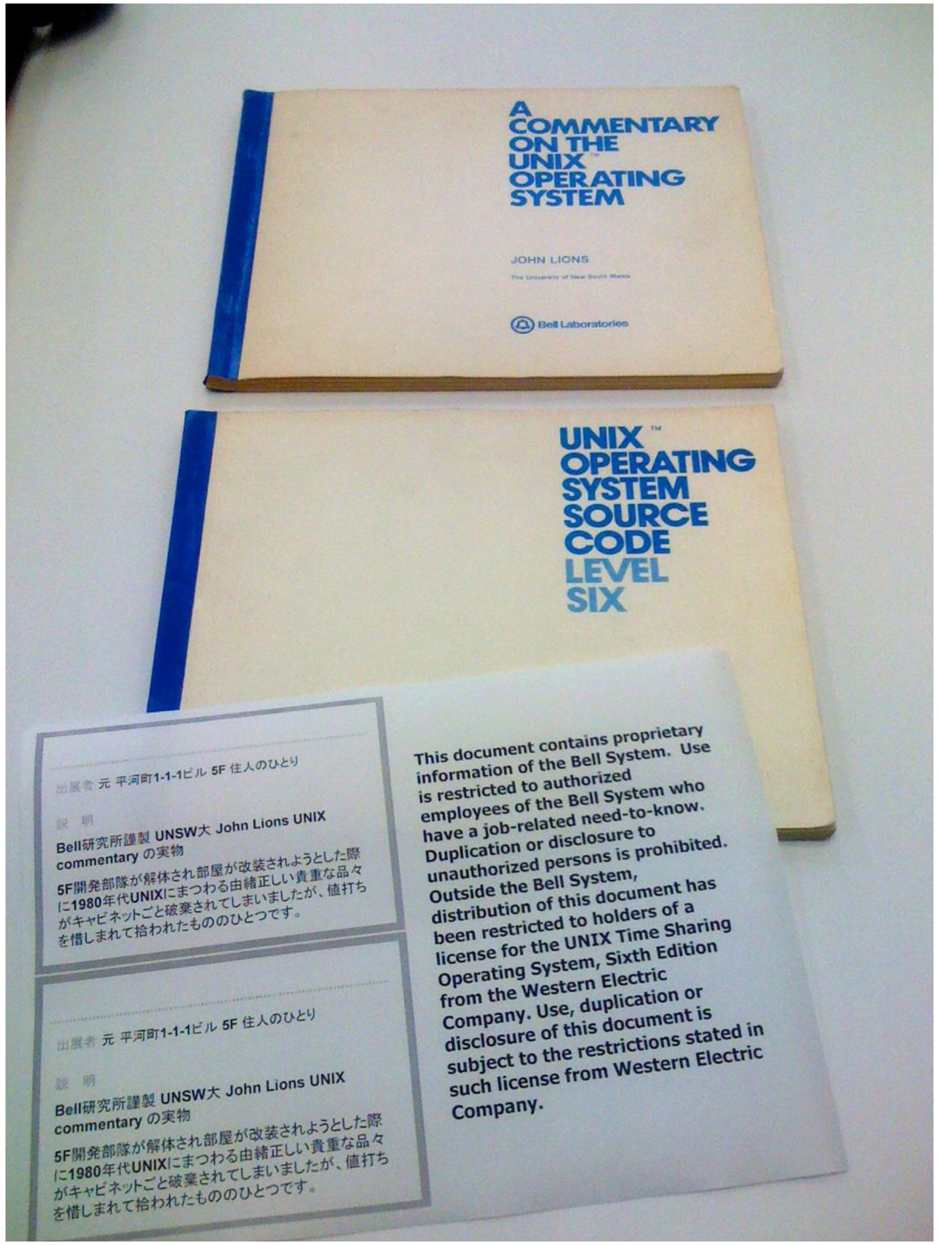
Structured Code Walkthroughs

A Guided Tour

- Code walkthroughs are a sort of guided tour of the program
- When used in code inspections, walkthroughs are conducted by the code reviewers in the meeting as a sort of dynamic exploration of how the code works
- Code walkthroughs can also be static – documented as an annotated version of the source code
- Both kinds of walkthroughs have been found to be very effective in training, to introduce new programmers to the code

