Predicting Developers' Negative Feelings about Code Review

Carolyn D. Egelman[1], Emerson Murphy-Hill[1], Elizabeth Kammer[1], Margaret Morrow Hodges[2], Collin Green[1], Ciera Jaspan[1], James Lin[1]
[1]Google, [2]Artech
{cegelman,emersonm,eakammer,hodgesm,colling,ciera,jameslin}@google.com

ABSTRACT

During code review, developers critically examine each others' code to improve its quality, share knowledge, and ensure conformance to coding standards. In the process, developers may have negative interpersonal interactions with their peers, which can lead to frustration and stress; these negative interactions may ultimately result in developers abandoning projects. In this mixed-methods study at one company, we surveyed 1,317 developers to characterize the negative experiences and cross-referenced the results with objective data from code review logs to predict these experiences. Our results suggest that such negative experiences, which we call "pushback", are relatively rare in practice, but have negative repercussions when they occur. Our metrics can predict feelings of pushback with high recall but low precision, making them potentially appropriate for highlighting interactions that may benefit from a self-intervention.

KEYWORDS

code review, interpersonal conflict

ACM Reference Format:
Carolyn D. Egelman[1], Emerson Murphy-Hill[1], Elizabeth Kammer[1], Margaret Morrow Hodges[2], Collin Green[1], Ciera Jaspan[1], James Lin[1]. 2020. Predicting Developers' Negative Feelings about Code Review. In *42nd International Conference on Software Engineering (ICSE '20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3377811.3380414

1 INTRODUCTION

It is well established that modern code review provides many benefits for a software organization, including finding defects [3, 17], knowledge sharing [3, 17, 19], and improving software maintenance [19]. However, code review can also lead to *interpersonal conflict* in the workplace; prior research in the social sciences describe interpersonal conflicts as a "consequential stressor in the workplace" [21]. Anecdotally, professional developers have reported their experiences with stressful [2], toxic [16], and insufferable [7] reviewers of their code. A code review in 2015 of Linux kernel code illustrates conflict vividly:

> *Christ people. This is just sh*t. The conflict I get is due to stupid new gcc header file crap. But what makes me upset is that the crap is for completely bogus reasons... anybody who thinks that the above [code snippet] is (a) legible (b) efficient (even with the magical compiler support) (c) particularly safe is just incompetent and out to lunch. The above code is sh*t, and it generates shit code. It looks bad, and there's no reason for it.*

Just like in the physical workplace, such interactions can have significant consequences. For example, the Linux kernel lost at least one developer due to its apparent toxic culture [22]:

> *I'm not a Linux kernel developer any more... Given the choice, I would never send another patch, bug report, or suggestion to a Linux kernel mailing list again... I would prefer the communication style within the Linux kernel community to be more respectful. I would prefer that maintainers find healthier ways to communicate when they are frustrated. I would prefer that the Linux kernel have more maintainers so that they wouldn't have to be terse or blunt.*

Unfortunately, we have little systematic understanding of what makes a code review go bad. This is important for three reasons. First, from an ethical perspective, we should seek to make the software engineering process fair and inclusive. Second, from a competitiveness perspective, organizations must retain software engineering talent. Third, happier developers report increased feelings of well-being and productivity [9].

This paper seeks to understand and measure negative experiences in code review. This paper contributes the first study, to the authors' knowledge, that quantifies feelings of pushback in the code review process. Measurement enables understanding of the prevalence of the bad experiences, whether negative experiences are occurring at different rates in subpopulations of developers, and whether initiatives aimed at reducing negative experiences, like codes of conduct, are working. In a first step towards enabling new initiatives, we use interviews, surveys, and log data from a single company, Google, to study pushback, which we define as "the perception of unnecessary interpersonal conflict in code review while a reviewer is blocking a change request." We articulate five main feelings associated with pushback, and we create and validate logs-based metrics that predict those five feelings of pushback. This paper focuses on practical measurement of feelings of pushback rather than developing theories around the root causes of pushback happening or why developers feel pushback in different ways.

We asked the following research questions:

RQ1: How frequent are negative experiences with code review?
RQ2: What factors are associated with pushback occuring?
RQ3: What metrics detect author-perceived pushback?

ICSE '20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7121-6/20/05.
https://doi.org/10.1145/3377811.3380414

2 METHOD

To study negative behaviors in code reviews, we combine qualitative and quantitative methods using surveys and log data from Google, a large international software development company. We developed three log-based metrics, informed by interviews with a diverse group of 14 developers, to detect feelings of pushback in code reviews and validated those metrics through a survey with a stratified sample of 2,500 developers that covered five feelings of pushback; 1,317 developers completed the survey. In the survey we collected qualitative feedback and asked respondents to volunteer code reviews that matched a list of problematic behaviors.

2.1 Code Review at Google

At Google, code review is mandatory. The artifact being evaluated in a code review we will call a change request, or CR for short. When a change request is ready, its author seeks acceptance through code review, meaning that the reviewer certifies that they have reviewed the change and it looks okay to check in. A given code review may be performed by one or more developers. Authors choose their reviewers, or use a tool-recommended reviewer. Once each reviewer's concerns (if any) have been addressed, the reviewer accepts the change and the author merges the code into the codebase. Unlike the open source community, almost all code that is reviewed is eventually merged, which makes metrics like "acceptance rate," which are common in studies on open source code reviews, inappropriate in the Google context.

Additionally, any code that is checked into the company's code repository must comply with internal style guides. To ensure compliance, code must either be written or reviewed by a developer with **readability** in the programming language used, which is a certification that a Google developer earns by having their code evaluated by already-certified developers. More information about Google's code review process can be found in prior work [19].

2.2 Interviews about Code Review Pushback

The goals of the interview were to refine our initial definition of pushback and to understand how developers deal with pushback. One primary interviewer conducted all 14 semi-structured, 1-hour interviews, with 1-2 notetakers present for most sessions. Given the sensitivity of the interview topics, we conducted interviews with informed consent and according to a strict ethical protocol.[1]

Participants were recruited in two samples by randomly inviting developers from the company's human resources database. Although Google has developers in multiple countries, all interviewees were based in US offices. The first sample was developers who replied to the recruitment screener ($n = 7$), consisting predominantly of white, male, and senior developers. As this uniform sample might not include experiences more common to less tenured developers or those from underrepresented backgrounds, we rewrote our recruitment materials to be more inclusive and adjusted our sampling script. Consequently, the second sample ($n = 7$) included more junior developers, included racial/ethnic diversity, and consisted entirely of women.

Developers' experience with code review is mostly positive. All 14 interviewees viewed the code review process as helpful, and newer developers particularly appreciated its inherent mentorship. Nonetheless, the interviews pointed out the challenges developers face during code review.

Defining pushback. To inform our definition of pushback, we asked developers what pushback meant to them. While their definitions largely aligned with our *a priori* notion that acceptance was being withheld by a reviewer, it included the idea that pushback also contained interpersonal conflict. Interpersonal conflict was also a recurring theme in the instances of pushback that interviewees shared during the interviews. Thus, our definition of pushback is "the perception of unnecessary interpersonal conflict in code review while a reviewer is blocking a change request."

Dealing with Pushback. Interviewees talked about their reactions, working towards resolution, and the consequences of the pushback they received:

- **Initial reaction.** Aggressive comments can provoke strong emotional reactions, followed by a desire to seek social support from trusted colleagues. Participants talked about shock, frustration, second guessing themselves, and a sense of guilt or shame that came from reading aggressive comments. From a behavioral standpoint, participants discussed asking teammates' opinions, asking a manager or team lead for advice, reviewing the code change and related materials, and reviewing code review guidelines or documentation.
- **Investing time to work towards a resolution.** Participants brought up adding comments asking for clarification, talking directly with reviewers (within the code review tool, over chat, video conference, or offline), making changes to code, and adding docs or other context in the CR feedback as the primary ways they worked toward resolution.
- **Appeal to authority or experts.** Some participants elicited support from someone with authority or expertise to mediate or to steer the review back on track. Techniques used here included adding a manager or team lead to the review, having offline meetings as a part of resolution, or reaching out to an internal company email alias designed to help code reviews (either privately or by directly adding the email alias as a participant in the code review).
- **Long-term consequences.** Participants also shared longer term effects that came about because of code reviews with pushback including not sending code changes to the same reviewer or codebase again, adding more context in future code submissions, building offline relationships before submitting code to a new reviewer or codebase, soliciting more feedback from teammates before sending code changes outside of the team, and avoiding working on projects that overlap with the same reviewer/codebase.

2.3 Design of Metrics

From the interviews we synthesized three potential metrics that we could calculate with existing tool logs to identify code reviews that may be likely to contain pushback. To develop the metrics, we

[1] We note specifically that we provided participants the option to participate with or without audio recording, and we paused recording when starting to discuss topics the interviewer or participant identified as potentially sensitive. We also provided materials on organizational resources for dealing with workplace concerns, and described the responsibilities the researchers and notetakers had related to reporting policy violations to reduce any ambiguity about what to expect during or after the session.

focused on the immediate consequences mentioned in the interviews of pushback: investing time to work towards a resolution and appealing to authority or experts.

We calculate each of the metrics for each developer and for each code review, based on tool logs. We collect user logs from a variety of developer tools and are able to associate logs to a particular code change based on task-identifying artifacts, such as common code review identifiers or version control workspace. We include time using the code review and code search tools, editing code, and browsing documentation. We cannot account for non-logged time such as in-person conversations.

Metric 1: Rounds of a review is a measure of working towards a resolution as mentioned by interviewees, as it captures the extent to which there was back-and-forth between the author and reviewer. We calculate this metric as the number of times an author or reviewer sent a batch of comments for the selected CR.

Metric 2: Active reviewing time is the time invested by the reviewer in providing feedback, as mentioned by interviewees. Another benefit to reviewing time as a metric is that it captures time spent for all reviewers, so it would capture time invested by a reviewer who was an escalation point. It also seems prudent to include time spent by reviewers to cover the case of a reviewer spending a lot of time developing and writing the feedback that is viewed as pushback. We calculate this metric as the total reviewer time spent actively viewing, commenting, or working on the selected CR. This may include time outside of the main code review tool used, such as looking up APIs or documentation. Notably, active review time is not wall-clock time of start-to-finish reviews, but based on time specifically spent working on the code review and related actions.

Metric 3: Active shepherding time covers most of the activities that interviewees mentioned as investing time to work towards a resolution. We calculate this metric by measuring time the author spent actively viewing, responding to reviewer comments, or working on the selected CR, between requesting the code review and actually merging the change into the code base. Similar to active reviewing time, this may include time outside of the primary code review tool, including time editing files to meet reviewers' requests but does not account for in-person conversations.

