




Engineering Impacts of Anonymous Author Code Review: A Field Experiment

Emerson Murphy-Hill , Jillian Dicker, Margaret Morrow Hodges, Carolyn D. Egelman, Ciera Jaspán , Lan Cheng, Elizabeth Kammer, Ben Holtz, Matthew A. Jorde, Andrea Knight Dolan, and Collin Green 

Abstract—Code review is a powerful technique to ensure high quality software and spread knowledge of best coding practices between engineers. Unfortunately, code reviewers may have biases about authors of the code they are reviewing, which can lead to inequitable experiences and outcomes. In principle, anonymous author code review can reduce the impact of such biases by withholding an author's identity from a reviewer. In this paper, to understand the engineering effects of using author anonymous code review in a practical setting, we applied the technique to 5217 code reviews performed by 300 software engineers at Google. Our results suggest that during anonymous author code review, reviewers can frequently guess authors' identities; that focus is reduced on reviewer-author power dynamics; and that the practice poses a barrier to offline, high-bandwidth conversations. Based on our findings, we recommend that those who choose to implement anonymous author code review should reveal the time zone of the author by default, have a break-the-glass option for revealing author identity, and reveal author identity directly after the review.

Index Terms—Code review, unbiasing

1 INTRODUCTION

WHILE developers believe that code changes are accepted based on change quality and fitness [1], prior research suggests that when women use profile pictures and gender-identifiable names, the acceptance of their open source code contributions drops, compared to peers with gender-neutral profiles [2]. Previous research suggests that such gender disparities are due to implicit gender bias and have been replicated in a variety of work contexts [3], and also extend beyond gender to race [4], age [5], and physical attractiveness [6].

Outside of code review, implicit bias in professional decision making is increasingly handled through anonymization, where irrelevant personal details are purposefully hidden from the decision maker. For example, research on performing orchestra auditions without seeing the person who auditioned “fostered impartiality in hiring and increased the proportion of women in symphony orchestras” [7]. Academic papers reviewed by scholars who are aware of author identity gives “a significant advantage to papers with famous authors and authors from high-prestige institutions” compared to when author identity is not revealed during review [8].

With a similar motivation, having engineers review code changes without being explicitly informed of who made those changes – *anonymous author code review*¹ – can in principle reduce the effect of bias in organizations. Indeed, in response [10] to the GitHub study on gender bias [2], Mozilla developed a browser extension that anonymizes GitHub pull requests [13], yet the extension has not been evaluated. In the social sciences, Kim and colleagues identified anonymous author code review as a step towards reducing structural sexism [9], a problem tech companies such as Google have been criticized for [14].

But we know essentially nothing about how anonymous author code review would work in practice. After Facebook reportedly replicated the GitHub study [15], news reports indicate that Facebook's VP of Engineering rejected the practice saying “Hiding the identity of authors or reviewers is counterproductive from an engineering perspective” [16]. But is it counterproductive in practice? Unfortunately, we know of no empirical evidence.

So while in principle anonymous author code review reduces the impact of biases, we don't know what other effects it might have on the engineering process. This paper seeks to understand these effects by answering the following research questions:

RQ1: How Often Can Reviewers Guess Author Identities?
When reviewers are aware of who the author is during

• The authors are with Core Systems, Google Inc., Mountain View, CA 94043 USA. E-mail: {emersonm, jdicker, hodgesm, cegelman, ciera, lancheng, eakammer, benholtz, majorde, aknight, colling}@google.com.

Manuscript received 8 Mar. 2020; revised 28 Jan. 2021; accepted 5 Feb. 2021. Date of publication 23 Feb. 2021; date of current version 18 July 2022. (Corresponding author: Emerson Murphy-Hill.)

Recommended for acceptance by A. Sarma.

Digital Object Identifier no. 10.1109/TSE.2021.3061527

1. Others [9], [10] have called this technique ‘blind’ code review, alluding to ‘blind reviewing’ of scientific articles. We avoid this term purposefully because ‘single-blind reviewing’ of scientific papers means that the author is not aware of reviewer's identity, whereas here we mean the reviewer is not aware of the author's identity. Some have also argued the metaphorical use of the word ‘blind’ in these contexts is ableist [11], [12].

anonymous author review, this undermines the effectiveness of anonymous author review. Analogously, prior research suggests that during double-blind paper academic paper review, reviewers can rarely guess author identities [17], [18]. We also investigate whether reviewers are less able to guess authors during *certain types of reviews*, making these types more amenable to anonymous author code review.

RQ2: How Does Anonymous Author Code Review Change Reviewers' Velocity? Speed of code review is a significant concern across companies and organizations that practice it [19], yet anonymous author code review may slow down the code review process, for example, when reviewers can't easily contact authors for high-bandwidth communications.

RQ3: How Does Anonymous Author Code Review Change Review Quality? Prior research suggests that double-blind research paper reviews are of equal [20], [21], [22], [23] or higher quality [24], [25] as single-blind reviews because reviewers are more critical when they are unaware of author identity [26]. We may expect similar quality effects for anonymous author code review.

RQ4: What Effect Does Anonymous Author Code Review Have on Reviewers' and Authors' Perceptions of Fairness? Fairness is both considered important by software engineers [27] and a central goal of anonymous author code review. While our field study is not well-suited to study fairness objectively, we take the approach of prior work [24], [28] to study it though the subjective perceptions of reviewers and authors.

RQ5: What do Engineers Perceive as the Biggest Advantages and Disadvantages of Anonymous Author Code Review? We next explore the tradeoffs involved in performing anonymous author code review, beyond the effects explored in the prior questions.

RQ6: What Features are Important to an Implementation of Anonymous Author Code Review? Understanding what features of an anonymous author code review tool is designed to help organizations decide how they might implement anonymous author code review.

In answering these questions, this paper contributes the first empirical study of anonymous author code review. While the motivation for this work is reducing bias during code review, in this paper we will not directly examine whether bias is actually reduced, which would likely require a larger-scale study than we performed here.

2 BACKGROUND: CODE REVIEW AT GOOGLE

Process. At Google, most of our code resides in one large repository. When an engineer wants to make a change, they:

- Create a changelist (CL) that contains diffs of one or more files, similar to pull requests on GitHub.
- Use the tool that facilitates this review process – Critique – that has the ability to display diffs, post and reply to comments about a CL, and display analysis results, such as code coverage and linter warnings.
- Choose one or more appropriate reviewers either manually or by getting a recommendation from Critique. If the author is not an owner of the code being changed, a reviewer must be an owner. The

majority of CLs are reviewed by someone on the author's team.

- Tell Critique to notify reviewers to begin their review. Reviewers add comments about the change and ask for further changes, if necessary.
- Make fixes and respond to comments about the change. The process of asking for changes and making changes may be repeated several times.
- Merge the changelist into the repository, once the requisite positive signal – a “looks good to me” or LGTM – is granted by reviewers.

More information about the review process at Google can be found in prior work [19].

Readability. Google has instituted a mandatory coding style and recommended best practices for each of the various languages in wide use, such as Java, C/C++, and Python. Additionally, for each language there is a process by which an engineer can demonstrate their knowledge of the best practices and agreement to enforce them. This process is known as readability. If a changelist author modifies a file in a language for which they do not have readability, the changelist must be approved by a reviewer with readability in that language. Linter tools catch most style violations automatically. Review for readability is primarily intended to ensure best practices are being followed, for example, in naming conventions, recommended APIs, modular design, and good testing practices.

Comparison to Other Code Review Processes. Sadowski and colleagues have compared the code review process with Critique at Google to code review processes elsewhere [19] using Rigby and Bird's convergent practices framework, which compares code review across multiple organizations [29]. Sadowski and colleagues conclude that Critique at Google is similar to other code review contexts insofar as it's a lightweight and flexible code review process, but the notion of explicit ownership and readability is unique. We also note that Critique supports asynchronous review with email notifications like other systems, but in contrast to systems like AMD's CodeCollaborator [30] and Microsoft's CodeFlow [31], Critique does not explicitly support live chatting between the author and reviewers.

3 METHOD

To answer our research questions, in early 2019, we built a browser extension that automatically hides information about the authors of changes from reviewers (Section 3.1), deployed the extension to volunteering developers at Google (Section 3.2), and collected (Section 3.3) and analyzed (Section 3.4) qualitative and quantitative data on code review experience and outcomes.

3.1 Implementation of Anonymous Author Code Review

While changing Critique would be the most seamless way to implement anonymous author code review at Google, a browser extension allowed us to evaluate the idea of author anonymous code reviews without any changes to critical engineering infrastructure. The extension we built removed authorship information from Critique, anywhere

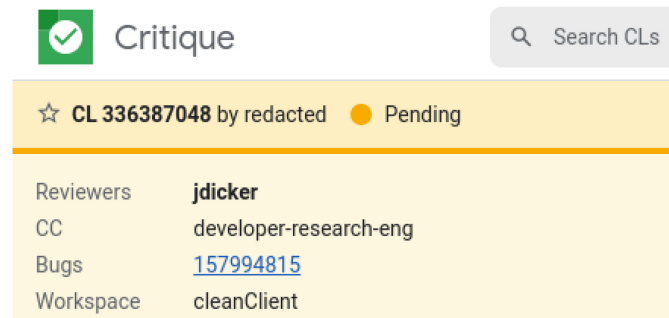


Fig. 1. What a reviewer (jdicker) using our extension sees at the top of Critique when reviewing.

the author's username appears; from our email client (Gmail), in Critique emails where the author's username or profile images appear; and in CL Monitor, another browser extension used by many engineers at Google that notifies them of incoming and outgoing reviews. Our browser extension also prompted participants for information about each CL they reviewed, directly after they provided LGTM. Fig. 1 shows a screenshot of how our extension removes the author's username from Critique. While the reviewer experience in Critique changes, authors see Critique as normal, with all usernames included.

