GenderMag Improves Discoverability in the Field, Especially for Women

A Multi-Year Case Study of Suggest Edit, a Code Review Feature

Emerson Murphy-Hill
emersonm@google.com
Google
USA

Alberto Elizondo
albertelo@google.com
Google
Germany

Ambar Murillo
ambarm@google.com
Google
Germany

Marian Harbach
mharbach@google.com
Google
Germany

Bogdan Vasilescu
empirical@google.com
Google
USA

Delphine Carlson
delphinec@google.com
Google
Germany

Florian Dessloch
dessloch@google.com
Google
Germany

ABSTRACT

Prior research shows that the GenderMag method can help identify and address usability barriers that are more likely to affect women software users than men. However, the evidence for the effectiveness of GenderMag is limited to small lab studies. In this case study, by combining self-reported gender data from tens of thousands of users of an internal code review tool with software logs data gathered over a five-year period, we quantitatively show that GenderMag helped a team at Google (a) correctly identify discoverability as a usability barrier more likely to affect women than men, and (b) increase discoverability by 2.4x while also achieving gender parity. That is, compared to men using the original code review tool, women and men using the system redesigned with GenderMag were both 2.4x more likely to discover the "Suggest Edit" feature at any given time. Thus, this paper contributes the first large-scale evidence of the effectiveness of GenderMag in the field.

KEYWORDS

software features, feature discovery, UX design, gender, inclusion

ACM Reference Format:
Emerson Murphy-Hill, Alberto Elizondo, Ambar Murillo, Marian Harbach, Bogdan Vasilescu, Delphine Carlson, and Florian Dessloch. 2024. GenderMag Improves Discoverability in the Field, Especially for Women: A Multi-Year Case Study of Suggest Edit, a Code Review Feature. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3597503.3639097

ICSE '24, April 14–20, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0217-4/24/04.
https://doi.org/10.1145/3597503.3639097

1 INTRODUCTION

Software has the potential to make all humans productive, better informed, and more connected to other people. But sometimes software does not support all of its users the way it should. For example, a recent lawsuit alleges that one of the largest online home goods retailer's website often does not work well for users of screen readers.[1] As another example, a recent study of AI detectors for student essays found that they were more likely to misclassify essays as being written by a bot when the writers spoke English as a second language.[2] As a final example, Black users of a popular social media platform threatened to leave because the platform allowed racial slurs in usernames.[3] These examples illustrate how a variety of software can be exclusionary in a variety of ways.

One of the most well-studied design methods for creating more inclusive software is GenderMag [5]. GenderMag was created as a way for software developers to make their software more gender inclusive. GenderMag is a specialization of the cognitive walkthrough method [16], where software developers take the perspective of different users by applying personas [25], then step through a use case of their software as if they were those personas. The personas in GenderMag were carefully designed based on the literature to reflect five specific cognitive styles that guide user interactions with software and also cluster by gender, on average. For example, the Abi persona is a "process-oriented learner" where they prefer learning an unfamiliar technology through step-by-step guidance. If the software happens to cater better to "tinkerers," Abi incurs an additional cognitive tax when using it because they have to figure out what a feature does by clicking around. Neither cognitive style is inherently "good" or "bad." However, when one style is insufficiently supported, the software becomes less usable by a segment of

[1] https://topclassactions.com/disability-class-action-lawsuit/wayfair-class-action-alleges-website-not-accessible-to-blind-visually-impaired-users/

[2] https://www.businessinsider.com/ai-detector-mislabels-essays-non-native-english-speakers-openai-chatgpt-2023-7

[3] https://techcrunch.com/2023/07/17/bluesky-racial-slurs-banned-list-usernames/

the user base. Research has found that the cognitive styles favored by women on average (personified in Abi) are the ones that are more often unsupported [5]. Thus, the GenderMag method helps software developers see the software they are designing through the eyes of users who may be different from them.