Each "long" cutoff point was at the 90th percentile[2] for that metric across all developers in the company. We used these criteria to flag reviews with potential pushback:

- 48 minutes for active reviewing time (which we'll call "review time" henceforth for brevity),
- 112 minutes for shepherding time ("shepherd time"), and
- 9 batches of comments for number of rounds of review ("rounds").

We acknowledge that there are situations where flagged, long reviews are useful and desirable (e.g., during mentoring), and part of the validation of these metrics is understanding how often the metrics predict negative interaction compared to other interactions.

[2]For reference, at the 10th percentile: 2 seconds active shepherding time, 14 seconds active reviewing time, 2 rounds of review; and at the median: 13 minutes active shepherding time, 4 minutes active reviewing time, 3 rounds of review.

2.4 Survey Design

To validate the metrics, and to understand developers' experience with pushback more broadly across Google, we designed a survey. We picked a pair of code reviews for each developer invited to take the survey. Each participant was asked to rate those code reviews, supply qualitative feedback, and asked to volunteer code reviews that matched a list of problematic behaviors.

2.4.1 Survey Overview. Below we include a summary of the types of questions we asked. We include question wording in the relevant places in the results, with the full survey in the Supplementary Material. The survey had several parts:

- **Overall code review perceptions.** We asked three overarching questions about perceptions of the code review process covering improvements to code quality, satisfaction and frequency of bad experiences with the code review process.
- **Rating of two selected change requests.** We asked developers similar questions about two CRs; one in which they were involved as either an author or reviewer, and one in which they were not involved. For the selected CRs, we asked respondents to rate the CR on each of the five feelings of pushback (Section 2.4.2) and asked developers to provide open-ended comments to give context.
- **Asking for problematic code reviews.** Finally, we asked our survey takers to volunteer CRs that matched any behaviors from a list of 29 potentially problematic behaviors, shown in Figure 1 drawn from interviewee experiences and literature on workplace bullying [15]. We also asked for open-ended comments, giving participants the option to paste excerpts of the CR they believed were problematic into a text box and to share context on why they believed the behavior occurred.

2.4.2 Feelings of Pushback. To validate our metrics, we began by defining several aspects of the "ground truth" of pushback through feelings expressed in the interviews (Section 2.2). The feelings were:

- **interpersonal conflict**, from the definition of pushback;
- **feeling that acceptance was withheld for too long**, one aspect of 'blocking' behavior from the definition of pushback;
- **reviewer asked for excessive changes**, another aspect of 'blocking' behavior;
- **feeling negatively about future code reviews**, from the long-term consequences of pushback; and
- **frustration**, a direct aspect from the initial reaction to aggressive comments and a consistent theme throughout interviews (e.g., people were frustrated about reviews taking a very long time or not having context on why a reviewer required certain changes).

These aspects allow us to ask developers about specific feelings of pushback, and evaluate to what extent our metrics predict them. These five feelings are not mutually exclusive of each other, and we do not think that they completely cover the range of potential feelings of pushback in the code review process. For example, we didn't use feelings of difficulty, pain, personal attacks, commonness, or fairness, which were all considered in the development of the interview script and survey. However, the feelings chosen were

		CR Criteria for Inclusion		
	Surveyed CRs	Flag: Review Time	Flag: Shepherd Time	Flag: Rounds of Review
0 Flags	250			
1 Flag	200	✓		
	200		✓	
	200			✓
2 Flags	100	✓	✓	
	100	✓		✓
	100		✓	✓
3 Flags	100	✓	✓	✓

Table 1: Dist. of surveyed CRs by metric flag combinations.

prominent themes that came up in our interviews that most closely aligned with our definition of pushback.

One challenge in developing the questions about these feelings for the survey was the appropriate scale to use. We chose to develop item-specific response scales rather than use Likert scale to reduce cognitive complexity and elicit higher quality responses [20]. For our analysis we binned the quantitative survey responses for two reasons: first, for some questions we only cared about one polarity (e.g., acceptance was given "too early" is not pushback by definition and neither are cases where less change was requested than necessary) and second, binning allowed us to treat different response scales uniformly.

2.4.3 Selecting the Code Reviews for the Survey. To evaluate our metrics, we compare the CRs flagged by one or more of our metrics to developers' feelings of pushback. Individual CRs can fall into any one of eight "categories" with respect to the metrics or metric combinations, as detailed in Table 1. We developed a stratified sampling plan to ensure that we selected CRs from all eight categories.

An insight from our interviews was the importance of bringing in a third party to help resolve or escalate any conflict. Because of this, we shaped the survey design to ask developers not only about a change request that they were involved in, but also to evaluate a change request they were not involved in to better understand how apparent feelings of pushback would be in cases where a change request was escalated. We asked the author, one reviewer, and two third-party developers to rate the feelings of pushback they perceived in the code review. See the Supplementary Material for details about our survey stratification process.

2.5 Analysis

Quantitative data. In addition to the survey ratings on the five feelings of pushback, we incorporate data about CRs, authors, and reviewers which may impact the metrics we are measuring including: CR size, if the CR needed readability or was part of the readability certification process, if the author was a new employee, author seniority, the number of reviewers, and the data from our three flags (high review time, high shepherd time, and long rounds). For analysis purposes, we treat the ratings of the pushback feelings as binary and evaluate incidence rates of these undesirable behaviors. We use a logit regression model to predict the likelihood of developers reporting each feeling of pushback based on our metrics controlling for CR attributes. The logit model is ideally suited to predict binary outcomes. We report the transformed odds-ratio by exponentiating the log-odds regression coefficient for all results in this report to aid in interpretation.

Qualitative data. To analyze the qualitative data from open ended survey responses, we used inductive coding to surface themes. One researcher coded the responses from the ratings of CRs selected for the survey and another researcher coded the responses from the volunteered CRs, each developing their own codebook.

Member Checking. We emailed a copy of our internal research report to 1,261 employees and the report was accessible to the entire company of Google; 643 employees viewed the report. Four developers reached out proactively adding their own anecdotal support for the findings. No one refuted the findings. Months after the findings we also distilled recommendations into a one-page article widely circulated at Google; and subsequently published the article on the external Google Testing Blog[3]. The blog article was one of the most viewed Google Testing Blog posts in 2018 or 2019 and generated lively discussion on both Reddit[4] and HackerNews[5].

3 RESULTS

We next describe the results of our survey, where all questions were optional. Of 2,500 survey invitees: 1,317 completed the first section; 606 authors and 573 reviewers completed the second section, about their own CRs; and 1,182 completed the section on third-party CR evaluation. We found no statistically significant response bias from any of the participant or CR attributes considered as part of this analysis. 78% of survey respondents worked in a US office; 16% in Europe, the Middle East, or Africa; 4% in the Asia Pacific region; and 2% in the Americas, but not the US.

3.1 How frequent are negative experiences with code review?

Overall, developers are quite happy with the code review process at Google, with 87% (±2% Clopper-Pearson confidence interval) of developers reporting being at least moderately satisfied, and 96% (±1%) saying code review improves the quality of their code at least a moderate amount. Still, 57% (±3%) report having negative experiences with the code review process at least once a quarter during their employment at Google, and 26% (±2%) have had negative experiences at least once a month. Note that the phrasing of the question on the survey used "bad experience" without any specific definition; consequently, some of these experiences likely fit within our definition of pushback, but most are probably not directly interpersonal conflict. As evidence, looking ahead at the types and frequency of developer-reported negative experiences in Figure 1, "excessive review delays" was the most common.

[3] https://testing.googleblog.com/2019/11/code-health-respectful-reviews-useful.html
[4] https://www.reddit.com/r/programming/comments/dsxpxp/tips_to_resolve_code_review_comments_respectfully/
[5] https://news.ycombinator.com/item?id=21474271

On the survey, we asked about each of the five different feelings of pushback for each change request we assigned to respondents. For analysis purposes, we treat the variables as binary (feeling pushback in that way, or not) and look at incidence rates of these undesirable behaviors. For example, we asked authors "Setting aside inherent difficulties in coding, how frustrating was it for you to get [the selected change request] through review?" and gave the five options of "none", "a little", "a moderate amount", "a lot", and "don't know"; we categorize any response options of "a moderate amount" and "a lot" as frustrating and the other response options as not frustrating. Exact survey question wording, response options and their categorization are in Supplementary Material.

As part of this first research question, we look at incidence rates of each of the five feelings of pushback. Given that we intentionally and disproportionately sampled change requests where we expected a higher likelihood of these feelings as part of our stratified sample, we weight our results by the incidence of each flagged metric combination.

Specifically, in the surveyed change requests ($n = 606$):

- 3% (±2%) of authors reported frustration with a review,
- 2% (±1%) of authors reported interpersonal conflict,
- 5% (±2%) reported reviewers withholding acceptance longer than necessary,
- 2% (±1%) reported that reviewers requested more change than was necessary, and
- 4% (±2%) reported they did not feel positively about submitting similar changes in the future.

Aggregating across the feelings of pushback, we find 11% (±3%) of unflagged CRs include at least one of these feelings of pushback, while 3% (±2%) include two or more feelings of pushback behavior.

3.2 What factors are associated with pushback occuring?

To investigate in what situations pushback is more likely to occur, we leverage both quantitative and qualitative data. First we look at incidence rates for each of our feelings of pushback for key CR attributes such as readability, CR size, new employee status, author seniority, and number of reviewers. Second, we look at how developers answered the question "Why do you think the [frustration, negative behavior] happened?"

3.2.1 Code Review Attributes. There are several important CR attributes we investigated which may impact the metrics we are measuring: readability (both needing readability approval and if the review was a part of the readability certification process), CR size, if the author was a new employee, the seniority of the author, and the number of reviewers. We ran a logit regression with all of these attributes on each of the pushback feelings to test which were predictive of the feelings of pushback. In the following paragraphs we describe which attributes are predictive. Full regression results are available in the Supplementary Material.

Readability. Our internal research at Google on readability suggests that reviews tend to take longer while a developer goes through the readability process, compared to before and after getting readability. That research also suggests that achieving readability is especially frustrating, based on surveys of Google developers. Surprisingly, neither code reviews that were undergoing the readability process nor code reviews that required a readability review were predictive of any of our feelings of pushback. Most open-ended comments related to readability in this dataset lamented the delays that time-zone difference with readability reviewers introduced, as the following participant noted along with a suggested remedy: *"The main frustration relating to this code review was the round-trip required to get the review through readability review. Ideally, there would be more reviewers in the same time zone."*

Code Review Size. Code review size was the one attribute that was predictive of most of the feelings of pushback, but different sized CRs were predictive of different pushback feelings. Google categorizes CRs into specific sizes[6], these sizes are indicated as part of the code review tool and in the notification to the reviewer of the code change, so we used these pre-set categorizations as the levels for CR size rather than a raw number of lines of code changed. Compared to very small code changes (less than 10 lines), changes that were between 250-999 lines long were 4.8 times more likely for authors to feel more change was requested than necessary and changes that were between 10-49 lines were 4.7 times more likely to have authors report they felt negatively about submitting a similar change in the future. For the aggregated pushback feelings of "any pushback" we see any change between 10-999 lines of code is between 2.2 and 2.6 times more likely have some feeling of pushback in the code review. This lends support to the general advice [5] to split change requests for easier and quicker reviews when possible, but it also points to the importance of using multiple metrics to predict pushback.