In taking the browser extension approach, we were unable to remove author information from the following sources:

- Emails and notifications from a users' mobile device,
- Emails and notifications on non-Google-owned devices, and
- Bugs that link to the CL being reviewed.

Due to these limitations, we had to ask participants to create an email filter where Critique emails would skip their inbox and to refrain from looking at such emails on mobile and non-corp devices. We also told participants that our extension would not be able to remove author information in all contexts and to work around such issues. Participant instructions can be found in the Supplemental Material Sections 1 through 3, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TSE.2021.3061527>.

Finally, we anticipated that in some situations, participants would need to know an author's identity, so the extension had a button to deanonymize a CL in Critique.

3.2 Participants

We recruited participants (Table 1) by sampling engineers across Google who met the following criteria:

- Employed by Google for at least 6 months, in an attempt to avoid conflating the experience of anonymous author code review with overall novelty of Google's review process;
- Been on the same team for at least 3 months, in an attempt avoid conflating the experience of anonymous author code review with the novelty of being on a new team; and
- Reviewed at least 10 CLs in the two weeks before the study, in an attempt to ensure that the participant

TABLE 1
Statistics About Who Participated and the Number of Reports They Submitted

| | | | | |
|-------------------------------|--------------|---------|---------|--------|
| Engineers Invited | 1,650 (350R) | | | |
| Engineers Volunteered | 439 (128R) | | | |
| | Treatment | | Control | |
| Engineers Enrolled | 330 | (96R) | 109 | (32R) |
| Engineers Reporting on 1+ CLs | 300 | (90R) | 95 | (29R) |
| Submitted CL Reports | 5,217 | (1858R) | 1,935 | (682R) |
| Submitted Final Reports | 282 | (85R) | 92 | (29R) |

Numbers in parentheses indicate readability. Example: 439 engineers volunteered, of which 128 had readability. Example: 5217 total CLs were reviewed by the treatment group, of which 1858 were by readability reviewers.

would perform enough reviews during the study period to generate a sufficient volume of data.

Additionally, using stratified random sampling we aimed to have a subsample of participants—25 percent—be readability reviewers. When a readability review is requested, the readability reviewer is assigned randomly from a queue of already-certified developers, a queue that contains hundreds of potential reviewers for popular languages like Java or C/C++. Thus, we hypothesize that for such readability reviews, reviewers are unlikely to know the identity of authors. If that hypothesis is true, readability reviews are thus good candidates for anonymous author code review. To recruit readability reviewers, for this subsample we required engineers to have performed at least 2 readability reviews per week in the two weeks prior to the study.

One third of all volunteers were randomly assigned to a control group (stratifying on readability), which enabled us to compare data collected from engineers reviewing with and without anonymous author code review. The control group performed reviews in a standard, author-visible manner, but completed reports similar to those in the treatment (that is, anonymous author) group. A post-hoc analysis shows that the random assignment to control and treatment groups yielded statistically similar levels of seniority (median level 4, one level above entry-level, $p = 0.59$), though the control group had statistically significantly longer tenure (median 3.67 years for control versus 3.63 years for control), using a Wilcoxon rank sum test ($p < .001$). This motivates the use of covariate controls in our analysis (Section 3.4).

The study complied with human Google ethics guidelines for conducting research with human participants and underwent internal employee privacy review. Participants could terminate their participation in the study at any time.

3.3 Data Collection

3.3.1 Quantitative Data

Quantitatively, we used metrics from several sources. First, we logged when participants pressed the deanonymize button to answer RQ1, in part. Second, we used metrics derived from tool logs to collect active reviewing time as a measure of the reviewers' velocity (RQ2). Active reviewing time is the time a reviewer spends actively viewing, commenting, or working on a changelist, which may include time outside

of Critique in other tools, such as time spent looking up APIs or documentation. Further information on the active reviewing time metric can be found in our prior work [32]. Third, we use quantitative metrics to measure which changelists were rolled back in the 10 months after the code reviews took place. We posit that the higher the quality of the code review (RQ3), the lower the likelihood of rollback.

3.3.2 Qualitative Data

Qualitative data was collected to through two report types completed by participants:

- *CL Reports.* Our browser extension asked participating reviewers to fill out a report immediately after every LGTM, asking the participating reviewer to guess the author's identity (RQ1, treatment group), and what perceived effect the anonymization (or identity, for the control group) had on review velocity (RQ2), quality (RQ3), and fairness (RQ4).
- *Final Report.* Sent by email at the end of study period, a report asking participants about the reviewer experience during the study period, including review velocity (RQ2), quality (RQ3), and fairness (RQ4).² Questions also asked participants about overall advantages and disadvantages of anonymous author code review (RQ5) and desired features of an anonymous author code review system (RQ6).

Table 1 indicates how the number of reports participants filled out. The texts of each question will be described with results (Section 4), and blank reports are available in the Supplemental Material, available online, Sections 4 through 9.

3.4 Data Analysis

Analyses varied from research question to research question, but all statistical analyses were performed in R. R scripts were code reviewed by both an experienced software engineer and a quantitative analyst. Statistical tests were run with an alpha value of .05 to determine significant effects.

In every inferential statistical analyses in this paper, we used one of two types of regressions: review-level regressions (RLR) and participant-level regressions (PLR). Regressions allowed us to isolate effects of interest by controlling for covariates; for example, to estimate whether review time was different between groups, we controlled for changelist size and the seniority of the reviewer, among other variables listed below. Both RLRs and PLRs included the following fixed effects:

- *Group:* Whether the participant was in the control or treatment group.
- *Participating reviewer variables:* tenure (years at Google), seniority (level), role (individual contributor versus tech lead, etc.), job code (software engineer versus research engineer, site reliability engineer, etc), region (US West versus Latin America, US South, etc), and whether the participant was a

readability reviewer (binary). We controlled for tenure, seniority, and readability because they mediate code review pushback [33]; role and job code because different types of developers have different code review motivations and expectations [31]; and region because culture influences engineering practice [34].

RLRs included a random effect (reviewer identity) to account for individual variation from reviewer to reviewer. RLRs included these additional fixed effects:

- *Author variables:* The same variables as the participating reviewer (above), but for the author of the change (e.g., the tenure of the author).
- *Changelist variables:* the log of the number of reviewers, the CL size,³ whether the reviewer was a readability reviewer, and whether the CL changed "code". Here code means that at least one file is changed that's attributed to one of the known 40 coding languages at Google, including C++, Java, and Go. Examples of non-coding changes are those that exclusively change documentation, access control, and build management. Additionally, we included a binary "large-scale change" variable; these are relatively low-risk changes to a broad swath of our monolithic codebase (e.g., changing all uses of one API method to another). Large-scale changes are typically split up into multiple smaller CLs, where each CL is sent to appropriate code owners for review. We controlled for number of reviewers, size, and readability because they mediate code review pushback [33]; controlled for code changes because programming language correlates with pull request acceptance [2]; and controlled for large-scale changes because they are fundamentally different than other types of changes.
- *Relationship:* We modeled past interactions between reviewer and author as 'insider', 'outsider', or an 'unclear' relationship. An insider relationship means that the reviewer has reviewed 10 or more CLs at Google for the author prior to the CL in question; an outsider relationship means the reviewer has reviewed 2 or fewer CLs for this author previously; and unclear otherwise. We included this control variable because a similar notion of relationship in open source software correlates with pull request acceptance [2].

To convey a sense of overall model fit, we report adjusted R^2 values for Ordinary Least Squares regressions and adjusted McFadden's pseudo R^2 [35] for other regressions.

4 RESULTS

This section is structured as follows. In the next section, we describe to what extent participants and the CLs they reported on are representative of Google's engineering population. Then, starting in Section 4.2, we report on the answers to our research questions.

2. We additionally asked *authors* of CLs reviewed as part of the study (both treatment and control) about their perceptions about fairness of the review they received (RQ4).

3. The categories of code review sizes are: XS (0-9 lines changed), S (10-49 lines changed), M (50-249 lines changed), L (250-999 lines changed), XL (over 1,000 lines changed), and U (could not be calculated).

4.1 Participation and Representativeness

Invited engineers could decide to participate or not, and if they participated, decide whether to report on an individual changelist or not. This section reports on the extent to which those engineers who participated and those changelists that were reported on were representative.

4.1.1 Who Was Likely to Participate and Not Participate

To determine whether some invited engineers were more or less likely to volunteer in our study, we created a logistic PLR that predicts participation (adjusted McFadden pseudo- $R^2 = 0.18$). We found:

- Engineers with readability were twice as likely to participate as those without (Odds Ratio = 2.0, $p < .001$).
- Engineers who have been at Google for 7 or more years were less likely to participate than engineers who have been at Google for less than a year (OR = 0.33, $p < .001$).
- Engineers in the US South were more likely to participate than in the US West region, with 4 out of 6 invitees volunteering in the US South (OR = 6.8, $p = .042$).