Existing research is limited about whether GenderMag finds and fixes gender disparities in software usage. The most direct evidence comes from Vorvoreanu and colleagues [32], who counted task failures in a laboratory study involving 20 faculty members and students, using two versions of the same search engine, before and after GenderMag was applied. The authors found that women had twice as many failures as men with the original version of the software, but the gender gap disappeared after the GenderMag-based redesign. Whether GenderMag is so effective at larger scale and outside the laboratory remain open questions.

In this paper, we contribute the first evaluation on the effectiveness of GenderMag in shaping users' behavior in the field. We do so in the context of Critique, a code review system built at Google and used inside the company. Critique has an existing feature called Suggest Edit, which allows for code reviewers to suggest a change to an author's code directly. After Suggest Edit was deployed for several years, in 2019 we examined it using GenderMag, then redesigned the feature to address the issues that GenderMag found. In 2023, we returned to Critique's usage logs – containing data from tens of thousands of users – to evaluate the redesign's impact.

Our evaluation focuses on one aspect of usability, *discoverability*, which we define as how quickly a group of users discover a feature in a piece of software. For a single user, a feature discovery is the first time that the feature is used in that user's history of using the product. In our context, a discovery is the first time that a code reviewer invokes Suggest Edit and the suggestion is sent for the code's author to see.

In our evaluation of GenderMag, we test two hypotheses:

- that Suggest Edit had a gender gap in discoverability, whereby the feature had higher discoverability for men compared to women, as predicted by GenderMag, and
- that the GenderMag-based redesign of Suggest Edit closed this gender gap, thereby achieving gender parity in discoverability.

By applying a survival analysis, modeling the time to first use of Suggest Edit while controlling for the employees' tenure and role, we confirmed both hypotheses, showing that GenderMag increased discoverability 2.4x overall and achieved gender parity.

In the remainder of this paper, we summarize the GenderMag method and relate prior research in the area (Section 2), describe why and how we applied GenderMag (Section 3), outline our quantitative methods to demonstrate GenderMag's effectiveness (Section 4), and finally describe the results (Section 5).

2 BACKGROUND AND RELATED WORK

GenderMag is a software design method, where participants (typically software developers or user experience researchers) "put themselves in the shoes" of users who are trying to use the software [5]. The main goal of the GenderMag method is to identify potential gender biases in user interfaces and the associated use cases involving those interfaces. One can expect gender biases (and many other types of biases) to creep up in software design, since typically it is less likely that the voices of women[4] are represented on software teams and, therefore, more likely for usability issues that disproportionately affect women to go unnoticed until later in the software lifecycle, if at all.

2.1 The GenderMag Personas

At one level, GenderMag is a cognitive walkthrough method [16], where potential usability issues are found (we call them "findings", for short). However, GenderMag specifically builds on empirical observations and theories of how men and women tend to differ statistically in their views and usage of technology [29], therefore the method is particularly suited to uncover *gendered* usability issues. The gender differences in GenderMag are encapsulated in five dimensions called facets, that each describe a continuous spectrum:

(1) *motivations*, where people at one end of the spectrum are driven by task accomplishment and at the other end of the spectrum by enjoyment of the technology itself;
(2) *information processing styles*, where one end tends to process information comprehensively while the other end tends to process selectively;
(3) *computer self-efficacy*, where one end tends to have low self-confidence in succeeding at tasks while the other end tends to have high self-confidence;
(4) *attitudes towards risk*, where one end tends to avoid risks more than the other; and
(5) *new technology learning style*, where one end is less likely to playfully experiment ("tinker") with new software features than the other [5].

All cognitive styles (ends of the spectrum for each of the five facets above) have their own strengths, but they can all be at a disadvantage if they are not supported by the right problem-solving software environment.