The Lions Commentary

- Perhaps the most famous code walkthrough in history is the 1977 "Source Code and Commentary on UNIX 6th Edition" by John Lions of the University of New South Wales


A COMMENTARY ON THE UNIX™ OPERATING SYSTEM
JOHN LIONS
The University of New South Wales
Bell Laboratories
UNIX™ OPERATING SYSTEM SOURCE CODE LEVEL SIX

出展者 元 平河町1-1-1ビル 5F 住人のひとり

説 明
Bell研究所謹製 UNSW大 John Lions UNIX commentary の実物
5F開発部隊が解体され部屋が改装されようとした際に1980年代UNIXにまつわる由緒正しい貴重な品々がキャビネットごと破棄されてしまいましたが、値打ちを惜しまれて拾われたもののひとつです。

出展者 元 平河町1-1-1ビル 5F 住人のひとり

説 明
Bell研究所謹製 UNSW大 John Lions UNIX commentary の実物
5F開発部隊が解体され部屋が改装されようとした際に1980年代UNIXにまつわる由緒正しい貴重な品々がキャビネットごと破棄されてしまいましたが、値打ちを惜しまれて拾われたもののひとつです。

This document contains proprietary information of the Bell System. Use is restricted to authorized employees of the Bell System who have a job-related need-to-know. Duplication or disclosure to unauthorized persons is prohibited. Outside the Bell System, distribution of this document has been restricted to holders of a license for the UNIX Time Sharing Operating System, Sixth Edition from the Western Electric Company. Use, duplication or disclosure of this document is subject to the restrictions stated in such license from Western Electric Company.

Source: https://upload.wikimedia.org/wikipedia/commons/5/50/Lions_Commentary_Unix.jpeg

A Famous Static Walkthrough

The Lions Commentary

- The walkthrough is organized as two parts:
  1. a line-numbered commented listing of the Unix kernel source code, and
  2. a detailed commentary organized in parallel with the source code structure, indexed by line number in the source code listing

A Famous Static Walkthrough

The Lions Commentary

- The commentary pointed out the meaning and effect of each section of code on the whole system, reminded the reader of what each mentioned variable represents and where it is defined in the ten thousand lines of code, and so on
- This book was quashed and never published because Western Electric (now AT&T) considered Unix a trade secret
- Underground photocopies made it out to hackers all over the world, and it was used as training material for students who later made FreeBSD, Linux, and all other Unix derivatives

Lightweight Code Review Practices

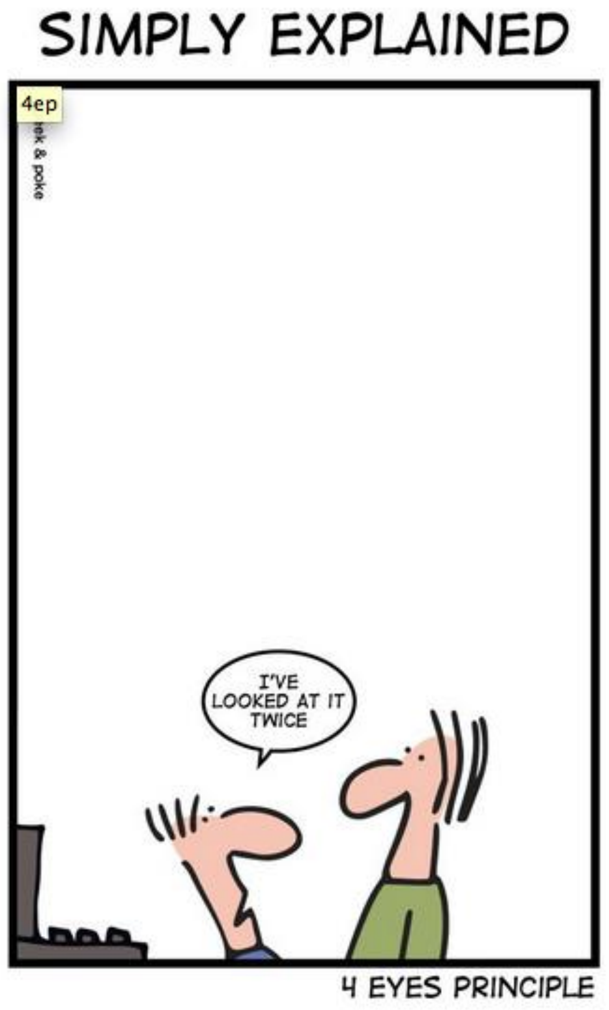
Chief Programmer Teams

- The four eyes principle is part of the chief programmer team software development method, an early predecessor of Agile
- CPT development involves weekly meetings in which the whole team informally reviews the system design, architecture, interfaces and schedule once each week, continuously updating each of these as the project proceeds