New Employee Status & Author Seniority. Being a new employee is not a statistically significant predictor of any of our feelings of pushback. Compared to authors at level 1 (entry level), authors at level 3 are 28% less likely to see conflict in their code review changes. Likewise, employees at level 2 are 31% less likely to feel negatively about submitting similar changes in the future.

Number of Reviewers. The number of reviewers is not predictive of any of our pushback feelings.

3.2.2 Selected code reviews: emergent themes on why pushback occurs. In addition to quantitative ratings, we asked respondents why they thought "that behavior occurred during this" change request and a general open ended question about "anything else important that we should know about this" change request. Respondents described diverse issues in flagged change requests, consistent with our interviews: time zones/delays (8% of comments on flagged CRs), mentorship and tenure (6%), conflict (4%), and team- or seniority-related politics (2%). With the exception of one mention of delays, none of these issues were described in any of the unflagged change requests, which bodes well for our metrics, considering they directly measure only engineering time and roundtrips. Although the smaller sample size may conceal some lower incidence issues (unflagged $n = 71$, flagged $n = 320$), the breadth of issues represented among the flagged reviews suggests our metrics successfully capture several important aspects of code review issues.

[6]The categories of CR sizes are: XS (0-9 lines changed), S (10-49 lines changed), M (50-249 lines changed), L (250-999 lines changed), and XL (over 1,000 lines changed). For the 3-month period for CRs eligible for this study, the distribution of all CRs was: 36% XS, 26% S, 25% M, 10% L, 3% XL; and for CRs selected for this study: 17% XS, 22% S, 35% M, 21% L, 4% XL.

Topic	Total Comments	Unflagged Review Comments	Flagged Review Comments
Time zone & delays	26	1	25
Offline Conversation	19	1	18
Readability	15	0	15
Code review was complex or large	14	0	14
Conflict	12	0	12
Code quality & formatting	12	1	11
Documentation & comment clarity	7	1	6
Seniority, tenure, politics	5	0	5

Table 2: Frequencies of codes within open-ended comments for unflagged and flagged change requests.

This meta comment by a participant describes why open-ended comments are a valuable addition to the quantitative ratings; they expose the perspectives of people who have context to interpret more than a neutral observer might pick up:

> *"Code review is the clearest place where interpersonal conflicts play out in the harsh light of day at Google. As a people manager I have had team members point me at code review comment threads as evidence of undercurrents of interpersonal conflict and indeed they are very telling."*

When presented with CRs they had authored or reviewed, respondents pointed out issues including interpersonal, tooling, organizational, and historical attributes of the review interaction or the code itself. Table 2 shows the frequencies of themes that emerged from coding of participant comments about their own CRs. Next, we describe how authors and reviewers view each party's responsibilities, intentions, and actions during contentious reviews, to elucidate some of the factors respondents considered when indicating whether or not a flagged CR contains feelings of pushback.

Authors infer negative and positive intents of their reviewers. Authors made varied assumptions about their reviewers' intentions, varying with how helpful or transgressive they viewed the reviewer's actions. On one hand, many flagged CRs were specifically called out as positive, constructive interactions by the authors:

> *"I was appreciated by [the] reviewer. He suggested a lot of better solution, approach, how I should follow the coding style. I enjoyed to learn from him [sic]."*

On the other hand, when reviews were frustrating, authors speculated about situational factors that might have affected the review rather than assuming negative intentions, particularly when the frustration was due to silence rather than perceived conflict.

Authors didn't interpret positive intent in situations where they believed factors beyond resource needs and the quality of the current CR affected reviewers' behavior, such as career advancement, previous conflict, or personality clashes:

> *"[The reviewer] probably wants to put [this particular migration] in his [promotion] packet...I think [this reviewer] is actually doing the right thing by trying to break this API, but I think [they] could have been a bit less obtuse about it…"*
>
> *"The relevant [developer] disagrees with decisions we made many months ago and is attempting to relitigate the same issue."*
>
> *"I tried to find the best intention - which is that the reviewer has a strong preference on how the code should look like, especially this is the code base that the reviewer usually worked on.* ***However, we did have some strong arguments for another project a few weeks before. I do hope that it's not a factor but I don't know.****"* (Emphasis ours.)

The final quote above shows a tension shared by other developers: not knowing the degree to which the other party brings past conflicts into the current code review can increase tensions moving forward. Displaying author names in the code review tool may facilitate faster code review where reviewers and authors share high trust, but other design choices such as anonymous author code review might alleviate tensions that spiral due to an accumulation of frustrating interactions.

We also note that, as we found in the interviews, power imbalances between code owners and authors were perceived as an important feature of difficult interactions to authors. This participant explained their perspective on this issue bluntly:

> *"Code owners use their privilege to force making unnecessary change."*

Authors' self-effacing comments about their limitations. Right or wrong, some authors blamed themselves for reviewers' irritations:

> *"My frustration was how weak was my knowledge of C++ and internal coding style."*
>
> *"They did not think my skills were sufficient (and to some extent they are certainly correct)."*

Comments like these may reflect authors' polite reluctance to place blame on reviewers for frustrating interactions; this may be a good strategy to maintain relationships with reviewers after conflicts.

3.2.3 Volunteered Code Reviews: Problematic Behaviors and Emergent Themes on why pushback occurs. Since our initial interviews suggest that pushback at Google is rare, we were concerned that of our surveyed flagged CRs may not include a sufficient number of developer-identified instances of pushback. To ensure that we would be able to collect a set of CRs that did exhibit pushback, we showed developers a list of 29 potentially problematic behaviors and gave them the option to volunteer a change request that matched any behaviors from the list. Again, they were asked why "that behavior occurred during this" change request and a general open ended question about "anything else important that we should know about this" change request. In contrast to the flagged CRs

that respondents were surveyed about, which contained a mix of CRs that respondents believed did and did not exhibit feelings of pushback, all of the CRs volunteered by respondents explicitly involved feelings of pushback. Analysis of the open comments about volunteered CRs consequently focused on identifying causes, consequences, and associated factors of feelings of pushback, rather than analyzing whether or not feelings of pushback were present, as was done with the flagged CRs.

The most common behaviors from the list which participants chose to volunteer examples for were excessive review delays (68 reviews), excessive nitpicking (57 reviews), long wait for review to start (53 reviews), and confrontational comments (45 reviews). Respondents could indicate if a CR had more than one of these behaviors. The full summary of the indicated behaviors is in Figure 1.

The relative frequency of the behaviors within this sample of 210 volunteered reviews cannot be considered representative of how often these behaviors occur overall within code review at Google, because the sample may have been biased by what respondents felt comfortable volunteering (i.e., respondents may have felt safer sharing a CR with a delay versus a more sensitive CR with an example of humiliation). However, the high incidence of certain behaviors within this sample does validate their existence as anti-patterns worth addressing, for example, excessive nitpicking and confrontational comments.

Additionally, for each of the 29 behaviors, we analyzed code review excerpts and their accompanying open-ended comments to pull out common themes and formulate a working definition for each category. The goal of this analysis was to better understand what was distinctive about each behavior as well as the context surrounding each type of occurrence.

Of the emergent themes that surfaced during qualitative analysis of the 210 volunteered CRs, some of the most commonly mentioned associated factors were delays (65), references to code style (46) and documentation (39), the tone in which feedback was delivered (43), familiarity with the codebase (35), and the CR being part of a readability review (34). The most common emergent themes which we classified as consequences were inefficient use of time (31), negative emotions (29), and offline conversations (28). These consequences align with those described by participants in the initial interviews, including reactions of shock, frustration and shame when experiencing feelings of pushback, and the need to invest additional time to work toward resolution.

Of the common associated factors, four that could be considered causes of pushback in code review were comments on code style, the tone in which feedback was delivered, low familiarity with the code base, and lack of rationale for suggestions or requests made.

Many respondents felt pressured to implement what they considered to be subjective preferences related to how code is written and formatted. One respondent summarized the issue with the example CR they volunteered in this way: *"the personal opinion of the reviewer was strong and they felt the need to point it out, even if it is not in guidelines, and it is a practice that is not uncommon in standard libraries."* Another respondent explained, *"there are a lot of reviewers that are unreasonable and take their criticism far beyond what's documented in the style guide."*

Figure 1: Frequency of behaviors indicated by developers in CRs they volunteered in the final section of the survey.

The tone of feedback also formed a key aspect of respondents' experiences in these scenarios, as one respondent stated, *"presumably the reviewer was trying to educate me on issues seen in the code. What I found objectionable was the way in which it was phrased, the scope creep, and the dismissive tone."* Another respondent described confrontational language in one review as akin to *"treating code review like a discussion on a web forum, i.e., ignoring the other person as an individual and arguing overly aggressively."* This theme aligns with findings from the interviews, during which participants framed aggressive comments as an initial factor often contributing to feelings of pushback

Lack of context on the codebase was also identified as a cause of friction within reviews: *"it was annoying that he was not familiar with the codebase and was questioning stuff about business logic and naming."* Another respondent described a situation in which the

Feeling of Pushback	0 Flags	3 Flags
frustration	1%	27%
interpersonal conflict	0%	22%
acceptance was withheld for too long	4%	27%
reviewer asked for excessive changes	1%	16%
feeling negatively about future CRs	4%	8%
any pushback behavior	7%	49%
2+ types of pushback behavior	1%	27%

Table 3: Incidence rates for each feeling of pushback as rated by authors of surveyed CRs; comparison of CRs not flagged by any metrics and flagged on all 3 metrics.

Pushback Feeling	Precision	Recall
frustration	0.11	0.98
interpersonal conflict	0.10	1.00
acceptance was withheld for too long	0.10	0.93
reviewer asked for excessive changes	0.07	0.97
feeling negatively about future CRs	0.06	0.87

Table 4: Precision and recall by feeling of pushback.

reviewer was *"not willing to stop asking for something when working in a completely different code base even after we told him it was not normal for our code base,"* concluding that they *"[did not let up] until a more senior (and male) engineer stepped in."* This factor echoes interview participants' descriptions of providing additional context or documentation during code review as a strategy to both prevent or mitigate feelings of pushback, particularly when submitting code outside of their team.