No other predictors emerged as statistically significant.

4.1.2 Which Changelists Were Reported On

While we instructed participating engineers to report on their experience reviewing every CL during the study period, sometimes they did not. We ran a logistic RLR that predicted whether a CL was reported on (adjusted McFadden pseudo- $R^2 = 0.09$). While the raw median percentage of CLs reported on by control group participants was 88 percent, and the median was 84 percent for treatment group participants, this difference was not statistically significant ($p = 0.8$), controlling for other covariates. However, a variety of other covariates were associated with an increase in the odds that a CL was reported on (e.g., CLs from outsiders and readability reviews), while other covariates were associated with a decrease in those odds (e.g., more reviewers and large scale changes). These findings underscore the importance of using RLR regressions, which control for these covariates, in the remainder of this paper.

Finally, during the study, we asked participants to what extent the CLs they reviewed during the study were typical of their personal experience. The response distribution between the two groups was similar, but the treatment group was slightly more likely to perceive the CLs they reviewed as being typical. 71 percent of the control group rated this as “Very typical”, versus 77 percent of the treatment group. “Somewhat typical” was chosen by 25 percent of the control group versus 22 percent of the treatment group, and “Not at all typical” was chosen by 5 and 1 percent, respectively.

4.1.3 Participating Reviewers Per Changelist

In the changelists that were part of this study, most were reviewed by just one (55 percent of reviews) or two (29 percent of reviews) reviewers. 0.5 percent of reviews had

TABLE 2
How Often Participants Were Able to Correctly Guess the Identity of the Authors of the Changelists They Reviewed

| | Normal reviews | | Readability reviews | |
|-----------------|----------------|----------|---------------------|---------|
| Correct Guess | 76.6% | (n=3239) | 4.6% | (n=24) |
| Uncertain | 21.3% | (n=903) | 95.1% | (n=500) |
| Incorrect Guess | 2.1% | (n=88) | 0.4% | (n=2) |

multiple study participants as reviewers. 0.15 percent of reviews had a treatment group and a control group reviewer, so cross-contamination of results was a limited risk.

4.2 RQ1: How Often Can Reviewers Guess Author Identities?

Reviewers may know the identity of authors without being explicitly informed. In this section, we investigate how often reviewers with anonymous authors can guess author identity, by what means, and for what reasons.

4.2.1 Author Guessability Rates

To examine how often authors were guessable, we asked treatment group participants to guess author usernames after saying they were either “Very Certain”, “Somewhat certain”, or “Uncertain” of author identity. If they chose “Uncertain”, we did not ask them to provide a guess. We performed two data cleaning steps: we manually inspected and fixed incorrectly-spelled guesses, then removed any remaining guesses that didn’t correspond to any known Google engineer (e.g., blank responses). We next took the remaining cases and categorized them into correct guesses and incorrect guesses. Results are displayed in Table 2. Let’s first examine readability reviews, since we hypothesized that authors and reviewers were unlikely to know each other during these reviews because reviewers are assigned randomly. The data confirms this – in 95 percent of anonymous author readability reviews, the reviewer reported being uncertain of the author’s identity. Of the cases where the readability reviewer did know the author’s identity ($n = 24$), the most common reasons were the author contacting the reviewer outside of code review ($n = 8$, 33 percent) and the part of the codebase being changed ($n = 6$, 25 percent). With non-readability reviews, in 77 percent of cases reviewers guessed the author correctly. In 21 percent of cases the reviewers were explicitly uncertain of the author’s identity. In 2 percent of cases, the authors were at least somewhat certain of the author’s identity, but guessed incorrectly.

4.2.2 Author Guessability Mechanisms

We asked treatment group participants how they knew author identities. These participants had a set of 11 pre-defined options, but could also state their own reasons; participants chose to do so in 660 cases. One author of this paper coded respondents’ reasons and grouped them into several reason categories. Because reviewers could choose multiple reasons, the total number of

reviews sums to more than the total number of reports submitted.

The most common reason that reviewers with anonymous authors knew the identity of authors was for two contextual reasons ($n = 2845$, 53 percent). The first was that authors could determine the author from the part of the codebase that was being changed. The second was from the nature of the change, such as the language or programming style. As one participant stated, “The author has been doing a large number of similar migration CLs”.

The second most common reason was because the author and reviewer communicated ($n=1294$, 24 percent). This was usually because the author contacted the reviewer. One participant gave an example as “The author pinged our chat about fixing a test”. This also occurred when the author and reviewer collaborated prior to a change or could guess identity by virtue of being on the same team. For instance, one participant noted “I know who is working on this specific change from sprint planning/standup”.

The third most common reason was because of some limitation of our implementation of anonymous author code review ($n = 579$, 11 percent). This was usually because the description of the change indicated – either explicitly or implicitly – the author’s identity. For instance, the “design doc linked in the CL description” revealed author identity.

The fourth most common reason was through deductive means ($n = 259$, 5 percent), typically because a change did or didn’t need approval from an owner of the code being changed. This is because at Google, at least one reviewer must be an owner if the author is not an owner. Thus, if Critique indicates that an owner’s review is not required, then the author must be an owner, which narrows the set of potential authors.

Other less common reasons include from the knowing what tasks coworkers are performing prior to the review ($n = 161$, 3 percent), because some part of our tool failed ($n = 147$, 3 percent), because the reviewer decided to deanonymize the author ($n = 43$, 1 percent), and because the reviewer was in close enough physical proximity to happen to see the CL on the author’s screen ($n = 7$, <1 percent).

Some of the less frequent reasons for author deanonymization during anonymous author code review can be alleviated in a straightforward way (e.g., redacting bug assignee when browsing a linked bug). However, the top two most common reasons – the part of the codebase being changed and the nature of the change – appear more challenging to alleviate. The next most common set of reasons – author to reviewer communication – may be alleviated in part by building anonymous communication, such as through anonymous messaging. If anonymous messaging between an author and reviewers were retained alongside a CL, side benefits might accrue, such as the retention of design rationales contained in author-reviewer discussions.

4.2.3 Why Reviewers Need to Know Author Identities

When treatment group participants filled out a report on their experience after a code review during the study, our chrome extension inserted logs about whether the participant explicitly requested to deanonymize the CL. In total, 47 reports with logs indicated that the participant requested

deanonymization, compared to 5,112 reports with logs where the participant did not.

To determine why this rare event occurred, we asked participants “If you [deanonymized] the CL using the extension, why did you do so?” We qualitatively coded the 53⁴ open ended responses into 10 types of reasons, with some responses containing multiple reasons.

- A set of common deanonymization reasons were about the context of the CL. For example, one participant stated, they were “curious about the reason of the change.” In $n = 10$ (19 percent) cases, the reviewer needed additional context to review the change that wasn’t contained in the original CL. In $n = 5$ (9 percent) cases, the reviewer may have needed to supply additional context about the change to the author, depending on the author’s identity. In $n = 8$ (15 percent) cases, the author needed to be made aware of other people relevant to this change, and the reviewer did not know if the author is one of these people. Similarly, in $n = 3$ (6 percent) cases, the reviewer needed to assess the level of knowledge the author has about the context.
- In $n = 7$ (13 percent) cases, the reviewer needed to know whether the author had implicit permission to make a change, including changes to access control lists (ACLs) and TODOs. As one participant stated, “CL author added themselves to OWNERS file. Diff showed addition as “redacted.” I needed to know who is being added to the OWNERS file.”
- In $n = 6$ (11 percent) cases, the reviewer mentioned deanonymization because they needed to contact the author.
- In $n = 5$ (9 percent) cases, our extension inadvertently redacted a relevant URL in Critique, so deanonymization was required to click through the URL.
- In $n = 2$ (4 percent) cases, the reviewer deanonymized to determine who made the change so that they could thank the author.
- In $n=2$ (4 percent) cases, the reviewer indicated simply being curious about the author’s identity.
- In $n=13$ (25 percent) cases, it was unclear from the reports why the reviewer needed to deanonymize the CL.

As the list above suggests, out-of-Critique communication between reviewers and authors occurs for some CLs. In the post-LGTM report, we asked respondents, “If you contacted the author or the author contacted you outside of Critique, why?” We received 1,009 responses to this question.

Authors were about four times more likely to contact reviewers than reviewers contacting authors. Although it was sometimes difficult to tell who initiated contact, based

4. The six-report discrepancy between the 47 logs indicating deanonymization and 53 open ended answers indicating deanonymization could be due to either our extension failing to log events correctly or to participants misinterpreting the question. Manual inspection of the open ended responses suggests a likely combination of these two. Similarly, the discrepancy between the 47 logs indicating deanonymization and the 43 reports where participants said they knew authors’ identity because they pressed the deanonymize button, may be explained by four engineers either forgetting that they deanonymized the review or forgetting author identity after they deanonymized.

on our reading of the responses, in 59 percent percent of cases authors contacted reviewers; in 14 percent of cases reviewers contacted authors, and in 27 percent of cases it was unclear.