Lightweight Code Review Practices

Four Eyes Principle

- One such practice is the four eyes principle, in which programmers work in loose pairs, where each module or class authored by one programmer is informally inspected by the other
- Both programmers are equally responsible for the quality and correctness of the code, and both must "sign off" on it before it is released for testing and integration
- In small teams, this method has been shown to produce code with very few defects



Source: http://geek-and-poke.com/2012/07/last-day-of-the-geekpoke-simply-explained-weekend.html

Lightweight Code Review Practices

Continuous Review in Agile Development

- The four eyes principle is the predecessor of pair programming, another lightweight inspection method
- Code refactoring is another continuous inspection method descended from CPT’s weekly design reviews

(Manual) White Box Testing

- When applied manually, most white box testing methods force the test author to examine the code in detail which can effectively act as a code review method.

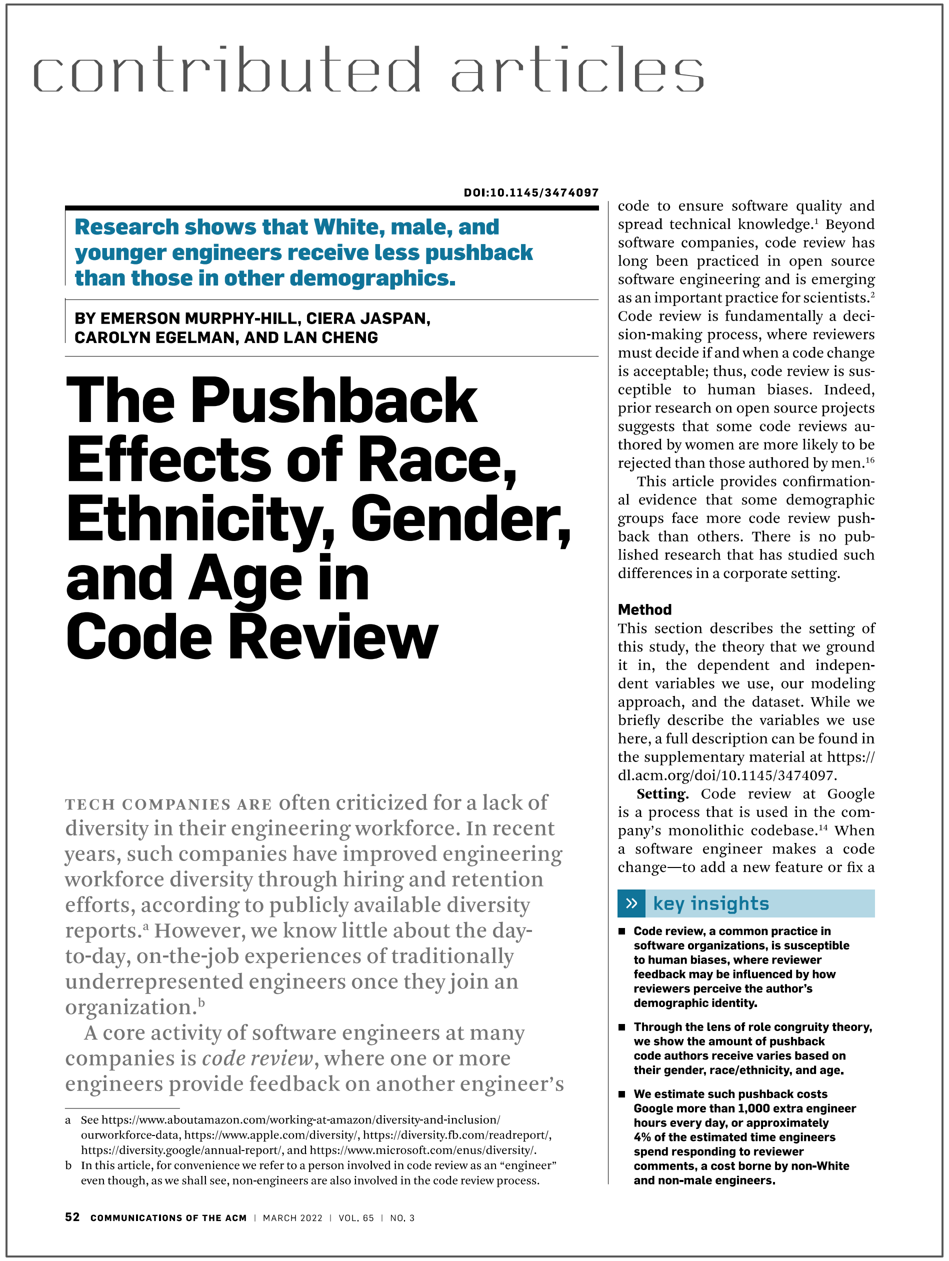
Lightweight Code Review Practices

Informal Review using Code Review Tools

- Code review tools are the main method of communication (no meetings)
- The author circulates the code to be reviewed to the reviewers directly and their review comments are sent back directly to the author.
- Upon receiving the review comments, the author implements any required changes and re-circulates the code to the reviewers to sign-off on the review.

Disadvantages

- The lack of a moderator in this process can cause issues with objectivity


contributed articles

DOI:10.1145/3474097

Research shows that White, male, and younger engineers receive less pushback than those in other demographics.

BY EMERSON MURPHY-HILL, CIERA JASPAN, CAROLYN EGELMAN, AND LAN CHENG

The Pushback Effects of Race, Ethnicity, Gender, and Age in Code Review