GenderMag embodies the five facets through different personas, driving analysts to walk through specific use case scenarios using the software from the perspective of those personas, thus uncovering usability issues regardless of the analysts' own gender identities. Most commonly, applications of GenderMag use two personas representing the opposite ends of the spectra, one representing the statistically average woman (often named "Abi") and the other representing the statistically average man (often named "Tim") [5]. Concretely, the Abi persona skews towards being motivated by task accomplishment, processing information comprehensively, having lower self-confidence and higher risk aversion, and being less of a tinkerer, characteristics which tend to occur more often in women. The Tim persona represents the other ends of each spectrum. Therefore, by walking through a software use scenario with both the Abi and Tim personas, GenderMag analysts can identify ways to make the software more gender-inclusive overall, since every user is expected to fall somewhere in between on each spectrum. It is important to note that the above-mentioned gender differences are statistical – while they tend to cluster by gender, individuals exhibit characteristics associated with both Abi and Tim.

[4] Possibly non-men more generally, though data on the representation of non-binary developers is scarce.

2.2 Prior Work on the Effectiveness of GenderMag

Reasoning about the effectiveness of GenderMag typically involves three dimensions: (1) the extent to which the walkthrough generates findings; (2) the extent to which the findings are actually experienced disproportionately by women in practice, as opposed to "merely" imagined as such by the analysts during the walkthrough; and (3) the extent to which redesigning the software based on the findings actually improves its usability and, ideally, eliminates the gender inequities. Prior work has touched on these dimensions to varying degrees.

Whether the Walkthrough Generates Findings. Several studies focused on counting findings generated by GenderMag in real software. In Burnett and colleagues' study of running GenderMag at two teams in US tech companies and two tech teams in a government agency [4], a total of 99 findings emerged, of which 25 were related to GenderMag's facets. Burnett and colleagues also ran GenderMag in a workshop by applying it to a programming game, where 10 findings emerged [4]. In the same study, they applied it to a second programming game, with 15 findings emerging. Cunningham and colleagues [7] had three primary findings when applying GenderMag to a library construction and maintenance tool. Shekhar and Marsden asked 49 students and professionals to apply GenderMag to an 11-step user story in a piece of learning management software; the average participant typically generated at least one and up to three findings per step [28]. In Padala and colleagues' recent study, by asking five teams of software engineering professionals to apply GenderMag to open-source project use cases, between one and 40 total findings emerged per use case [24].

Whether the Findings Are Gendered. In contrast, the literature validating the findings is less rich, and is typically restricted to small samples of software users observed in controlled laboratory settings. Some of the most robust evidence comes from Padala and colleagues, who compared GenderMag's findings about open-source project onboarding against challenges reported by 22 students who journaled their own open-source onboarding experiences; although links between findings and challenges were not investigated directly, the authors argue that GenderMag's findings "really do disproportionately affect women" [24]. In a different controlled study with 20 men and women faculty members and graduate students, Vorvoreanu and colleagues applied GenderMag to a search engine [32]. The authors found that women got stuck twice as often as men with the original version of the software, but the gender gap disappeared after the redesign; moreover, "task performance improved for both the participating genders." Less formally, after Burnett and colleagues applied GenderMag to a game, 13 of 14 findings were reported to be consistent with the development team's observations over a two-year period [5].

Whether Addressing the Findings Improves Usability. Finally, there is evidence that software teams act, or at least plan to act, on the GenderMag findings by redesigning the software to address the usability issues, and that these efforts pay off. However, the evidence on the industrial, practical impact of applying GenderMag is scarce and rarely comes from software deployed "in production" and the software's actual, day-to-day users. Instead, as before, the evidence tends to come from small samples of users observed in laboratory settings. Prior studies discuss the software developers' commitments to fix findings, the post-fix impact, and more broadly the support for using GenderMag also outside of the respective studies.