Several respondents expressed frustration when reviewers requested non-trivial changes without providing a justification. For example, one respondent described how, *"some of the comments seemed in general hard to understand, and required a chain of replies from the reviewer/author until the reason why the change is asked is explained,"* concluding, *"this prevents the person asking for review from learning."*

3.3 What metrics detect author-perceived pushback?

3.3.1 Occurrence of undesirable behaviors. As discussed in Section 3.1, our five feelings of undesirable pushback behavior are rare occurrences in unflagged CRs with 3% (±2%) of reviews including two or more types of pushback feelings and 11% (±3%) including at least one type ($n = 606$).

However, all these feelings are more frequent in the reviews we identified with our candidate pushback measures, as seen in Table 3. Most notable is the case where reviews are long on all three of our metrics. Of CR authors with long shepherding time, long reviewing time, and many rounds of review, 27% report frustration, 22% report interpersonal conflict, 27% report delayed acceptance, 16% report more change requested than necessary, and 8% report they feel negatively about submitting similar changes in the future. 49% of these authors report feeling at least one type of pushback and 27% report 2 or more types of pushback. The incidence rates of these feelings of pushback suggest that our metrics are effective at flagging CRs where the author has feelings of pushback.

3.3.2 Precision & Recall of the Metrics. We quantify the performance of our metrics through precision and recall in Table 4. To get the top-level performance of our metrics we consider if a CR is flagged by any one or combination of the three metrics to count as flagging that CR for possible pushback. For more detailed results on the performance of each metric or metric combination we direct the reader to the regression results in Section 3.3.3, which account for control variables. The metrics have a high recall rate (between 93% and 100%) and low precision (between 6% and 11%), meaning that while we are capturing most incidents of pushback with our metrics, we are also flagging many reviews as potentially containing pushback that did not include any feelings of pushback for the author.

A caveat when interpreting Table 4 is that it's not fully representative of the expected precision and recall in the population of CRs, due to our stratified sampling, which undersamples unflagged reviews and oversamples flagged reviews. Consequently, in the full population of CRs, we would expect to see higher precision and lower recall than shown in Table 4; nonetheless, we expect the low-precision and high-recall trend to generalize for our metrics.

3.3.3 Robustness of metrics to confounding factors. While we notice patterns across flagged and unflagged CRs (and within different combinations of our flags), it is important to account for other aspects of code review that influence the time to review or shepherd and the number of comments such as the change request attributes we discuss in Section 3.2 (e.g., readability, change size, if the author was a new employee, the seniority of the author, and the number of reviewers). Our regression results[7], in Figure 2, look at how effective our metrics are at detecting feelings of pushback taking into account those change request-level attributes. We report below on the odds ratio for flag combinations that are statistically significant. Taking these into account, compared to a typical change request, authors of reviews which take a lot of round trips (9 or more) and a long reviewing time (more than 48 minutes) are:

- 21 times more likely to be frustrated,
- 13 times more likely to say the reviewer requested more changes than necessary, and
- 7 times more likely to report reviewers withholding acceptance longer than necessary.

When a similar CR additionally takes a long shepherding time, those likelihoods of negative feelings increase dramatically:

- 45 times more likely to be frustrated,
- 20 times more likely to say the reviewer requested more changes than necessary, and
- 14 times more likely to report reviewers withholding acceptance longer than necessary.

[7]Full regression results are also in the Supplementary Material.

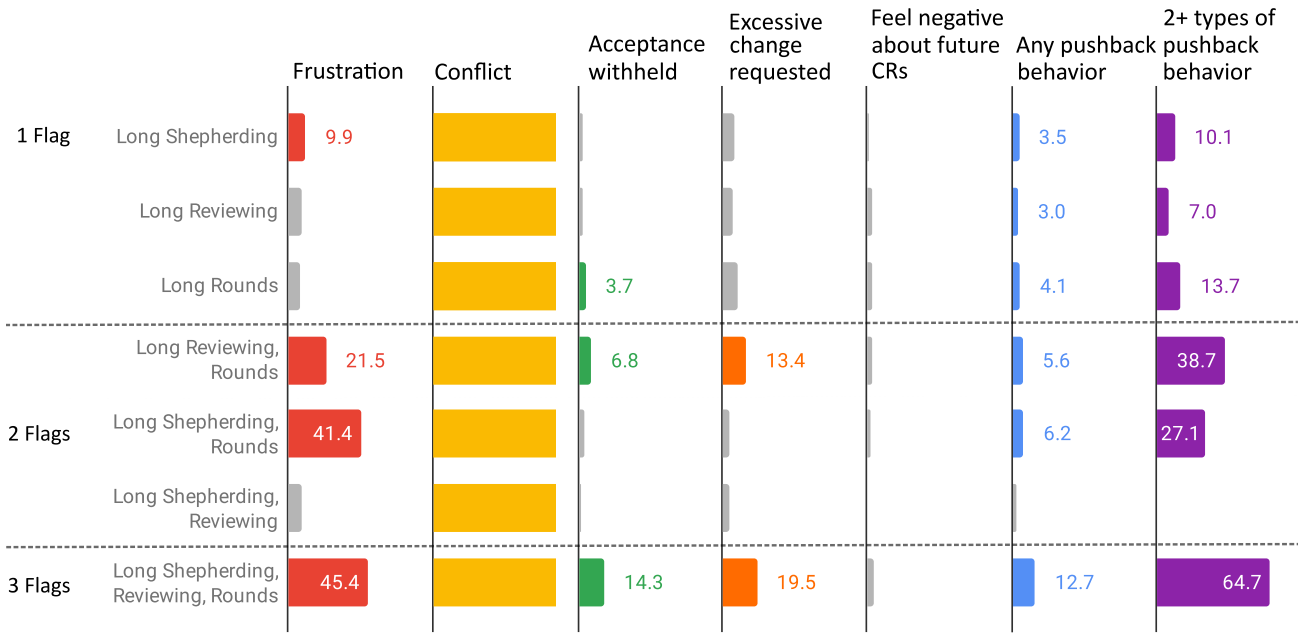

Figure 2: Regression results predicting the likelihood of CR having undesirable, pushback behaviors. Numbers shown are the odds ratio from the fitted logistic regression model where the baseline category is an unflagged CR. Gray bars indicate a coefficient was not statistically significant at the $p < 0.05$ level.

When we look at the aggregate measures of feelings of pushback, we see that a flag on any one of our metrics indicates that authors are between 3.0 and 4.1 times more likely to experience at least one feeling of pushback (compared to an unflagged review), and between 7.0 and 13.7 times more likely to experience multiple feelings of pushback. If a change request is flagged on having a large number of rounds of review plus either long reviewing or shepherding time, the likelihood of at least one feeling of pushback is between 5.6 and 6.2 times that of an unflagged review, and for multiple feelings of pushback is between 27.1 and 37.7 times that of an unflagged review. Interestingly, if a change request is flagged for both long reviewing time and long shepherding time, but not a large number of rounds of review, then that review isn't statistically more likely to have feelings of pushback than an unflagged review.

To help illustrate what this means, consider these interactions:

- **Situation 1:** An author requests review for a change, the reviewer spends a long time looking at the code and providing comments over 9+ different rounds of review, the author is able to quickly address all comments as they come in, but with each round there is more, detailed feedback to address.
- **Situation 2:** An author requests review for a change, the reviewer gives short comments for each round, the author spends a lot of time addressing the comments, this goes on for 9+ rounds of back-and-forth.
- **Situation 3:** An author requests review for a change; the reviewer takes a long time giving rich, detailed feedback; the author then spends substantial time addressing all reviewer comments. All of this is done in just a few rounds of review.

Situations 1 and 2 come across with our analysis as very frustrating to authors, especially situation 2. Situation 1 also generates feelings that acceptance is withheld and that excessive changes are being requested. In both cases, authors are more likely to experience at least two feelings of pushback. However, situation 3 doesn't create any of these negative feelings of pushback. It seems developers don't mind spending an extended time shepherding a change through the process when they receive feedback (even if it came from a very time-consuming review) as long as it is in a succinct number of rounds of feedback. This may be due to reviewers being especially clear about what they want in their initial feedback.

A Note on Conflict & Feelings about Future Changes. In our analysis on authors, two of our feelings of pushback did not yield statistically significant results: conflict and negative feelings about submitting future changes. We next describe why.

Negative Feelings About Future Changes. Some developers indicated that particular CRs generated negative feelings about submitting future changes, but none of our metrics or any controls were predictive of answers to this question. Additionally, we only asked this question of authors on the survey, so we cannot test the strength of this measure from other perspectives. We conclude here that our metrics don't detect this feeling of pushback.

Conflict. The results on conflict tell a different story. For the change requests where authors replied to the survey, none of the 0-flag reviews included conflict, whereas authors did indicate conflict on some of the reviews that were flagged by our metrics. While this is generally positive evidence for the utility of our metrics, it does create a problem for statistical analysis: a naïve interpretation

here is that our metrics are perfect at detecting potential conflict. That would be surprising given that conflict can manifest in ways other than delays or in comment threads. Still, conflict appears to be rare even for flagged reviews, and we see the metrics' flagging of reviews with conflict as an encouraging sign that they effectively select problematic reviews.

3.3.4 Third-party evaluators detect pushback similar to authors. How third parties' perceive feelings of pushback is important because escalation or looping in a new person is one avenue to address or mitigate pushback. We replicated the regression analysis[8] for the third-party rated CRs, and found that developers who were not involved in a particular CR detect feelings of pushback in similar patterns to authors of those CRs. Specifically, when a CR has a large number of round trips and a long active review time, third party developers are 4.8 times more likely to report frustration for the author and 14.3 times more likely to report that there was excessive change requested by the reviewer ($n = 1,182$).

3.3.5 Metrics identify other problematic code reviews. Echoing the low precision and high recall from our five explicit feelings of pushback, we find 1) most code reviews are uneventful and our current metrics filter out many of those; unflagged CRs (75%) were over three times more likely to be described as "boring" by respondents compared to the CRs flagged by our metrics (23%) and 2) that of the 210 CRs that developers volunteered as exhibiting problematic behaviors in the last section of the survey, our metrics would have flagged 93% of them. The later analysis demonstrates that our metrics do well at detecting pushback beyond the five feelings we explicitly covered in the survey.

4 LIMITATIONS

Readers should consider several limitations when interpreting our results which we detail here.

Our results have limited generalizability to other companies. Replication at other companies is necessary to determine how well this result generalizes in a corporate context. In an open source context, we hypothesize that pushback is more common, based on previous work. Some open source communities use "frequent rejections and…harsh language deliberately as a 'congestion control' mechanism"[1] and that, for women in particular, negative interpersonal interactions are "as constant as [they are] extreme"[14].