TECH COMPANIES ARE often criticized for a lack of diversity in their engineering workforce. In recent years, such companies have improved engineering workforce diversity through hiring and retention efforts, according to publicly available diversity reports.[a] However, we know little about the day-to-day, on-the-job experiences of traditionally underrepresented engineers once they join an organization.[b]

A core activity of software engineers at many companies is *code review*, where one or more engineers provide feedback on another engineer's code to ensure software quality and spread technical knowledge.[1] Beyond software companies, code review has long been practiced in open source software engineering and is emerging as an important practice for scientists.[2] Code review is fundamentally a decision-making process, where reviewers must decide if and when a code change is acceptable; thus, code review is susceptible to human biases. Indeed, prior research on open source projects suggests that some code reviews authored by women are more likely to be rejected than those authored by men.[16]

This article provides confirmational evidence that some demographic groups face more code review pushback than others. There is no published research that has studied such differences in a corporate setting.

Method

This section describes the setting of this study, the theory that we ground it in, the dependent and independent variables we use, our modeling approach, and the dataset. While we briefly describe the variables we use here, a full description can be found in the supplementary material at https://dl.acm.org/doi/10.1145/3474097.

Setting. Code review at Google is a process that is used in the company's monolithic codebase.[14] When a software engineer makes a code change—to add a new feature or fix a

a See https://www.aboutamazon.com/working-at-amazon/diversity-and-inclusion/ourworkforce-data, https://www.apple.com/diversity/, https://diversity.fb.com/readreport/, https://diversity.google/annual-report/, and https://www.microsoft.com/enus/diversity/.