Commitments to Fix Findings. In Burnett and colleagues' study of four professional teams [4], after applying GenderMag, the development team committed to fix 14 of 25 findings. In an action research study at Microsoft [3], one informant reported fixing a compiler message based on a GenderMag finding. In Hilderbrand and colleagues' action research study of 10 teams from companies and universities [14], "all 10 teams decided to fix their products according to the GenderMag [findings]". In Burnett and colleagues' application of GenderMag to a game, the development team agreed to fix six of the 14 findings [5].

Post-Fix Impact. Three studies have argued for the effectiveness of GenderMag by linking the software changes made after GenderMag was applied with a variety of outcomes. In Burnett and colleagues study at Microsoft, one team reported that customer ratings improved by 40%, but also noted that the ratings were difficult to compare because "they measured slightly different things" [3]. In Burnett and colleagues' application of GenderMag to a game, users were reportedly 47% female, but the gender distribution before GenderMag was not reported [5]. One participant in Hilderbrand and colleagues' study noted that GenderMag's fixes "reduced help desk tickets on common questions" [14]. As mentioned above, Vorvoreanu and colleagues found that GenderMag induced an improvement in task completion for all users while also removing the gender gap [32], while Guizani and colleagues report on a similar overall usability improvement, but without gender differences [13].

Method Adoption. Two studies have argued for GenderMag's effectiveness by noting study participants' desire to adopt GenderMag in their day-to-day work. After asking five teams to use GenderMag, Burnett and colleagues [4] reported that three teams planned to use GenderMag again. After asking 10 teams to use GenderMag, Hilderbrand and colleagues [14] found that seven teams showed evidence of using GenderMag after the study period.

2.3 The Knowledge Gap and Our Contribution

In summary, while it is clear from prior research that GenderMag *can* be highly effective, there is little evidence of its demonstrated impact on industrial practice and real-world users of deployed software.

Moreover, it's notable that nearly all the studies of GenderMag's effectiveness were conducted by researchers at Oregon State University (OSU), where GenderMag was created. Independent evaluations are generally critical for advancing scientific understanding [9], but especially so for GenderMag, for two reasons. First, the largest-scale field studies to date [3, 14] used action research, where "researchers are also participants, and the participants are also researchers" [14], so it remains unclear whether GenderMag would have been as effective without the deep involvement of GenderMag's creators. Second, the two exceptional evaluations that were not conducted by the OSU team – Cunningham and colleagues [7] and Shekhar and Marsden [28] – showed only that GenderMag produces some usability findings, but not that those findings correspond to any challenges experienced by real users, nor that fixes to address those findings would actually yield more usable software.

Our research – conducted without substantial involvement of the OSU team[5] – decisively fills this gap in the literature by reporting on a case study of applying GenderMag to a code review tool used by tens of thousands of engineers at Google, a large software company. Our study shows, with large-scale quantitative evidence from tens of thousands of users, that a sizable gender gap in using the code review tool's Suggest Edit feature, uncovered by applying GenderMag, disappeared after the feature was redesigned. The redesign also resulted in a substantial improvement in the usability of the feature overall, for engineers of all genders.

3 APPLICATION OF GENDERMAG TO SUGGEST EDIT

In the late 2010s, a team within Google that developed the company's internal code review tool, called Critique, needed to rewrite the frontend of Critique because the old frontend framework was being deprecated. This offered the team an opportunity to make usability improvements to Critique. One such improvement started with the application of GenderMag to a specific feature called Suggest Edit. We selected Suggest Edit for two reasons:

- We hypothesized that the overall usability of Suggest Edit could be improved, based on a few user reports. As a cognitive walkthrough method, GenderMag would allow us to broadly examine the usability of Suggest Edit.
- Prior research on GenderMag convinced us that many software features work better for some genders more than others. If Suggest Edit was such a feature, the GenderMag method would help us discover why.

The authors of this paper collaborated with the Critique team to apply GenderMag.

In this section, we describe how Suggest Edit worked before GenderMag, enumerate our major findings from the GenderMag session, and then explain how Suggest Edit was redesigned to incorporate GenderMag's findings.