Likewise, our pushback definition and feelings we used in our survey were derived from interviews with 14 developers. Also, while we purposefully attempted to sample a diverse pool of interviewees, they nonetheless all were employed by one company and do not represent the full range of diverse experiences at that company. Additionally, the taxonomy of feelings we used was not designed to be comprehensive of all possible feelings of pushback.

While we asked survey respondents to determine whether certain code reviews contained feelings of pushback, they may have been reluctant to do so over concerns about outing their peers, despite our assurance that individual results are confidential. Consequently, the frequency of pushback described in our study may be an undercount.

[8]Full regression results are also in the Supplementary Material.

We developed our own item-specific scales for the five feelings of pushback. We performed limited validation on these question scales through draft question review with four researchers and pilot testing with two developers, and so these scales may not measure developer feelings with full accuracy. It is possible there may be knock-on effects from a non-accurate question scale and that our logs-based metrics do not predict those feelings as well as reported in this paper. Future work should look at validating any new scales used, or draw on work on existing scales, such as Harrington's frustration discomfort scale [10].

Similarly, the pushback metrics for this paper are an initial attempt at predicting developer-identified feelings of pushback. Some of these metrics – active review time and shepherding time in particular – are currently unavailable in many software development contexts, but we anticipate that these will become increasingly available as development in the cloud becomes more popular (e.g., GitHub with the Atom editor). Other available code review data, such as textual comment data analyzed using machine learning, can almost certainly improve pushback prediction accuracy. Moreover, our 90% threshold in each metric for "high" pushback is somewhat arbitrary – tuning this threshold with a larger dataset would likely improve our metrics.

5 DISCUSSION

Overall, we find that pushback is relatively rare but does occur in code review at Google.

Our metrics have high recall but low precision when predicting which code reviews included feelings of pushback. Our metrics detected code reviews that were significantly more likely to be frustrating for authors (21 times more likely than unflagged reviews) and invoke feelings that acceptance was withheld (7 times) and/or that excessive changes were requested (15 times). We also found that most code reviews are uneventful; our current metrics filter out many of those. Unflagged reviews (75%) were over three times more likely to be described as boring compared to the reviews flagged by our metrics (23%). Moreover, our metrics would have flagged 93% of code reviews that developers volunteered as exhibiting problematic behaviors.

One particularly generalizable finding from this work is the understanding that third parties generally detect pushback in the same way that authors do. This demonstrates that a previously uninvolved person, such as a manager or mentor, is likely to observe pushback within the review if the author reaches out for help and advice. In our study, developers who were not involved in a particular code review do detect feelings of pushback in similar patterns to authors of those CRs: when a CR was flagged by our metrics, third-party developers are 5 times more likely to report frustration for the author and 14 times more likely to report that excessive changes were requested by the reviewer.

Our metrics can predict feelings of pushback with high recall but low precision. The low precision makes them potentially appropriate for highlighting interactions that may benefit from a self-intervention (e.g., escalating the review to a manager, moving a conversation offline, or seeking guidance from a colleague or mentor). For instance, once a code review is flagged by our metrics, a bot could comment on the code review, suggesting that the review

is statistically exceptional and that the involved parties should consider alternative strategies to make progress. For instance, the bot could suggest that discussions could be taken offline, that another expert be brought in to provide an outside perspective, or that a change should be broken into smaller parts. We suggest a tool like this bot, which is at the weaker end of tools that could be built, due to the high false positive rate of our current metrics. Fundamentally, because of the high recall but low precision, a knowledgeable developer must still be the one to judge whether pushback is truly occurring and which strategy is appropriate. High recall makes the metrics appropriate to apply in aggregate when looking at pushback rates within a group or groups of developers to determine the impact of any interventions targeted at reducing pushback, for example anonymous code review.

There are several ways these metrics should not be used. We do not think our metrics should be used to flag reviews to management or human resources. We do not think that our metrics should be used as a justification to say that a change request has gotten "enough" feedback. And we do not think our metrics can be used as a substitute for human judgement or empathy.

6 RELATED WORK

Code review has been described in the software engineering literature in a variety of companies. Bacchelli and Bird described code review at Microsoft, finding that different teams had different cultures and practices around code review [3]. Like Google, Microsoft does have a centralized code review tool, which shows change diffs that reviewers can comment on. Likewise, Facebook's Phabricator provides developers the ability to comment on proposed changes [26]. Salesforce reviews code using the Collaborator tool [29], with similar features.

Several papers have examined the benefits of code review. Alami and colleagues interviewed 21 open source developers about why code review works for open source communities, finding that rejection is important because it helps maintain code quality and that success in contributing to open source depends on being able to handle rejection [1]. Open source communities have mechanisms to mitigate rejection, including an ethic of passion and caring. Although Google code review does not have an explicit notion of rejection, withholding acceptance is similar; we extend Alami and colleagues' work by examining the effect of withholding acceptance. Bosu and colleagues found that within OSS code review has a positive effect on friendship and participants' perceptions of expertise, reliability, and trustworthiness [6]. Our work extends this research by focusing on the cases where negative outcomes occur.

Other research has examined how decisions about how reviewers are chosen. German and colleagues found that fairness is a concern in open source code reviews [8]. They describe four types of fairness: distributive, procedural, interaction, and information. Pushback is most related to interaction fairness, as it involves authors having negative interactions with reviewers during the code review process. German and colleagues' discussion of their framework was focused on which code contributions are picked to be reviewed. Ruangwan and colleagues find that reviewers tend to decide which review requests to respond to based on their familiarity with the author, their review workload, and how active the reviewer has been in the past [18]. Yang and colleagues used social network analysis to study the most important roles in the code review process [28]. Our work builds on this prior work by focusing on the interaction between author and reviewer during the code review process.

There have been several studies on whether a code change is accepted. Baysal and colleagues [4] and Kononenko and colleagues [13] discuss what non-technical factors affect whether a change request is approved and how buggy the code may be. Terrell and colleagues investigated the effect of gender on the acceptance of code reviews for open source projects and found that, overall, women's code reviews were approved at a higher rate than men's, but that code reviews from outsiders who were identifiable as women were accepted at a lower rate than male outsiders [25]. von Krogh and colleagues found that in one open source community, potential contributors who followed an implicit "joining script" have a better chance of having their code reviews accepted [27]. Jiang and colleagues found, for patches to the Linux kernel, that the more experience the patch's developer had and the more discussion and iteration the patch went through, the more likely the patch was accepted [12]. Our paper builds on this work by focusing on the interaction between an author and the reviewers.

Finally, prior researchers have studied newcomers to open source projects about their initial experiences, finding that newcomers sometimes feel their initial contributions are unwelcome. Jenson and colleagues found that about while most newcomer contributions received positive replies, 1.5% were characterized as "rude or hostile" [11]. Steinmacher and colleagues found that of 13 surveyed developers who had dropped out of open source projects, six reported a poor reception was the cause [24]. Steinmacher and colleagues observe that more contextualized research is needed to understand these negative interpersonal interactions [23]; this paper provides a step in that direction.

7 CONCLUSION

Pushback during code review can have significant negative consequences for individuals and organizations. In this paper, we found that pushback is rare, can be identified by five negative developer feelings, and that our three metrics for detecting author-perceived pushback have high recall but low precision. While our predictions are far from perfect, we believe such predictions are needed to support future interventions designed to help reduce pushback so that we can monitor the effectiveness of those interventions. Detecting pushback may also help identify any subpopulations where pushback may be more prevalent.

8 ACKNOWLEDGEMENTS

Thanks to anonymous participants and reviewers, Adam Bender, Daniel Berlin, Marian Harbach, Max Kanat-Alexander, Ash Kumar, Andrew Macvean, Kristóf Molnár, Ambar Murillo, Rachel Potvin, Niranjan Tulpule, and Jeff Warshaw.

REFERENCES

[1] Adam Alami, Marisa Leavitt Cohn, and Andrzej Wąsowski. 2019. Why does code review work for open source software communities?. In *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press, 1073–1083.

[2] KD Singh Arneja. 2015. Code reviews do not have to be stressful. Available from https://medium.com/idyllic-geeks/code-reviews-do-not-have-to-be-stressful-919e0a8377a1.

[3] Alberto Bacchelli and Christian Bird. 2013. Expectations, outcomes, and challenges of modern code review. In *International Conference on Software Engineering (ICSE)*. IEEE Press, 712–721.

[4] Olga Baysal, Oleksii Kononenko, Reid Holmes, and Michael W Godfrey. 2013. The influence of non-technical factors on code review. In *Working Conference on Reverse Engineering (WCRE)*. 122–131.

[5] Atlassian Blog. 2018. The (written) unwritten guide to pull requests. Available from https://www.atlassian.com/blog/git/written-unwritten-guide-pull-requests.

[6] Amiangshu Bosu and Jeffrey C Carver. 2013. Impact of peer code review on peer impression formation: A survey. In *Empirical Software Engineering and Measurement (ESEM)*. IEEE, 133–142.

[7] Erik Dietrich. 2018. How to Deal with an Insufferable Code Reviewer. Available from https://daedtech.com/insufferable-code-reviewer/.

[8] Daniel German, Gregorio Robles, Germán Poo-Caamaño, Xin Yang, Hajimu Iida, and Katsuro Inoue. 2018. "Was My Contribution Fairly Reviewed?" A Framework to Study the Perception of Fairness in Modern Code Reviews. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE, 523–534.

[9] Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. 2018. What happens when software developers are (un) happy. *Journal of Systems and Software* 140 (2018), 32–47.

[10] Neil Harrington. 2005. The frustration discomfort scale: Development and psychometric properties. *Clinical Psychology & Psychotherapy: An International Journal of Theory & Practice* 12, 5 (2005), 374–387.

[11] Carlos Jensen, Scott King, and Victor Kuechler. 2011. Joining free/open source software communities: An analysis of newbies' first interactions on project mailing lists. In *44th Hawaii International Conference on System Sciences*. IEEE, 1–10.

[12] Yujuan Jiang, Bram Adams, and Daniel M German. 2013. Will my patch make it? and how fast?: Case study on the linux kernel. In *International Working Conference on Mining Software Repositories (MSR)*. IEEE Press, 101–110.

[13] Oleksii Kononenko, Olga Baysal, Latifa Guerrouj, Yaxin Cao, and Michael W Godfrey. 2015. Investigating code review quality: Do people and participation matter?. In *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 111–120.

[14] Dawn Nafus. 2012. 'Patches don't have gender': What is not open in open source software. *New Media & Society* 14, 4 (2012).