It was clear that a substantial number of responses were largely about how the reviewer knew the author's identity, rather than about the *reason* behind the contact. Including only answers to the latter ($n = 691$), several themes emerged:

- The most common theme ($n = 211$, 31 percent) was that the author contacted the reviewer at the start of the review to introduce the CL, request a review, offer some up-front context, or ask for a swift review due to the urgency of the CL. For example, one participant noted, "The author promised me to deliver a fix for a tool. Later I saw a fix for the said tool. Inferred the author's identity."
- The second most common theme ($n = 185$, 27 percent) was making contact to discuss a CL or address/clarify a comment made during the review. For instance, one participant commented, "To ask follow-up questions on the comments I've left in Critique."
- The third most common theme ($n = 107$, 15 percent) was to ask the author or reviewer a question related to the review. As one participant noted, "He asked me questions about...the tests he was going to change, as he wasn't sure of how they worked."
- Other reasons include the author or reviewer mentioning their preference for a conversation outside of Critique due to the relative speed or ease as compared to going back and forth in comments ($n = 45$, 7 percent); the author contacting the reviewer to remind them about the CL awaiting review ($n=35$, 5 percent); contact made convenient by close physical proximity of author and reviewers ($n=33$, 5 percent); expediting a CL rollback ($n=15$, 2 percent); mentioning the change fixing something that's broken or breaking something ($n = 11$, 2 percent); asking for clarification around the readability process ($n=10$, 1 percent); and needing permission to access a shared resource ($n=6$, 1 percent).

4.3 RQ2: How Does Anonymous Author Code Review Change Reviewers' Velocity?

We measured the effect on review velocity through both qualitative and quantitative measures.

During the course of the study, we asked the following after a participant LGTM'd a CL:

- Treatment participants were asked "what effect did the [anonymous author] code review process have on reviewing velocity, compared to [non-anonymous] code review?" For 89 percent of CLs, participants said anonymous author code review did not change velocity; for 6 percent anonymous author review had a negative impact on velocity; for 2 percent a positive impact; and the rest were not known.
- Conversely, we asked control participants to estimate the effect of knowing the identity of the author compared to not knowing their identity in

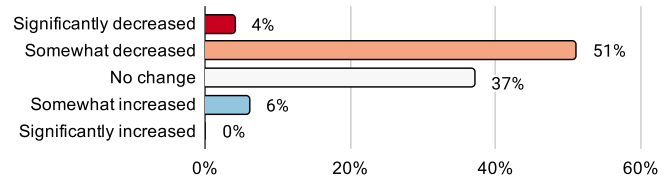


Fig. 2. Treatment participants' expectations about what effect anonymous author code review would have on review velocity if practiced regularly at Google. An additional 2 percent of participants responded "I don't know".

terms of review velocity. Here, for 69 percent of CLs the reviewers said knowing the identity had no effect; for 3 percent knowing identity decreased velocity; for 26 percent knowing identity increased velocity.

After the study, we asked treatment participants "what [effect on velocity] did the [anonymous author] code review process have, compared to [non-anonymous] code review?" 59 percent said anonymous author code review had no effect on velocity; 5 percent said a positive effect on velocity, 36 percent said a negative effect, and the remainder did not know.

We also wanted to know the impact of anonymous author code review on velocity more broadly, so we asked all participants, "If [anonymous author] code review was regularly practiced at Google, I expect that my engineering velocity would be...", followed by a rating. As shown in Fig. 2, most (55 percent) treatment group participants thought that anonymous author code review would have a negative effect on review velocity, with most of the remainder of participants (37 percent) expecting no effect.

Overall, this data suggests that *participants perceived that anonymous author code review generally has a negative or neutral effect on code review velocity.*

Although perceived review time is important, we can also measure active reviewing time as an objective measure of velocity. To do so, we compared not only the control and treatment groups during the study, but also those groups against their own review velocities pre- and post-study. Thus, we separated reviews into those performed in the approximately two weeks during the study, in the two-week period before the study began, and in the two-week period after the study ended. We defined study beginning and ending on a per-participant basis, since participants may have begun using our chrome extension or filled in the final report at any point after we invited them to do so. We defined the study beginning for a participant as the time when the participant made the first comment on a CL that they filed a report for. We defined the study ending for a participant as the time when the participant submitted their last report.

Fig. 3 illustrates raw reviewing time differences between the control group and treatment group, and in the pre-study ($n_{\text{control}} = 2351$ and $n_{\text{treatment}} = 7612$ changelists), during-study ($n_{\text{control}} = 2169$ and $n_{\text{treatment}} = 6308$ changelists), and post-study ($n_{\text{control}} = 1702$ and $n_{\text{treatment}} = 6149$ changelists) periods. In the figure, we show boxplots that summarize reviewing time distributions, overlaid with individual code review times as circles. We do not display any code reviews that took more than about 30 minutes, but these are accounted for by the boxplots.

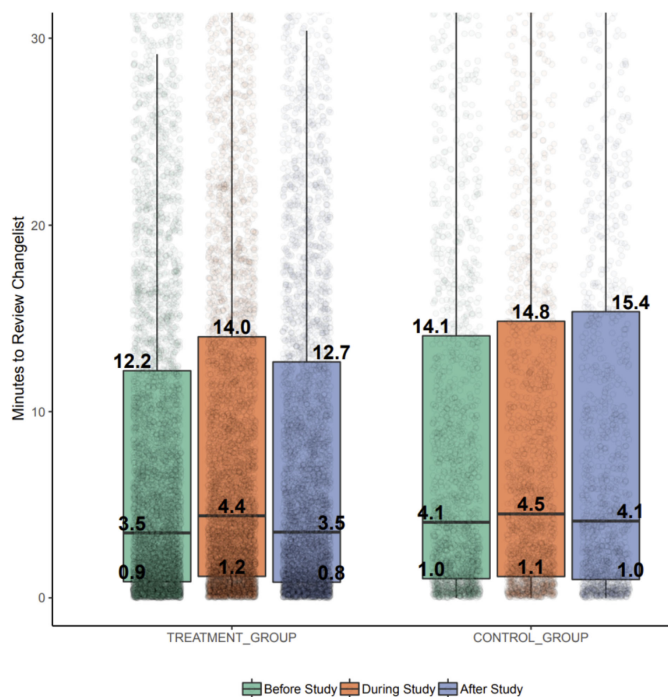


Fig. 3. The time participants spent actively reviewing each changelist in minutes, comparing treatment and control groups in terms of review time before (left), during (center), and after the study (right).

This raw data suggests that reviewing time increased during the study period for both control and treatment groups – from 3.5 to 4.4 minutes for the treatment group, and from 4.1 to 4.5 for the control group. So was review time lengthened by anonymous author review? To examine this, we created a linear RLR predicting log review time (adjusted McFadden pseudo- $R^2 = 0.07$). We use log review time since review time is highly skewed (most reviews take less than 5 minutes, but some reviews can take 30 minutes or more). In addition to the existing linear RLR covariates, we add a three-level fixed-effect for time period (before, during, and after the study).

This RLR indicates that, indeed, changelist review time was longer during the study than before the study (by 10 percent, $p < .001$) and after the study (by 6 percent, $p < .001$). However, this RLR also indicates that the time taken to review in the treatment (anonymous author) group was not statistically significantly different than the time taken in the control group ($p = .346$).

Why might both the control and treatment group show an increase in reviewing time during the study? Potential explanations include both groups of participants being influenced by modulating behavior simply by being aware of identity-issues and both groups changing their behavior due to being part of a study [36].⁵

In sum, because both control and treatment groups reviewed more slowly during the study to a statistically similar extent, our results suggest that *anonymous author code review did not substantially change objective reviewing velocity during the study, compared to non-anonymous review, in our study setting.*

5. The increase is not due to filling out the post-LGTM report, as review time does not include the time to fill out this form.

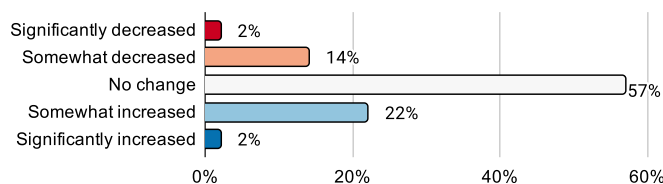


Fig. 4. Treatment participants' expectations about what effect anonymous author code review would have on review quality if practiced regularly at Google. An additional 1 percent of participants responded "I don't know".

This conclusion applies to anonymous author code review in practice at companies like Google, that is, when author identities can often be guessed (Section 4.2). Other contexts may exist where guessability is lower, such as open source projects that accept pull requests from a wide variety of external contributors; in these contexts, would we expect different reviewing velocities? We answer this question by comparing normal treatment group reviews where the reviewer knew the author identity to those where the author's identity was not known. A detailed description of this analysis is in the Supplemental Material Section 12, available online; in short, we find that there exists fundamental differences in code reviews when the author is guessable compared to non-guessable. Therefore, comparing guessable to non-guessable author code reviews on metrics like velocity or quality is not a reasonable comparison, so we make no further attempt to do so in the remainder of the paper.

4.4 RQ3: How Does Anonymous Author Code Review Change Review Quality?

Given the effect of code review on software quality [37], [38], [39], [40], we asked how participants perceived their review quality during and after the study.