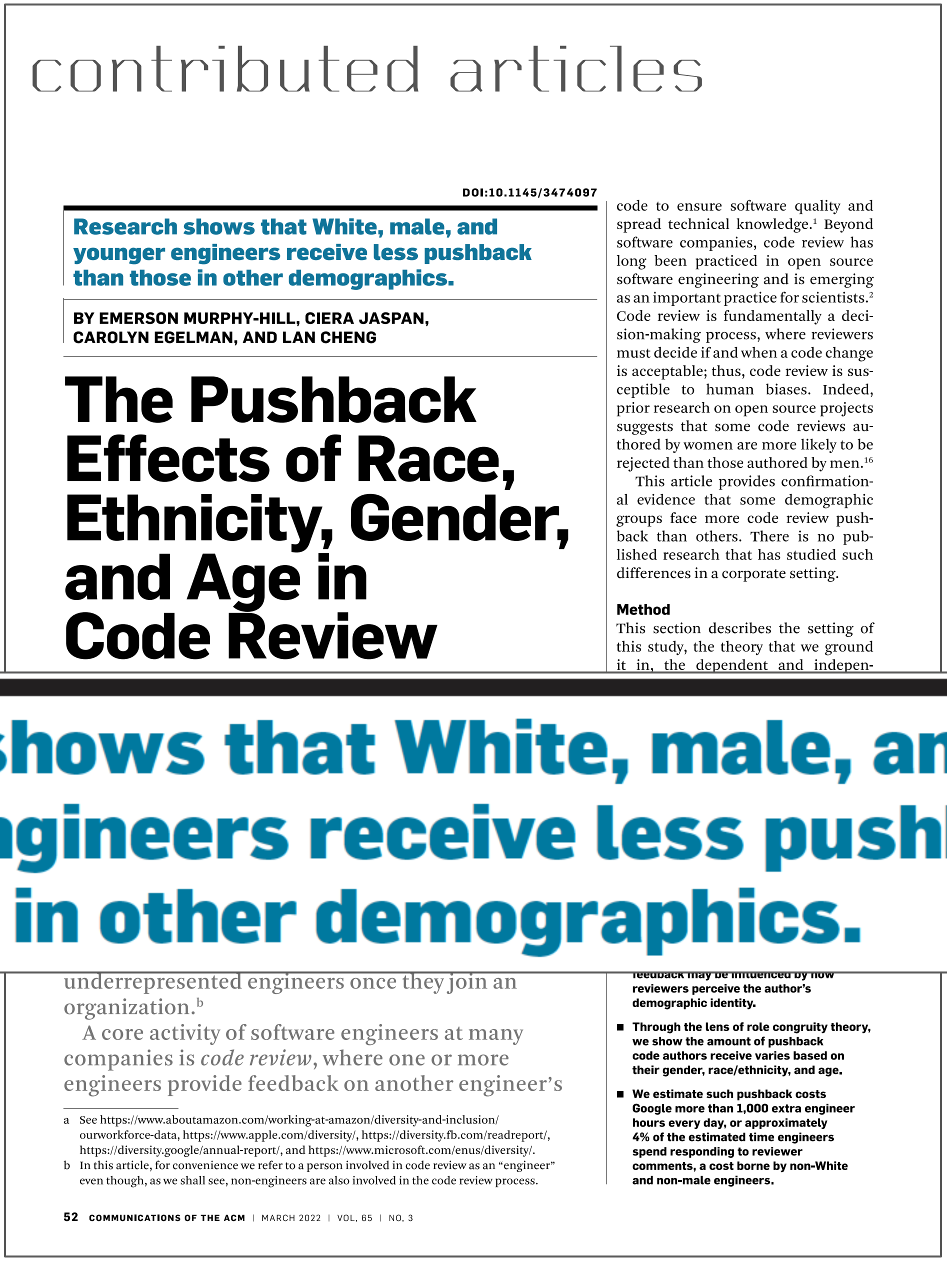
b In this article, for convenience we refer to a person involved in code review as an "engineer" even though, as we shall see, non-engineers are also involved in the code review process.

key insights

- **Code review, a common practice in software organizations, is susceptible to human biases, where reviewer feedback may be influenced by how reviewers perceive the author's demographic identity.**
- **Through the lens of role congruity theory, we show the amount of pushback code authors receive varies based on their gender, race/ethnicity, and age.**
- **We estimate such pushback costs Google more than 1,000 extra engineer hours every day, or approximately 4% of the estimated time engineers spend responding to reviewer comments, a cost borne by non-White and non-male engineers.**

52 COMMUNICATIONS OF THE ACM | MARCH 2022 | VOL. 65 | NO. 3

Emerson Murphy-Hill, Ciera Jaspan, Carolyn Egelman, Lan Cheng. "The Pushback Effects of Race, Ethnicity, Gender, and Age in Code Review," Communications of the ACM, March 2022, Vol. 65 No. 3, Pages 52-57



contributed articles

DOI:10.1145/3474097

Research shows that White, male, and younger engineers receive less pushback than those in other demographics.