[15] Lyn Quine. 1999. Workplace bullying in NHS community trust: staff questionnaire survey. *BMJ* 318, 7178 (1999), 228–232.

[16] Philipp Ranzhin. 2019. I ruin developers' lives with my code reviews and I'm sorry. Available from https://habr.com/en/post/440736/.

[17] Peter C Rigby and Christian Bird. 2013. Convergent software peer review practices. In *International Symposium on Foundations of Software Engineering (FSE)*. 202–212.

[18] Shade Ruangwan, Patanamon Thongtanunam, Akinori Ihara, and Kenichi Matsumoto. 2019. The impact of human factors on the participation decision of reviewers in modern code review. *Empirical Software Engineering* 24, 2 (2019), 973–1016.

[19] Caitlin Sadowski, Emma Söderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. 2018. Modern code review: a case study at google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*. ACM, 181–190.

[20] Willem Saris, Melanie Revilla, Jon A. Krosnick, and Eric M. Shaeffer. 2010. Comparing Questions with Agree/Disagree Response Options to Questions with Item-Specific Response Options. *Survey Research Methods* 4, 1 (May 2010), 61–79. https://doi.org/10.18148/srm/2010.v4i1.2682

[21] Scott Schieman and Sarah Reid. 2008. Job authority and interpersonal conflict in the workplace. *Work and Occupations* 35, 3 (2008), 296–326.

[22] Sage Sharp. 2015. Closing A Door. Available from http://sage.thesharps.us/2015/10/05/closing-a-door/.

[23] Igor Steinmacher, Marco Aurelio Graciotto Silva, Marco Aurelio Gerosa, and David F Redmiles. 2015. A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology* 59 (2015), 67–85.

[24] Igor Steinmacher, Igor Wiese, Ana Paula Chaves, and Marco Aurélio Gerosa. 2013. Why do newcomers abandon open source software projects?. In *6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 25–32.

[25] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. 2017. Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Computer Science* 3 (2017), e111.

[26] Alexia Tsotsis. 2011. Meet Phabricator, The Witty Code Review Tool Built Inside Facebook. https://techcrunch.com/2011/08/07/oh-what-noble-scribe-hath-penned-these-words/.

[27] Georg Von Krogh, Sebastian Spaeth, and Karim R Lakhani. 2003. Community, joining, and specialization in open source software innovation: a case study. *Research policy* 32, 7 (2003), 1217–1241.

[28] Xin Yang, Norihiro Yoshida, Raula Gaikovina Kula, and Hajimu Iida. 2016. Peer review social network (PeRSoN) in open source projects. *Transactions on Information and Systems* 99, 3 (2016), 661–670.

[29] Tianyi Zhang, Myoungkyu Song, Joseph Pinedo, and Miryung Kim. 2015. Interactive code review for systematic changes. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 111–122.

Supplement to "Predicting Developers' Negative Feelings about Code Review" Carolyn D. Egelman, Emerson Murphy-Hill, Elizabeth Kammer, Margaret Morrow Hodges, Collin Green, Ciera Jaspan, James Lin ICSE '20: International Conference on Software Engineering

This information is added as a supplement to aid in reproducibility. Note that we have chosen not to release our dataset and code for calculating metrics. The reason for not releasing our dataset is that – at its core – the dataset is the code reviews themselves, which contain both proprietary information and potentially sensitive discussions that may inadvertently reveal employee identities. The reason for not releasing our code for calculating the pushback metric is that its constituent parts – review time, shepherding, and rounds – are all gathered from tools not available outside our company; consequently, the code would be neither usable nor understandable outside of the context of our company.

A INTERVIEW SCRIPT

A.1 Preparation & Front Matter (5 minutes)

Go through an information sheet and informed consent with participant, start recording

A.2 Discuss code review roles and definitions (5 minutes)

To start the conversation today, I'd like to understand how you think about a couple of aspects of code review.

- In your view, what is the purpose of going through code review?
- In your view, what are the reviewer's responsibilities when they review a [CR]?
- What would be out of scope of the reviewer's responsibilities during code review?
- What are the author's responsibilities when going through code review?
- What is out of scope for the author's responsibilities when going through code review?
- If a [CR] author said "The reviewer pushed back on my [CR] a lot" what would you think the author meant? / How would you define pushback?
- If a [CR] reviewer said "I pushed back on that [CR] a lot" what would you think the reviewer meant?
- We're talking to several people about their experiences, so as we go through the rest of the study, we want to make sure people have the same definitions for terms in mind. So when we're talking about pushback, we're referring to instances where a reviewer is, in your view, unnecessarily blocking approval of the [CR].
- How does that feel as a definition of pushback to you?

A.3 Thinking about [CR]s

Categories:

- Reviewer takes longer than expected to start review
- Reviewer takes longer than expected to finish review
- Reviewer has pushed back on my previous change in a similar way
- Reviewer feedback takes many rounds to resolve
- Reviewer requires changes without justifying them
- Reviewer asks for the change to be split up
- Reviewer phrases feedback as about you rather than the code
- Reviewer asks for a change that will delay a launch
- Reviewer requests changes based on their opinion rather than style guide/policy
- Reviewer asks for changes without suggesting a fix
- Reviewer provides feedback inconsistent with other reviewers
- Reviewer phrases feedback in an aggressive way
- (Participant suggestion)

Steps:

- Have the participant look at these categories on cards. We're going to go through a series of different rating scales for them. For each one, I'll have you take a few minutes to go through the rating, and if you can talk me through how you're deciding where to put them.
  - Are there any categories we should add? If so, write on a card and rate with the rest.
- Rank: Very common, Somewhat common, Neither common nor uncommon, Somewhat uncommon, Very common
  - Look at the ones on the [common] side: what do those kinds of CRs have in common?
  - Same on the [uncommon] side.
- Rank: Extremely frustrating, Very frustrating, Moderately frustrating, Slightly frustrating, Not at all frustrating
  - Look at the ones on the [more frustrating] side: what do those kinds of CRs have in common?

  - Same on the [less frustrating] side.
- Rank: Very fair, Somewhat fair, Neither fair nor unfair, Somewhat unfair, Very unfair
  - Look at the ones on the [fair] side: what do those kinds of CRs have in common?
  - Same on [unfair] side.

A.4 Highest pushback deep dive (5 minutes)

Okay, we've talked a lot about these hypothetical situations with code reviews. Now I'd like to ask you about experiences you've had with code review.

- If I ask about the most unfair pushback that you've received during code review as the author of a change at Google, does a particular experience come to mind?
  - If so: Tell me about that experience.
  - If not: Tell me about the most recent time you felt like a reviewer was unfair during a code review.
- In your opinion, what factors contributed or led to that situation?
- What were your options in responding to the reviewer?
- How did you decide what to do?
- How was the situation ultimately resolved?
- How did that experience affect your expectations about future code reviews, or did it not have an effect?

A.5 General Questions and Wrap-up (10 minutes)

Let's talk about code review in general for a few minutes to wrap up.

- What factors make a change more or less likely to elicit pushback from reviewers?
- Have you seen code review practices differ across different locations or teams?
- If you could wave a magic wand and change something about code reviews at Google, what would you change, and why?
- Is there anything else you'd like to share about your experiences with code review at Google?
- Anything you think we could ask future participants to get at their experiences with pushback in code review?

Debrief and thank participant

B SURVEY

B.1 Full Survey

B.1.1 Welcome Text. You've been selected to tell us about your experiences with the code review process as a Google engineer. Answering the following questions should take about 13 minutes and will have four parts:

(1) This page
(2) We will ask you about a [CR] you were involved in
(3) We will ask you about a [CR] you were not involved in
(4) We will ask you to provide a [CR]

The aggregate survey results will inform proposed improvements to the code review process and be shared with relevant leadership teams within Google. Your participation is voluntary and your responses to this survey are confidential, as per our employee privacy design doc and are compliant with Google's employee privacy policy.

This survey may cover some [CR]s which provoked negative experiences. Congruent with blameless postmortems, we're not interested in placing blame after the fact, but we are seeking to understand the whole code review process and looking for examples to avoid future unnecessary conflict.

In the course of our review, serious issues or potential policy violations may come to our attention. These issues may warrant further investigation by HR, including following up with the submitter. You are encouraged to report concerns via any of the channels listed on <internal website> (including the anonymous helpline open to employees which is run by a helpline provider that is entirely independent of Google) or talk to your HR representative, so it can be addressed appropriately. If you have any questions about this survey, please email <research team email alias>.

B.1.2 Part 1 - Initial Questions.

Item#	Question	Response(s)
Q01	How much has code review at Google improved the quality of your code?	None A little A moderate amount A lot Don't know
Q02	How satisfied are you with the code review process at Google?	Very dissatisfied Somewhat dissatisfied Neither satisfied nor dissatisfied Somewhat satisfied Very satisfied
Q03	Since joining Google, on average how often have you had a bad experience with the code review process?	Once or more a week Once or more a month Once or more a quarter Once or more a year Never

B.1.3 Part 2 - A [CR] for which you were an author or reviewer. This set of questions asks about cr/[XXX]. Please take a moment to review the [CR] and then return to this page to answer the following questions.

Item#	Question	Response(s)
Q04	Given the importance and initial quality of this [CR], the amount of change requested by reviewers was:	Substantially less than was necessary A little less than was necessary The right amount A little more than was necessary Substantially more than was necessary Don't know
Q05	The reviewers [accepted]:	Quite a bit before the [CR] was of sufficient quality A little before the [CR] was of sufficient quality When the [CR] was of sufficient quality A little after the [CR] was of sufficient quality Quite a bit after the [CR] was of sufficient quality Don't know
Q06	Setting aside inherent difficulties in coding, how frustrating was it for you to get this [CR] through review?	None A little A moderate amount A lot Don't know
Q07	How much interpersonal conflict was there between the author and reviewer(s) in this [CR]?	None A little A moderate amount A lot Don't know
Q08	[Authors only] When I received feedback about this [CR], I felt positively about submitting a similar change in the future.	Strongly disagree Somewhat disagree Neither agree or disagree Somewhat agree Strongly agree Don't remember
Text	Based on our prior research at Google, we know that most [CR]s contain respectful interactions. But we also want to understand those rare, negative behaviors. Congruent with blameless postmortems, we're not interested in placing blame after the fact, but we are looking for examples to avoid future unnecessary conflict.	
Q09	Given any frustration and interpersonal conflict you noted above, please copy and paste below any text from the [CR] that signals that behavior.	Long text
Q10	Why do you think that behavior occurred during this code review?	Long text
Q11	Please share anything else important that we should know about this [CR].	Long text
Q12	Please check this box if we may we contact you by email about these responses if we have follow-up questions.	Checkbox

B.1.4 Part 3 - A [CR] for which you had no involvement. This set of questions asks about cr/[XXX]. Please take a moment to review the [CR] and then return to this page to answer the following questions.