After participants LGTM'd a CL:

- We asked treatment participants to rate what effect the anonymous author code review process had on code quality, compared to non-anonymous code review. For 92 percent of CLs, participants said anonymous author code review had no impact on quality; for 1 percent anonymous author review had a negative impact on quality; for 4 percent a positive impact; and the rest were not known.
- Conversely, we asked control participants to estimate the effect of knowing the identity of the author compared to not knowing their identity in terms of review quality. Here, for 79 percent of CLs the reviewers said knowing the identity had no effect; for 7 percent knowing identity decreased quality; for 12 percent knowing identity increased quality.

After the study, we asked treatment participants "what [effect on quality] did the [anonymous author] code review process have, compared to [non-anonymous] code review?" 75 percent said anonymous author code review had no effect on quality; 18 percent said a positive effect on quality, 7 percent said a negative effect, and the remainder did not know.

We examined the impact of anonymous author code review on quality more broadly, so we asked all participants, "If anonymous author code review was regularly practiced at Google, I expect that..." As Fig. 4 shows, most

(57 percent) treatment group participants thought that anonymous author code review would be neutral toward review quality, with most of the remainder of the treatment group participants (24 percent) expecting positive impact on quality.

Finally, we ran a logistic RLR to predict whether the changelists reviewed by treatment participants were more or less likely to be rolled back in the roughly 17 month period after the experiment took place, one measure of review quality (adjusted McFadden pseudo- $R^2 = 0.0008$). We found no statistically significant difference in the odds of rollbacks, comparing the treatment CLs reviewed during the study period compared to the CLs in the two week period before the study ($p = .71$) and the two week period after the study ($p = .46$). However, surprisingly, control group participants' changelists saw a slight but statistically significant rise in the odds of a rollback for CLs reviewed during the study period compared to the pre-period (Odds Ratio 1.02, $p < .001$) and post-period (OR 1.01, $p < .001$). This evidence suggests that review quality is at least as good during anonymous author code review as during non-anonymous review.

Overall, this data suggests that *anonymous author code review generally has a neutral effect on code review quality*.

4.5 RQ4: What Effect Does Anonymous Author Code Review Have on Reviewers' and Authors' Perceptions of Fairness?

At its core, the purpose of anonymous author code review is to allow engineers to review code without being burdened by any biases they might have about the author. Thus, we examined whether code reviewers (Section 4.5.1) and authors (Section 4.5.2) were able to perceive any differences in how reviewers treated authors during anonymous author code reviews, specifically from a fairness perspective.

4.5.1 Reviewer Fairness Perceptions

We first asked reviewers whether they perceived any differences in their own fairness while performing anonymous author code reviews. After participants LGTM'd a CL:

- We asked treatment participants to rate what effect the anonymous author code review process had on reviewing fairness, compared to non-anonymous code review. For 93 percent of CLs, participants said anonymous author code review did not change fairness; for 4 percent anonymous author review made the review more fair; for < 1 percent, less fair; and the rest were not known.
- Conversely, we asked control participants to estimate the effect of knowing the identity of the author compared to not knowing their identity in terms of review fairness. Here, for 89 percent of CLs the reviewers said knowing the identity had no effect; for 3 percent knowing identity decreased fairness; for 2 percent knowing identity increased fairness; and the rest were not known.

After the study, we asked treatment reviewers "what [effect on fairness] did the anonymous author code review process have, compared to [non-anonymous] code review?"

66 percent said anonymous author code review had no effect on fairness; 31 percent said a positive effect on fairness, 1 percent said a negative effect, and the remainder did not know.

Overall, this data suggests that *most reviewers perceived that anonymous author code review has a neutral or positive effect on code review fairness*.

4.5.2 Author Fairness Perceptions

We also asked code review authors if they perceived any fairness differences. Note that we don't necessarily expect an increase or decrease in perceived fairness. On one hand, if authors perceived previous non-anonymous reviews as unfair, then they may perceive anonymous author review as more fair than they expected. On the other hand, authors who previously had been treated with undue deference might perceive anonymous author reviews as less fair than they expected.

After reviews were completed by study participants, we asked authors of those CLs "During code review for this changelist, did you experience being treated as fairly as you expected?" We asked authors about the following CLs:

- All control group CLs. These represent CLs when the reviewer is aware of author identity.
- Treatment group CLs where the reviewer indicated that they were "Uncertain" of the author's identity. These represent CLs where the reviewer did not have knowledge of the author's identity.

We also surveyed authors once, and only once, to reduce survey fatigue. We did not indicate in this author survey that the reviewers were participating in an experiment, nor whether they were reviewing with anonymous authors (treatment group) or non-anonymously (control group). Moreover, study participants were discouraged from discussing the study with other Google engineers, so as to avoid biasing authors.

We separately analyze reviews where the study participant was the readability reviewer. Table 3 shows authors' fairness ratings.

The 'Normal reviews' columns shows that for most reviews in both control and treatment groups, authors did not feel treated more or less fairly than they expected (93 percent of control group authors, 94 percent of treatment group authors). The control and treatment group were not significantly different ($p = .97$), according to a linear RLR (adjusted McFadden pseudo- $R^2 = -2.6$) that predicts perceived fairness, where "less fair" is coded as -1, "more fair" is coded as 1, and otherwise as 0.

The 'Readability reviews' columns tell a curious story; control group participants' readability reviews were perceived as more fair than those from the treatment group. The interaction between control/treatment group and readability is statistically significant ($p = .002$), according to the same RLR.

Since we were surprised by this finding, we evaluated the hypothesis that the control group participants gave inordinately fair reviews.⁶ We asked authors who received readability reviews from reviewers not invited to be part of the study

6. Our analyses do not support three alternate hypotheses; see Supplementary Material, Section 13, available online.

TABLE 3
How Authors of Changelists Perceived Fairness of Normal and Readability Reviews From Developers in the Control Group, the Treatment Group, and a Set of Non-Participants After the Study

| | Normal reviews | | Readability reviews | | |
|-----------------------------------|------------------|------------------|---------------------|------------------|------------------|
| | Treatment | Control | Treatment | Control | Post Study |
| More fairly than expected | 5.6% (n=23) | 6.6% (n=29) | 6.5% (n=17) | 17.1% (n=24) | 9.5% (n=33) |
| About the same / don't know / n/a | 93.6% (n=383) | 92.7% (n=406) | 92.0% (n=242) | 82.9% (n=116) | 89.7% (n=312) |
| Less fairly than expected | 0.7% (n=3) | 0.7% (n=3) | 1.5% (n=4) | 0% (n=0) | 0.9% (n=3) |

to report the fairness of the reviews that they received. Results are shown in the 'Post Study' column in Table 3.

These readability CLs reviewed by non-participants show levels of author-perceived fairness slightly closer to the treatment group than the control group. An Ordinary Least Squares regression with the same fixed effects as an RLR predicting perceived fairness (adjusted $R^2 = .16$) provides more definitive evidence of this: non-participant CLs are significantly different than control group CLs ($p < .001$) but not significantly different from treatment group CLs ($p = .38$), in terms of author-perceived fairness.⁷ In other words, control group CLs appear to be the outliers here. The hypothesis that the control group gave fairer reviews than the treatment group – rather than the treatment group giving particularly unfair ones – is supported.

Overall, this data suggests that *authors who unknowingly received anonymous author reviews did not perceive a significant difference in fairness, compared to authors who received non-anonymous reviews.*

4.6 RQ5: What do Engineers Perceive as the Biggest Advantages and Disadvantages of Anonymous Author Code Review?

4.6.1 Observed Benefits

We asked treatment group participants "What was the main benefit you experienced during this study while performing [anonymous author] code review?" One author qualitatively coded each of the 198 responses into 10 emergent categories.

The definitions of these categories are:

- *No personal benefits / not sure.* The most commonly mentioned benefit was none that was perceptible to the participant ($n = 71$, 36 percent). To quote one participant, "It is hard for me to find a clear benefit as I nearly always knew who I was reviewing (due to the change being made)."
- *More thorough review without reliance on author identity.* The second most commonly mentioned benefit was that reviews were more thorough ($n=65$, 33 percent), because the reviewer could not rely on a high

level of author expertise and consequently conducted their reviews in a more neutral way, independent of the authority of the author, the relationship between the reviewer and the author, or trust based on prior experiences with the author. Many comments mentioned the necessity of taking a closer look without being able to assume the author possessed certain knowledge, as well as feeling more empowered to leave feedback without regard to whether it was appropriate given the author's status. To quote one participant, "When I didn't know for sure that an author was a subject matter expert on a CL, it made me pay more attention to the content than I would have otherwise."