BY EMERSON MURPHY-HILL, CIERA JASPAN, CAROLYN EGELMAN, AND LAN CHENG

The Pushback Effects of Race, Ethnicity, Gender, and Age in Code Review

underrepresented engineers once they join an organization.[b]

A core activity of software engineers at many companies is *code review*, where one or more engineers provide feedback on another engineer's code to ensure software quality and spread technical knowledge.[1] Beyond software companies, code review has long been practiced in open source software engineering and is emerging as an important practice for scientists.[2] Code review is fundamentally a decision-making process, where reviewers must decide if and when a code change is acceptable; thus, code review is susceptible to human biases. Indeed, prior research on open source projects suggests that some code reviews authored by women are more likely to be rejected than those authored by men.[16]

This article provides confirmational evidence that some demographic groups face more code review pushback than others. There is no published research that has studied such differences in a corporate setting.

Method

This section describes the setting of this study, the theory that we ground it in, the dependent and indepen-

a See https://www.aboutamazon.com/working-at-amazon/diversity-and-inclusion/ourworkforce-data, https://www.apple.com/diversity/, https://diversity.fb.com/readreport/, https://diversity.google/annual-report/, and https://www.microsoft.com/enus/diversity/.

b In this article, for convenience we refer to a person involved in code review as an "engineer" even though, as we shall see, non-engineers are also involved in the code review process.

feedback may be influenced by how reviewers perceive the author's demographic identity.

- **Through the lens of role congruity theory, we show the amount of pushback code authors receive varies based on their gender, race/ethnicity, and age.**
- **We estimate such pushback costs Google more than 1,000 extra engineer hours every day, or approximately 4% of the estimated time engineers spend responding to reviewer comments, a cost borne by non-White and non-male engineers.**

Research shows that White, male, and younger engineers receive less pushback than those in other demographics.

Emerson Murphy-Hill, Ciera Jaspan, Carolyn Egelman, Lan Cheng. "The Pushback Effects of Race, Ethnicity, Gender, and Age in Code Review," Communications of the ACM, March 2022, Vol. 65 No. 3, Pages 52-57

Study Background

- Because code review is a decision process it is susceptible to bias
- Study is grounded in role congruity theory – *"a member of a group will receive negative evaluations when stereotypes about the group misalign with the perceived qualities necessary to succeed in a role."*
- Authors are studying if code review evaluations will vary by gender race/ethnicity and age (independent variables)
- Measure pushback (dependent variable) – *"the perception of unnecessary interpersonal conflict in code review while a reviewer is blocking a change request."*
  - Can take different forms including excessive change requests and withholding of approval

Study Context – Google

- Study looks at code review in Google – *“When a software engineer makes a code change—to add a new feature or fix a defect—that code must be reviewed by at least one other engineer. Reviewers evaluate the fitness for purpose of the change, as well as its quality. If they have concerns or questions, they express those comments in the code review tool.”*
- Analyzed code reviews at Google between Jan. 2019 and June 2019
- Code reviews were restricted to one of the main code review tools used at Google

Study Results

- Code changes by women **1.21x** more likely to receive pushback (when compared to changes by men)
- Code changes by Black+ (**1.54x**), Hispanic or Latinx+ (**1.15x**) and Asian+ (**1.42x**) received more pushback (when compared to changes by White+ engineers)
- Code changes by older engineers faced more likelihood of pushback when compared to changes by younger engineers
  - 60+ year old engineers faced **3x** more pushback than 18-24 year old authors at the same level and with the same tenure at Google

Study Results

"We estimate that the total amount of excess time spent during the study period was 1,050 engineer hours per day, or about 4% of the estimated time engineers spend responding to reviewer comments, a cost borne by non-White and non-male engineers."

Addressing bias in code review

- The authors' include several options for addressing code review bias:
  - Bias training
  - Anonymous code reviews

- In group discuss other options for addressing code review bias and report back to the class.

Code Review

Summary

- Today we learned about reviews, walkthroughs and inspections in software development
  - checklist-directed code reviews, code paraphrasing, structured code walkthroughs, and lightweight code review methods
- We concluded with reviewing a recent study on bias in code reviews at Google

References

- https://cacm.acm.org/magazines/2022/3/258909-the-pushback-effects-of-race-ethnicity-gender-and-age-in-code-review/fulltext