Item#	Question	Response(s)
Q13	Given the importance and initial quality of this [CR], the amount of change requested by reviewers was:	Substantially less than was necessary A little less than was necessary The right amount A little more than was necessary Substantially more than was necessary Don't know
Q14	The reviewers [accepted]:	Quite a bit before the [CR] was of sufficient quality A little before the [CR] was of sufficient quality When the [CR] was of sufficient quality A little after the [CR] was of sufficient quality Quite a bit after the [CR] was of sufficient quality Don't know
Q15	Setting aside inherent difficulties in coding, how frustrating did it seem for the author to get this [CR] through review?	None A little A moderate amount A lot Don't know
Q16	How much interpersonal conflict was there between the author and reviewer(s) in this [CR]?	None A little A moderate amount A lot Don't know
Text	Based on our prior research at Google, we know that most [CR]s contain respectful interactions. But we also want to understand those rare, negative behaviors. Congruent with blameless postmortems, we're not interested in placing blame after the fact, but we are looking for examples to avoid future unnecessary conflict.	
Q17	Given any frustration and interpersonal conflict you noted above, please copy and paste below any text from the [CR] that signals that behavior.	Long text
Q18	Why do you think that behavior occurred during this code review?	Long text
Q19	Please share anything else important that we should know about this [CR].	Long text
Q20	Please check this box if we may we contact you by email about these responses if we have follow-up questions.	Checkbox

B.1.5 Part 4 - Choosing a [CR]. This form is a continuation of a survey[1] about code review at Google from the [internal research team name] team. You should have landed on this page from the final page of that survey. If you did not land on this page through the primary survey link (or redirecting to this page to submit additional [CR]s), please exit this page and do not complete the form.

The core survey is confidential, this form is anonymous; we do not collect your username as part of this form.

If you have any questions about this survey, please email <research team email alias>.

Item#	Question	Response(s)
text	Based on our prior research at Google, we know that most [CR]s contain respectful interactions. But we also want to understand those rare, negative behaviors. Congruent with blameless postmortems, we're not interested in placing blame after the fact, but we are looking for examples to avoid future unnecessary conflict. Reviewing the list of behaviors in the next question, can you provide a [CR] which exhibits one or more of those behaviors? (If there are other behaviors which have prompted a negative experience, please use the Other box in the next question to elaborate). This can be a [CR] that you were an author of, a reviewer of, or neither. If you do not have such as [CR] to share, please jump to the section labeled "Your own experience".	
Q21	Which, if any, behaviors occurred during this code review? (Check all that apply.)	[See list below]
Q22	Given the behavior you noted above, please copy and paste below any text from the [CR] that signals that behavior.	Long text
Q23	Why do you think that behavior occurred during this code review?	Long text
Q24	Please share anything else important that we should know about this [CR].	Long text
Q25	Is there anything you would like to share about negative/positive experiences you have had with the code review process?	Long text

Behavior list for Q21: Which, if any, behaviors occurred during this code review? (Check all that apply.)

- Attacks on work quality
  - Attempts to undermine effort
  - Attempts to undermine work
  - Attempts to undermine personal integrity
  - Unjustified monitoring of your work
- Coercion
  - Confrontational comments
  - Aggression
  - Unnecessary pressure to make changes
- Harsh communication
  - Harsh feedback
  - Curtness
  - Unjustified criticism
- Unhelpful technical contributions
  - Excessive nitpicking
  - Request for out-of-scope change
  - Shifting of goal posts
  - Setting unreasonable deadlines
- Personal attacks
  - Teasing
  - Belittling
  - Inappropriate jokes about a Google employee(s)
  - Humiliation
  - Attempts to demoralize
- Poor quality review

[1]Note, that we implemented part 4 in a different form so we can anonymously collect these responses.

  - Requesting a change without justification
  - Withholding of necessary information
  - Lots of automated warnings in comments
  - Freezing out, ignoring, or excluding
- Very slow reviews
  - Long wait for review to start
  - Excessive review delays
- Threat to professional status
  - Intimation of disciplinary procedures
  - Intimation of negative perf repercussions
- Other [text box]

B.2 Survey questions and undesirable response options for analysis

Question Text	**Response Options** **Undesirable behaviors are bolded**
Given the importance and initial quality of this [CR], the amount of change requested by reviewers was:	Substantially less than was necessary A little less than was necessary The right amount **A little more than was necessary** **Substantially more than was necessary** Don't know
The reviewers [accepted]:	Quite a bit before the [CR] was of sufficient quality A little before the [CR] was of sufficient quality When the [CR] was of sufficient quality **A little after the [CR] was of sufficient quality** **Quite a bit after the [CR] was of sufficient quality** Don't know
[Own CR Authors/Reviewers] Setting aside inherent difficulties in coding, how frustrating was it for you to get this [CR] through review? [Third Party] Setting aside inherent difficulties in coding, how frustrating did it seem for the author to get this [CR] through review?	None A little **A moderate amount** **A lot** Don't know
How much interpersonal conflict was there between the author and reviewer(s) in this [CR]?	None **A little** **A moderate amount** **A lot** Don't know
[Authors only] When I received feedback about this [CR], I felt positively about submitting a similar change in the future.	**Strongly disagree** **Somewhat disagree** Neither agree or disagree Somewhat agree Strongly agree Don't remember

B.3 Sampling Plan Details

As we were validating 3 candidate metrics plus any combinations of these metrics at a CR-level and we wanted multiple perspectives on these CRs (author, reviewer, third party), we developed a stratified sampling plan to ensure we would have an adequate sample of each metric type and combination for analysis. Any CR might be flagged (or not) as involving pushback on each of the three metrics we were evaluating. Thus, CRs can fall into any one of 8 "categories" with respect to the metrics, as detailed in Table X, below. We selected CRs to cover all the combinations and we assigned CRs to participants to ensure our data set included authors, reviewers, and uninvolved third party developers.

CR Criteria	**Number of surveyed CRs**
0 Flags	250
1 Flag: Long shepherding time	200
1 Flag: Long reviewing time	200
1 Flag: Long # rounds of review	200
2 Flags: Long shepherding time & long reviewing time	100
2 Flags: Long shepherding time & long # rounds	100
2 Flags: Long review time & long # rounds	100
3 Flags: Long shepherding time & long reviewing time & long # rounds	100

Next, we translated the CR-level data we wanted into what this would look like for the 2,500 survey invitations:

Invitation Criteria	**Number of invitations**	**Authors**	**Reviewers**	**Third party CRs**
Both CRs unflagged	250	125 unflagged CR authors	125 unflagged CR reviewers	250 unflagged CRs
Own CR unflagged, other CR probable pushback	250	125 unflagged CR authors	125 unflagged CR reviewers	250 probable pushback CRs
Own CR probable pushback, other CR unflagged	250	125 probable pushback CR authors	125 probable pushback CR reviewers	250 unflagged CRs
Both CRs probable pushback	1750	875 probable pushback CR authors	875 probable pushback CR reviewers	1750 probable pushback CRs

Finally, we developed an algorithm to ensure each CR identified for its own authors and reviewers was distributed as a third party to CR to developers who were not involved in that CR. Note that each CR was distributed twice as a third party CR.

Candidate CRs: Any developer in Google across all product areas. New employees are okay. Limiting to CRs ended and submitted in the 3 months prior to 2018-11-12 which avoided any CRs from the Thanksgiving week.

C REGRESSION RESULTS

C.1 Code Review Attributes

Table 1

	Dependent variable:						
	Frustration	Conflict	Approval Withheld	Excessive Change Requested	Feel Negative for Future CRs	Any Pushback	2+ Pushback Feelings
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Size: S	1.626 p = 0.434	2.072 p = 0.240	1.765 p = 0.314	3.032 p = 0.173	4.723 p = 0.050**	2.611 p = 0.013**	2.168 p = 0.205
Size: M	2.383 p = 0.137	1.831 p = 0.320	1.854 p = 0.254	2.978 p = 0.171	2.647 p = 0.229	2.227 p = 0.035**	2.427 p = 0.132
Size: L	2.609 p = 0.108	2.542 p = 0.128	2.117 p = 0.180	4.753 p = 0.050**	1.305 p = 0.769	2.460 p = 0.022**	3.049 p = 0.062*
Size: XL	1.886 p = 0.491	4.280 p = 0.081*	1.815 p = 0.500	4.557 p = 0.146	2.091 p = 0.563	2.912 p = 0.068*	3.935 p = 0.094*
Needs Readability	1.681 p = 0.133	1.135 p = 0.730	1.408 p = 0.327	1.063 p = 0.884	1.316 p = 0.548	1.388 p = 0.172	0.966 p = 0.921
Readability Review	0.901 p = 0.846	0.373 p = 0.202	0.784 p = 0.678	1.441 p = 0.551	0.00000 p = 0.988	0.836 p = 0.652	0.699 p = 0.583
New Employee	1.594 p = 0.191	0.920 p = 0.844	1.257 p = 0.551	0.929 p = 0.879	1.120 p = 0.824	1.060 p = 0.832	1.391 p = 0.397
Num. Reviewers: 2	0.830 p = 0.580	1.594 p = 0.202	1.042 p = 0.901	1.056 p = 0.891	0.512 p = 0.149	1.181 p = 0.470	1.031 p = 0.929
Num. Reviewers: 3	1.341 p = 0.465	1.918 p = 0.144	0.554 p = 0.260	0.821 p = 0.720	0.668 p = 0.503	1.031 p = 0.922	1.144 p = 0.756
Num. Reviewers: 4+	0.745 p = 0.712	1.354 p = 0.706	1.835 p = 0.325	0.611 p = 0.649	1.135 p = 0.908	1.712 p = 0.265	0.461 p = 0.465
Author Level: 2	1.442 p = 0.289	1.350 p = 0.394	1.437 p = 0.309	1.277 p = 0.565	0.306 p = 0.026**	1.230 p = 0.389	1.182 p = 0.635
Author Level: 3	0.974 p = 0.952	0.279 p = 0.030**	0.945 p = 0.896	0.900 p = 0.836	0.654 p = 0.395	0.676 p = 0.184	0.724 p = 0.460
Author Level: 4	1.122 p = 0.868	0.264 p = 0.213	1.058 p = 0.935	0.805 p = 0.793	0.517 p = 0.426	0.602 p = 0.305	0.493 p = 0.374
Author Level: 5+	0.00000 p = 0.984	0.00000 p = 0.984	0.00000 p = 0.984	0.00000 p = 0.990	0.00000 p = 0.996	0.00000 p = 0.983	0.00000 p = 0.984
Constant	0.033 p = 0.000***	0.037 p = 0.00000***	0.043 p = 0.00000***	0.019 p = 0.00000***	0.043 p = 0.0001***	0.114 p = 0.000***	0.042 p = 0.00000***
Observations	606	606	606	606	606	611	611
Log Likelihood	−182.199	−155.920	−179.090	−135.045	−108.964	−314.064	−174.660
Akaike Inf. Crit.	394.399	341.840	388.180	300.090	247.927	658.127	379.320