- *Reduce bias.* Another commonly mentioned benefit was a reduction in bias towards the author ($n = 45$, 23 percent). A specifically mentioned category of reducing bias was *reducing level/tenure bias* ($n = 8$, 4 percent). Note that there was a high overlap between the benefits of "reduce bias" and "more thorough review without reliance on author identity," and the latter benefit can be considered a form of reduced bias. During analysis, only comments that explicitly mentioned the terms bias and fairness or the concept of treating all CLs equally were tagged with the theme "reduce bias". To quote one respondent, "I think that I (and I would daresay most reviewers) probably apply some bias to CLs based on various attributes we attach to the author (tech level, tenure at Google, readability, role, past interactions or discussions with the authors, etc.). In some these cases, the biases serve as a bit of a short-circuit and I'm more willing to give an LGTM to someone that I "trust" more... even if I don't fully understand the content of the change."
- *Other benefits.* Several participants mentioned experiencing other benefits, including the review process being quicker or simpler when they didn't have to consider additional context related to the author's identity ($n = 20$, 10 percent); special value for readability reviews where reviewers and authors have little relationship ($n = 17$, 9 percent); enjoying and reflecting on a different kind of review process ($n = 12$, 6 percent); needing to provide better documentation and context in a CL ($n = 9$, 5 percent);

7. A caveat of this model is requires the omission of the reviewer random effect, because all non-participant CLs were reviewed by different reviewers. Use of a fixed-effect model here requires us to relax the assumption of independence.

encouraging promptness (n = 4, 2 percent); and removing CL notifications from email (n=3, 2 percent).⁸

4.6.2 Observed Drawbacks

As with benefits, we also asked treatment group participants to state the main drawback they observed during the study of anonymous author code review. In the 233 responses, the most commonly observed themes were:

- *Lack of context to aid in decisions.* This refers to additional information that can be gleaned from knowing an author's identity, which reviewers stated would have aided in a swifter or more tailored review (n = 111, 48 percent). Examples of such contextual information include the background and motivation behind the CL, the time zone of the author, the author's familiarity with the part of the codebase being modified, and the CL's level of urgency. One participant said, "Often, CLs come in with context that was chatted about offline or over other channels (bugs, emails) and knowing who authored the CL helps trigger the context of the CL."
- *Barriers to offline communication.* Similarly, this category refers to the need for reviewers and authors to communicate outside of a CL to quickly convey information (n = 48, 21 percent). One participant said, "At times, it is useful to have an in person conversation to clarify a point or to make a design decision more quickly than trading back and forth over critique. [Anonymous author code review] made that process a little bit harder."
- *Minimal or no impact/often knew author.* This category describes minor or no effects of anonymous author code review, often because the reviewer could guess the author (n = 45, 19 percent). One participant said, "I'm able to identify the author for most of the CLs anyway, making this experience less efficient to me."
- *Reduced velocity.* Some participants reported that their reviews took longer when reviewing with anonymous authors (n=35, 15 percent). As one participant said, "Because of the increased fairness and quality, reviews took longer."
- *Inconvenience/performance issues with extension or reports.* This theme refers to the limitations of the study that were drawbacks (n = 30, 13 percent), such as the extension redacting many usernames in error or slowing down Gmail inordinately. One participant said, "Not really a drawback of the process, per se, but the aggressiveness with which the Chrome extension redacted names sometimes made it hard to get additional information". A similar theme was *lack of email notifications*, which was a limitation required by our chrome extension (n = 13, 6 percent).
- *Other drawbacks.* Participants reported other drawbacks, including reviewers being unable to tailor

feedback to the background of the author, such as giving detailed explanations to new employees (n = 21, 9 percent); not being able to distinguish core contributors from external contributors (n = 18, 8 percent); not being sure or stating non-drawbacks (n = 12, 5 percent); reducing personal interactions with peers (n = 12, 5 percent); being less knowledgeable about colleagues' work (n = 11, 5 percent); challenges in looking back over old CLs without seeing author names (n = 8, 3 percent); awkwardness around the code review process (n = 7, 3 percent); and difficulty in prioritizing important CLs over less important ones (n = 6, 3 percent).

4.7 RQ6: What Features are Important to an Implementation of Anonymous Author Code Review?

At the end of the study, we asked participants about a set of features that anonymous author code review might have, if implemented at Google. The top of Fig. 5 shows a variety of features that we asked participants to rate the importance of. Of these features, participants were most strongly positive about the ability to deanonymize author the author during review, with 52 percent of participants noting the feature as essential. Most respondents rated the following features as at least worthwhile: the ability of the author to request anonymous author review, making the reviewer anonymous to author identity when no specific reviewer is required (e.g., when using a reviewer queue), showing which reviewers are reviewing with anonymous authors, and deanonymizing the author after granting LGTM. Showing which reviewers deanonymized a CL was rated positively by some respondents (27 percent), but also was rated as unwise by 19 percent of participants.

We also asked about what information should be revealed about the author, if the author's identity were anonymized, as shown in the middle of Fig. 5. This figure indicates a variety of opinions, overall it indicates that most participants support revealing author time zones, whether the author is on the reviewer's team, and whether the author has readability. Author tenure and level were the two types of information that participants were more likely to believe would be unwise to show than worthwhile or essential.

We also asked "Suppose [an anonymous author] review system is opt-in, where those who opt-in perform [anonymous author] review by default, but reviewers can [deanonymize] as they see fit. How important is it to allow opt-in to be chosen by each of the following?" Results at the bottom of Fig. 5. The results indicate participants were positive about having individual reviewers and teams opt-in, mixed about whether product areas (PAs) opt-in, and somewhat negative about having the entire company opt-in.

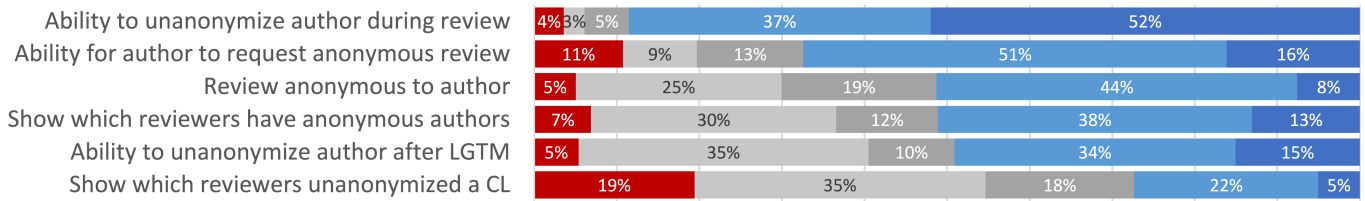
5 LIMITATIONS

Readers should consider several limitations of this study while interpreting its results.

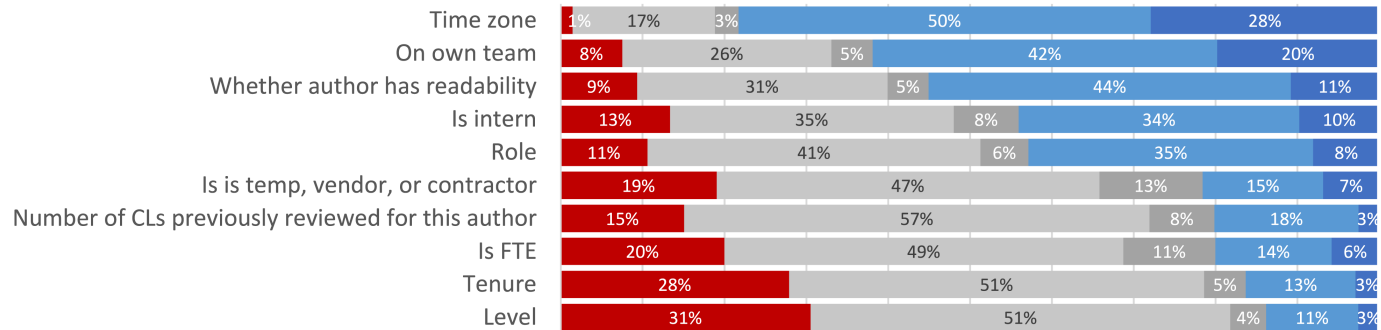
- Results obtained from a similar study, run in another organization or an open source software project, may differ.

8. "Removed CLs from the inbox" was a requirement of the study that participants should have Critique emails bypass their inbox. This would not be a constraint in a full implementation of anonymous author code review.

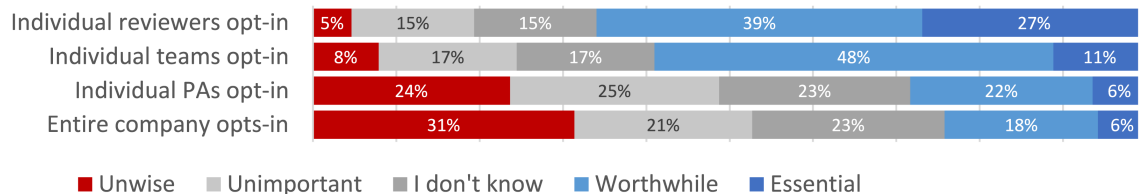
Importance of features



Importance of showing information about the author



Importance of who should opt-in



■ Unwise ■ Unimportant ■ I don't know ■ Worthwhile ■ Essential

Fig. 5. How participants rated the importance of various potential features of an anonymous author code review system.

- Although we randomly selected participants, those who opted to participate likely do not represent the full population of developers, both within Google and beyond.
- Participants used anonymous author code review over a relatively short period of time. As we describe in Section 6, the effect of anonymous author code review may differ if practiced over a longer period.
- Participants' self-reports are likely influenced by cognitive biases and by question wording. For instance, one question asked authors to rate fairness relative to their expectations as a baseline and another asked reviewers to rate fairness relative to a hypothetical counterfactual baseline. Both baselines makes interpreting the answer challenging.
- While we reported the number of times various themes from our qualitative data were mentioned, these numbers may not capture the frequency or severity. Rather, they likely capture how salient or memorable the theme was for participants.
- Given the number of open-ended responses and the amount of work required, we opted to have just one coder categorize the open-ended themes. More robust themes may have emerged with more coders.

- While this paper examined the effects of anonymous author code review in several dimensions, the practice may yet have other effects not fully explored in this study, effects such as relationship building and knowledge sharing.

6 DISCUSSION

We found that for about 77 percent of non-readability reviews, developers performing anonymous author review correctly guessed the author's identity. Rather than concluding that blind code review doesn't work or is impossible in practice, we instead argue that guessability is simply the unavoidable reality of contemporary code review. Nonetheless, our results suggest that "low context" changes are less guessable than other changes, and thus may be particularly effective places for implementing anonymous author review; such low context changes include readability reviews, large scale changes, and small changes. But even for reviews where authors are guessable, there may yet be benefits to anonymous author code review. First, as McKinley argues of double blind paper review, "the very act of omitting author details on the paper...reminds reviewers that they should judge the paper on its merits

rather than based on whomever they guess the authors might be" [41]. Second, we hypothesize that if author information is regularly hidden during code review, it may become less prominent in reviewers' minds (this was not the case in this study, since we frequently prompted developers to guess author identity). Third, we hypothesize that even if reviewers are 95 percent certain of author identity, the remaining 5 percent of doubt will be enough to change one's behavior. These hypotheses could be evaluated in future studies.

One finding from the study was that code reviews took longer, for both the treatment group and the control group. While we interpret this to mean that anonymous author code review *per se* did not cause the increase in reviewing time, it is still possible that anonymous author code review could nonetheless increase reviewing time in a non-experimental field deployment. If the time increase is due to a novelty effect [42], we would expect active review time to come back down after the novelty wears off. On the other hand, anonymous author code reviewers might take more time to review because the nature of their feedback changes, for example by being more explicit, which may in turn yield higher quality reviews. Which of these possibilities will materialize in practice requires further study in a larger scale but less experimental (e.g., no reports) study.

We found that fairness was, for most reviews, not perceptibly different from reviewers' and authors' expectations. We do not interpret this to mean that anonymous author code review cannot make reviews more fair, but rather that any changes in fairness were not perceptible to the group of respondents as a whole. We instead argue that anonymous author code review is more fair by construction, because irrelevant personal details are excluded, to the extent possible, from the decision making process.

For organizations that choose to implement anonymous author code review, we have several recommendations based on our results. The first is that the downsides appear minor for implementing author-anonymous review on low-context changes, where the reviewer has limited contextual information about the change prior to review. In open source, for instance, pull requests from newcomers might be a low-risk place to start implementing anonymous author code review. In terms of which features to implement for anonymous author code review, we recommend:

- Implement a break-the-glass option for revealing author identity, which participants were strongly in favor of. The downsides to this option are minimal; our results suggest the feature is rarely used, and when it is, it's used largely for important reasons such as understanding who's making access control changes.
- Implement displaying author time zone information, which participants were also strongly in favor of. This allows developers to make informed decisions about when to review.
- Reveal author information by default after LGTM is granted. This would allow developers to maintain familiarity with their colleagues' work, which was a downside of anonymous author code review as implemented in this study.

7 RELATED WORK

Studies on the impact of code review have become increasingly popular. Microsoft [31] found that developers report using code review not only for finding bugs, but also for knowledge transfer. Developers reported that they provided more useful and detailed feedback when they were also experts in the code being reviewed, as they had relevant context for the review. Sadowski and colleagues also found that at Google, there were similar practices around using code review for knowledge transfer, and that reviewers with context provided valuable feedback to the author [19]. A later study at Microsoft confirmed that a reviewer provided more useful comments when they had prior experience with the code being reviewed, though there was no difference based on whether the reviewer and author were on the same team [43]. This is particularly important for the effectiveness of anonymous author code review: reviewers provide more useful comments if they are familiar with the code, but our study suggests that reviewers who are familiar with the code may be more likely to successfully guess the author's identity.

Prior work has shown that in open-source code reviews, female authors are less likely to get their patch accepted when their gender is known [2]. Additionally, German and colleagues have found that fairness is a concern in open-source code reviews [27]. German and colleagues describe four types of fairness relevant to code review: distributive fairness, procedural fairness, interaction fairness, and information fairness. Anonymous author code review addresses distributive fairness by improving the equity and equality in how authors are treated in code review. Anonymous author code review also adds procedural fairness as it is a form of bias suppression.

While there is little prior work on anonymous author code reviews, there is significant prior work in anonymization in the peer-reviewed research papers. Prior work has found that anonymizing the author and affiliations from the peer reviewers increases representation of female authors [44] and also increases the representation of less-famous authors and authors from lower-prestige institutions [8]. These results have also famously held in areas outside of peer reviewed publications, such as in orchestral auditions [7] and in hiring [45].

There has been some prior work in non-technical factors that impact the outcome of code reviews. Baysal and colleagues examined several such factors, including the author's prior experience [46]. They found that prior experience improved the likelihood that a patch is accepted, and reduced the time spent in code review. However, it is not known whether this is a direct result of the author's prior experience or an artifact of bias on the part of the reviewer. Kononenko and colleagues also examined the impact of an author's prior experience on the quality of the review and found that it does not affect review quality [47].

8 CONCLUSION

While in principle anonymization reduces bias during code review by removing decision-irrelevant information (e.g., when an engineer has gender biases, removing identity removes the most salient gender signal), the principle is

threatened by practical considerations, such as the ability of reviewers to guess author identity. In this paper we described a field study of anonymous author code review, building an understanding of its benefits and drawbacks. Building on this increased understanding, we encourage researchers to investigate whether anonymous author code review reduces disparities in code review outcomes, such as the gender gap in pull request acceptance rate [2].

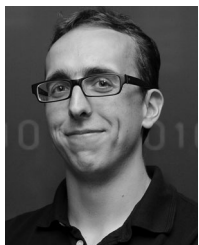
ACKNOWLEDGMENTS

The authors would like to thank all engineers who volunteered, without whom this study would not have been possible. They would also like to thank Danny Berlin, Ash Kumar, Ambar Murillo, and Rachel Potvin.