Note: *p<0.1; **p<0.05; ***p<0.01

C.2 Robustness of metrics to confounding factors

Table 2

	Dependent variable:						
	Frustration	Conflict	Approval Withheld	Excessive Change Requested	Feel Negative for Future CRs	Any Pushback	2+ Pushback Feelings
	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Long Shepherding	9.859 p = 0.039**	26,574,897.000 p = 0.987	2.019 p = 0.315	7.002 p = 0.084*	0.869 p = 0.864	3.464 p = 0.009***	10.097 p = 0.037**
Long Reviewing	7.993 p = 0.060*	12,554,240.000 p = 0.987	1.887 p = 0.348	5.577 p = 0.125	2.529 p = 0.181	2.994 p = 0.019**	6.974 p = 0.082*
Long Rounds	6.679 p = 0.096*	29,180,621.000 p = 0.987	3.744 p = 0.046**	8.667 p = 0.054*	2.551 p = 0.200	4.123 p = 0.003***	13.736 p = 0.018**
Long Reviewing, Rounds	21.484 p = 0.008***	65,854,764.000 p = 0.986	6.804 p = 0.008***	13.377 p = 0.028**	3.002 p = 0.219	5.555 p = 0.002***	38.696 p = 0.002***
Long Shepherding, Rounds	41.353 p = 0.001***	61,590,205.000 p = 0.986	3.315 p = 0.114	4.159 p = 0.269	2.437 p = 0.315	6.164 p = 0.0005***	27.119 p = 0.004***
Long Shepherding, Reviewing	7.379 p = 0.099*	6,619,864.000 p = 0.988	0.646 p = 0.709	3.891 p = 0.292	0.00000 p = 0.992	1.984 p = 0.241	0.00000 p = 0.988
Long Shepherding, Reviewing, Rounds	45.395 p = 0.001***	103,009,703.000 p = 0.986	14.301 p = 0.0003***	19.532 p = 0.011**	3.741 p = 0.132	12.733 p = 0.00001***	64.655 p = 0.0003***
Size: S	1.287 p = 0.702	1.532 p = 0.518	1.591 p = 0.428	2.197 p = 0.346	4.412 p = 0.067*	2.146 p = 0.059*	1.658 p = 0.431
Size: M	1.385 p = 0.600	1.026 p = 0.969	1.460 p = 0.509	1.840 p = 0.458	2.151 p = 0.361	1.557 p = 0.267	1.427 p = 0.567
Size: L	1.080 p = 0.906	1.079 p = 0.908	1.068 p = 0.916	2.247 p = 0.332	0.972 p = 0.976	1.318 p = 0.518	1.212 p = 0.768
Size: XL	0.793 p = 0.816	2.011 p = 0.448	0.961 p = 0.967	2.034 p = 0.518	2.073 p = 0.587	1.648 p = 0.431	1.808 p = 0.516
Needs Readability	1.226 p = 0.584	0.862 p = 0.705	1.103 p = 0.796	0.946 p = 0.900	1.276 p = 0.620	1.177 p = 0.521	0.676 p = 0.313
Readability Review	1.101 p = 0.865	0.419 p = 0.273	0.845 p = 0.784	1.481 p = 0.532	0.00000 p = 0.992	0.875 p = 0.745	0.803 p = 0.749
New Employee	1.312 p = 0.464	0.777 p = 0.564	1.049 p = 0.907	0.790 p = 0.637	1.109 p = 0.844	0.921 p = 0.772	1.125 p = 0.773
Num. Reviewers: 2	0.557 p = 0.105	0.989 p = 0.977	0.769 p = 0.454	0.821 p = 0.628	0.380 p = 0.047**	0.882 p = 0.610	0.631 p = 0.203
Num. Reviewers: 3	0.775 p = 0.560	0.951 p = 0.918	0.306 p = 0.033**	0.552 p = 0.305	0.446 p = 0.213	0.648 p = 0.196	0.532 p = 0.176
Num. Reviewers: 4+	0.466 p = 0.362	0.600 p = 0.544	0.803 p = 0.739	0.307 p = 0.285	0.633 p = 0.682	0.971 p = 0.954	0.170 p = 0.104
Author Level: 2	1.333 p = 0.419	1.368 p = 0.386	1.508 p = 0.269	1.300 p = 0.545	0.270 p = 0.016**	1.222 p = 0.420	1.111 p = 0.775
Author Level: 3	1.032 p = 0.943	0.309 p = 0.049**	0.988 p = 0.978	0.918 p = 0.869	0.604 p = 0.332	0.686 p = 0.215	0.753 p = 0.536
Author Level: 4	1.473 p = 0.592	0.384 p = 0.381	1.293 p = 0.720	0.981 p = 0.983	0.471 p = 0.380	0.691 p = 0.470	0.594 p = 0.530
Author Level: 5+	0.00001 p = 0.985	0.00000 p = 0.998	0.00000 p = 0.985	0.00000 p = 0.990	0.00000 p = 0.997	0.00000 p = 0.983	0.00000 p = 0.996
Constant	0.007 p = 0.00001***	0.000 p = 0.985	0.026 p = 0.00000***	0.006 p = 0.00002***	0.035 p = 0.0001***	0.058 p = 0.000***	0.009 p = 0.00002***
Observations	606	606	606	606	606	611	611
Log Likelihood	−164.532	−139.097	−165.804	−128.557	−103.490	−298.041	−152.277
Akaike Inf. Crit.	373.065	322.195	375.607	301.115	250.980	640.082	348.554

Note: *p<0.1; **p<0.05; ***p<0.01

C.3 Third-party evaluators detect pushback similar to authors

Table 3

	Dependent variable:					
	Frustration	Conflict	Approval Withheld	Excessive Change Requested	Any Pushback	2+ Pushback Feelings
	(1)	(2)	(3)	(4)	(5)	(6)
Long Shepherding	2.809 p = 0.055*	1.809 p = 0.304	3.664 p = 0.066*	4.084 p = 0.220	3.177 p = 0.004***	1.491 p = 0.550
Long Reviewing	1.601 p = 0.402	2.454 p = 0.102	5.290 p = 0.013**	5.765 p = 0.114	3.478 p = 0.002***	2.232 p = 0.196
Long Rounds	1.805 p = 0.326	1.416 p = 0.580	2.321 p = 0.264	8.313 p = 0.057*	2.702 p = 0.019**	1.341 p = 0.670
Long Reviewing, Rounds	4.765 p = 0.010***	2.083 p = 0.276	1.937 p = 0.455	14.295 p = 0.019**	4.511 p = 0.001***	2.586 p = 0.187
Long Shepherding, Rounds	4.032 p = 0.022**	2.458 p = 0.161	4.936 p = 0.038**	1.854 p = 0.674	4.440 p = 0.001***	1.903 p = 0.386
Long Shepherding, Reviewing	0.888 p = 0.878	0.837 p = 0.820	7.045 p = 0.008***	10.927 p = 0.035**	3.062 p = 0.014**	1.385 p = 0.669
Long Shepherding, Reviewing, Rounds	8.304 p = 0.0003***	3.478 p = 0.047**	4.628 p = 0.051*	28.029 p = 0.003***	8.255 p = 0.00001***	5.935 p = 0.007***
Size: S	0.865 p = 0.771	3.175 p = 0.082*	0.903 p = 0.840	1.118 p = 0.882	1.127 p = 0.724	1.850 p = 0.322
Size: M	1.268 p = 0.603	3.995 p = 0.032**	0.869 p = 0.768	1.829 p = 0.371	1.397 p = 0.291	2.205 p = 0.182
Size: L	1.166 p = 0.758	4.738 p = 0.020**	1.134 p = 0.802	1.563 p = 0.535	1.488 p = 0.239	2.744 p = 0.104
Size: XL	3.011 p = 0.074*	3.549 p = 0.150	0.771 p = 0.733	2.456 p = 0.292	2.092 p = 0.106	3.743 p = 0.083*
Needs Readability	0.963 p = 0.901	0.902 p = 0.731	1.018 p = 0.958	1.171 p = 0.670	1.063 p = 0.766	0.777 p = 0.448
Readability Review	1.035 p = 0.942	0.758 p = 0.592	0.937 p = 0.903	0.853 p = 0.774	0.984 p = 0.961	0.856 p = 0.772
New Employee	0.985 p = 0.969	0.711 p = 0.404	0.681 p = 0.370	0.829 p = 0.665	0.780 p = 0.326	0.807 p = 0.626
Num. Reviewers: 2	0.833 p = 0.557	1.252 p = 0.475	1.136 p = 0.710	0.775 p = 0.517	0.976 p = 0.906	1.266 p = 0.495
Num. Reviewers: 3	0.620 p = 0.258	0.960 p = 0.926	1.142 p = 0.781	0.671 p = 0.424	0.722 p = 0.254	0.977 p = 0.960
Num. Reviewers: 4+	1.211 p = 0.720	1.391 p = 0.585	4.162 p = 0.007***	2.344 p = 0.120	2.144 p = 0.039**	3.130 p = 0.029**
Author Level: 2	2.065 p = 0.027**	1.354 p = 0.352	1.025 p = 0.941	1.048 p = 0.896	1.169 p = 0.453	2.147 p = 0.033**
Author Level: 3	1.829 p = 0.114	1.083 p = 0.837	0.720 p = 0.430	0.626 p = 0.342	0.953 p = 0.847	1.134 p = 0.779
Author Level: 4	1.105 p = 0.886	0.866 p = 0.831	0.570 p = 0.476	0.688 p = 0.652	0.734 p = 0.498	0.787 p = 0.768
Author Level: 5+	0.00000 p = 0.987	0.00000 p = 0.986	2.441 p = 0.435	0.00000 p = 0.986	1.033 p = 0.977	0.00000 p = 0.987
Constant	0.018 p = 0.000***	0.009 p = 0.000***	0.017 p = 0.000***	0.005 p = 0.00001***	0.041 p = 0.000***	0.008 p = 0.000***
Observations	1,082	1,083	1,084	1,086	1,182	1,182
Log Likelihood	−249.203	−237.277	−215.768	−171.095	−470.425	−213.961
Akaike Inf. Crit.	542.405	518.555	475.536	386.191	984.850	471.921

Note: *p<0.1; **p<0.05; ***p<0.01