REFERENCES

- [1] G. Gousios, A. Zaidman, M.-A. Storey, and A. van Deursen, "Work practices and challenges in pull-based development: The integrator's perspective," in *Proc. 37th Int. Conf. Softw. Eng.*, 2015, vol. 1, pp. 358–368. [Online]. Available: /pub/pullreqs-integrators.pdf
- [2] J. Terrell *et al.*, "Gender differences and bias in open source: Pull request acceptance of women versus men," *PeerJ Comput. Sci.*, vol. 3, 2017, Art. no. e111.
- [3] H. K. Davison and M. J. Burke, "Sex discrimination in simulated employment contexts: A meta-analytic investigation," *J. Vocational Behav.*, vol. 56, no. 2, pp. 225–248, 2000.
- [4] J. H. Greenhaus, S. Parasuraman, and W. M. Wormley, "Effects of race on organizational experiences, job performance evaluations, and career outcomes," *Acad. Manage. J.*, vol. 33, no. 1, pp. 64–86, 1990.
- [5] R. A. Gordon and R. D. Arvey, "Age bias in laboratory and field settings: A meta-analytic investigation 1," *J. Appl. Soc. Psychol.*, vol. 34, no. 3, pp. 468–492, 2004.
- [6] M. Hosoda, E. F. Stone-Romero, and G. Coats, "The effects of physical attractiveness on job-related outcomes: A meta-analysis of experimental studies," *Personnel Psychol.*, vol. 56, no. 2, pp. 431–462, 2003.
- [7] C. Goldin and C. Rouse, "Orchestrating impartiality: The impact of 'blind' auditions on female musicians," *Amer. Econ. Rev.*, vol. 90, no. 4, pp. 715–741, Sep. 2000. [Online]. Available: <http://www.aeaweb.org/articles?id=10.1257/aer.90.4.715>
- [8] A. Tomkins, M. Zhang, and W. D. Heavlin, "Reviewer bias in single-versus double-blind peer review," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 48, pp. 12 708–12 713, 2017.
- [9] J. Y. Kim, G. M. Fitzsimons, and A. C. Kay, "Lean in messages increase attributions of women's responsibility for gender inequality," *J. Pers. Soc. Psychol.*, vol. 115, no. 6, 2018, Art. no. 974.
- [10] D. Marti, "Blind code reviews experiment," 2017. [Online]. Available: <https://blog.zgp.org/blind-code-reviews-experiment/>
- [11] S. Schalk, "Metaphorically speaking: Ableist metaphors in feminist writing," *Disability Stud. Quart.*, vol. 33, no. 4, 2013.
- [12] E. A. Largent and R. T. Snodgrass, "Blind peer review by academic journals," in *Blinding as a Solution to Bias: Strengthening Biomedical Science, Forensic Science, and Law*, Cambridge, MA, USA: Academic Press, 2016, pp. 75–95.
- [13] E. Humphries, "Web extension for debiasing code reviews in splinter experiment," 2017. [Online]. Available: https://bugzilla.mozilla.org/show_bug.cgi?id=1366429
- [14] A. Nelson, "Google and the structural sexism of the american workplace," 2018. [Online]. Available: <https://www.forbes.com/sites/amynelson/2018/10/30/google-and-the-structural-sexism-of-the-american-workplace>
- [15] D. Seetharaman, "Facebook's female engineers claim gender bias," 2017. [Online]. Available: <https://www.wsj.com/articles/facebook-female-engineers-claim-gender-bias-1493737116>
- [16] J. C. Wong, "Facebook: Leaking info about gender bias damages our 'recruiting brand'," 2017. [Online]. Available: <https://www.theguardian.com/technology/2017/may/02/facebook-gender-bias-female-engineers-code>
- [17] S. J. Ceci and D. Peters, "How blind is blind review?," *Amer. Psychol.*, vol. 39, no. 12, 1984, Art. no. 1491.
- [18] C. L. Goues, Y. Brun, S. Apel, E. Berger, S. Khurshid, and Y. Smaragdakis, "Effectiveness of anonymization in double-blind review," *Commun. ACM*, vol. 61, no. 6, pp. 30–33, 2018.
- [19] C. Sadowski, E. Söderberg, L. Church, M. Sipko, and A. Bacchelli, "Modern code review: A case study at Google," in *Proc. 40th Int. Conf. Softw. Eng.: Softw. Eng. Pract.*, 2018, pp. 181–190.
- [20] M. Alam *et al.*, "Blinded vs. unblinded peer review of manuscripts submitted to a dermatology journal: A randomized multi-rater study," *Brit. J. Dermatol.*, vol. 165, no. 3, pp. 563–567, 2011.
- [21] M. Fisher, S. B. Friedman, and B. Strauss, "The effects of blinding on acceptance of research papers by peer review," *Jama*, vol. 272, no. 2, pp. 143–146, 1994.
- [22] A. C. Justice *et al.*, "Does masking author identity improve peer review quality?: A randomized controlled trial," *Jama*, vol. 280, no. 3, pp. 240–242, 1998.
- [23] S. van Rooyen, F. Godlee, S. Evans, R. Smith, and N. Black, "Effect of blinding and unmasking on the quality of peer review: A randomized trial," *Jama*, vol. 280, no. 3, pp. 234–237, 1998.
- [24] R. A. McNutt, A. T. Evans, R. H. Fletcher, and S. W. Fletcher, "The effects of blinding on the quality of peer review: A randomized trial," *Jama*, vol. 263, no. 10, pp. 1371–1376, 1990.
- [25] K. Okike, K. T. Hug, M. S. Kocher, and S. S. Leopold, "Single-blind vs double-blind peer review in the setting of author prestige," *Jama*, vol. 316, no. 12, pp. 1315–1316, 2016.
- [26] R. M. Blank, "The effects of double-blind versus single-blind reviewing: Experimental evidence from the american economic review," *The Amer. Econ. Rev.*, vol. 81, pp. 1041–1067, 1991.
- [27] D. German, G. Robles, G. Poo-Caamaño, X. Yang, H. Iida, and K. Inoue, "Was my contribution fairly reviewed? A framework to study the perception of fairness in modern code reviews," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng.*, 2018, pp. 523–534.
- [28] M. Ware, "Peer review in scholarly journals: Perspective of the scholarly community—results from an international study," *Inf. Serv. Use*, vol. 28, no. 2, pp. 109–112, 2008.
- [29] P. C. Rigby and C. Bird, "Convergent software peer review practices," in *Proc. 9th Joint Meeting Found. Softw. Eng.*, 2013, pp. 202–212.
- [30] J. Ratcliffe, "Moving software quality upstream: The positive impact of lightweight peer code review," in *Proc. Pacific Northwest Softw. Qual. Conf.*, 2009, pp. 171–180.
- [31] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *Proc. 35th Int. Conf. Softw. Eng.*, 2013, pp. 712–721.
- [32] C. Jaspan *et al.*, "Enabling the study of software development behavior with cross-tool logs," *IEEE Softw.*, vol. 37, no. 6, pp. 44–51, Nov./Dec. 2020.
- [33] C. Egelman *et al.*, "Predicting developers' negative feelings about code review," in *Proc. ACM/IEEE 42nd Int. Conf. Softw. Eng.*, 2020, pp. 174–185.
- [34] H. Shah, N. J. Nersessian, M. J. Harrold, and W. Newstetter, "Studying the influence of culture in global software engineering: Thinking in terms of cultural models," in *Proc. 4th Int. Conf. Inter-cultural Collaboration*, 2012, pp. 77–86.
- [35] D. McFadden, *Conditional Logit Analysis of Qualitative Choice Behavior*, P. Zarembka, Ed. Cambridge, MA, USA: Academic Press, 1974.
- [36] J. G. Adair, "The hawthorne effect: A reconsideration of the methodological artifact," *J. Appl. Psychol.*, vol. 69, no. 2, 1984, Art. no. 334.
- [37] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "An empirical study of the impact of modern code review practices on software quality," *Empir. Softw. Eng.*, vol. 21, no. 5, pp. 2146–2189, 2016.
- [38] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida, "Revisiting code ownership and its relationship with software quality in the scope of modern code review," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng.*, 2016, pp. 1039–1050.
- [39] G. Bavota and B. Russo, "Four eyes are better than two: On the impact of code reviews on software quality," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol.*, 2015, pp. 81–90.
- [40] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida, "Investigating code review practices in defective files: An empirical study of the QT system," in *Proc. IEEE/ACM 12th Work. Conf. Mining Softw. Repositories*, 2015, pp. 168–179.
- [41] K. S. McKinley, "Improving publication quality by reducing bias with double-blind reviewing and author response," *ACM SIGPLAN Notices*, vol. 43, no. 8, pp. 5–9, 2008.

- [42] J. D. Wells, D. E. Campbell, J. S. Valacich, and M. Featherman, "The effect of perceived novelty on the adoption of information technology innovations: A risk/reward perspective," *Decis. Sci.*, vol. 41, no. 4, pp. 813–843, 2010.
- [43] A. Bosu, M. Greiler, and C. Bird, "Characteristics of useful code reviews: An empirical study at microsoft," in *Proc. 12th Work. Conf. Mining Softw. Repositories*, 2015, pp. 146–156.
- [44] A. E. Budden, T. Tregenza, L. W. Aarssen, J. Koricheva, R. Leimu, and C. J. Lortie, "Double-blind review favours increased representation of female authors," *Trends Ecol. Evol.*, vol. 23, no. 1, pp. 4–6, 2008.
- [45] O. Åslund and O. N. Skans, "Do anonymous job application procedures level the playing field?," *ILR Rev.*, vol. 65, no. 1, pp. 82–107, 2012.
- [46] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, "The influence of non-technical factors on code review," in *Proc. 20th Work. Conf. Reverse Eng.*, 2013, pp. 122–131.
- [47] O. Kononenko, O. Baysal, L. Guerrouj, Y. Cao, and M. W. Godfrey, "Investigating code review quality: Do people and participation matter?," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol.*, 2015, pp. 111–120.



Emerson Murphy-Hill received the PhD degree in computer science from Portland State University, Portland, Oregon. He is a research scientist at Google in Developer Intelligence, Mountain View, California.



Jillian Dicker received the BSc degree in mathematics and computer science from the University of British Columbia, Canada, and the MSc degree in computer science from Simon Fraser University, Canada. She is currently a software engineer at Google in Developer Intelligence, Mountain View, California.



Margaret Morrow Hodges received the master's degree of public health from the University of California, Berkeley, Berkeley, California. She is currently a user experience researcher at Google in Developer Intelligence, Mountain View, California.



Carolyn D. Egelman received the PhD degree from Carnegie Mellon University, Pittsburgh, Pennsylvania in engineering & public policy. She is a quantitative user experience researcher at Google in Developer Intelligence, Mountain View, California.



Ciera Jaspán received the PhD degree in software engineering from Carnegie Mellon University, Pittsburgh, Pennsylvania. She is currently a software engineer at Google with Developer Intelligence, Mountain View, California.



Lan Cheng received the PhD degree in economics from the University of California, Davis, Davis, California. She is a quantitative user experience researcher at Google in Developer Intelligence, Mountain View, California.



Elizabeth Kammer received the master's degree in computer science from the University of Alabama, Tuscaloosa, Alabama. She is a software engineer at Google, Mountain View, California.



Ben Holtz received the master's degree in computer science from Stanford University, Stanford, California. He is currently a software engineer at Google in Developer Intelligence, Mountain View, California.



Matthew A. Jorde received the master's degree in computer science from the University of Nebraska-Lincoln, Lincoln, Nebraska. He is currently a software engineer at Google in Developer Intelligence, Mountain View, California.



Andrea Knight Dalon received the degree in information systems and human-computer interaction at Carnegie Mellon University, Pittsburgh, Pennsylvania. She is currently a user experience researcher at Google, Mountain View, California.



Collin Green received the BSc degree in psychology from the University of Oregon, Eugene, Oregon, and the PhD degree in psychology from the University of California, Los Angeles, Los Angeles, California. He is a currently user experience researcher and manager at Google in Developer Intelligence, Mountain View, California.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.