3.1 How Suggest Edit was Used Before GenderMag

Before we explain how we applied GenderMag, we first describe the software we applied it to, Critique, and specifically a feature within Critique, Suggest Edit.

In Google's main monolithic code repository, an author of a code change generally must have that change reviewed by at least one other Google employee before the change can be integrated into software products. Code authors use the Critique code review tool to solicit reviews, and reviewers asynchronously provide comments on the author's change, typically called a changelist.

In addition to providing comments on a change, reviewers can use the Suggest Edit feature to directly propose changes to a changelist. For example, if the author did not use camel case when they should have, rather than requesting that change in the form of a textual change (e.g., "Please change this variable to camel case."), the reviewer can propose a change directly to the file in the changelist. The author can then preview the reviewer's change, and integrate it into their own change if the author is in agreement. The Suggest Edit feature was introduced into Critique in 2014, and is functionally similar to GitHub's Suggested Changes feature[6] for pull requests, introduced in 2018.

So that the reader understands GenderMag's findings about this feature, in the remainder of this subsection we provide additional details on how reviewers proposed changes in the version of Critique prior to our application of GenderMag. This version of Critique was deprecated years ago and is no longer functional today, and we did not have any appropriate screenshots to include in this paper. Nonetheless, we intersperse two illustrations that show what Critique looked like prior to GenderMag (Figures 1 and 2).

The Suggest Edit use case began when a reviewer received a notification to review a changelist, either by receiving a notification via email or by looking at a Critique dashboard to find a changelist that needed review. The reviewer then clicked on the changelist to review, where metadata appeared, including who the other reviewers were, what static analysis checks were run, and which tests had run. Further down the page, the reviewer saw a list of files that the author modified. Each file could then be opened for review, in one of two ways:

- If the reviewer clicked on a file link, the author was taken to a separate Critique page just for that file. We will call this Critique's "file view," because the entirety of one file was shown on this page. After reviewing the file, the reviewer navigated with their browser's back button to the main changelist page, and then reviewed the next file.
- Alternatively, the reviewer could expand the file bar on Critique's main page. Doing so did not cause the browser to navigate to the file view, but instead the file was expanded in-place, and elided parts of the file that had not changed. We call this Critique's "in-place file view." When the reviewer was done inspecting the relevant parts of the file, rather than navigating back, they only needed to scroll down to the next file to continue the review.

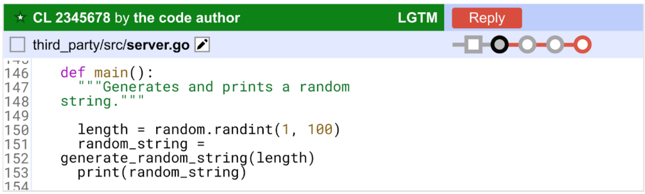
If a reviewer wished to make a comment on a file, they could do so from either view. Crucially, however, the reviewer could use Suggest Edit from *only* in the file view.

From the file view, the reviewer suggested an edit by navigating to the top of the file view page, then clicked on a pencil icon next to the file name (Figure 1). An editable file was then visually overlaid on top of the file being reviewed, inside which the reviewer could make any edit. Once satisfied, the reviewer clicked "Save". After saving the change, the reviewer-made edit was shown just like a normal comment would be, with the comment text "Reviewer-suggested edit (Click 'Show' in Critique)".

Once satisfied with all comments and suggestions made to the changelist, the reviewer opened a "reply" dialog, then clicked "send"

[5] More precisely, the OSU team was not involved in choosing the software we applied GenderMag to, conducting the walkthroughs, or collecting and analyzing any of the data for the evaluation. However, one researcher on the OSU team gave a GenderMag training six months before we used GenderMag, and one researcher provided feedback on a draft of this paper.

[6] https://github.blog/changelog/2018-10-16-suggested-changes/



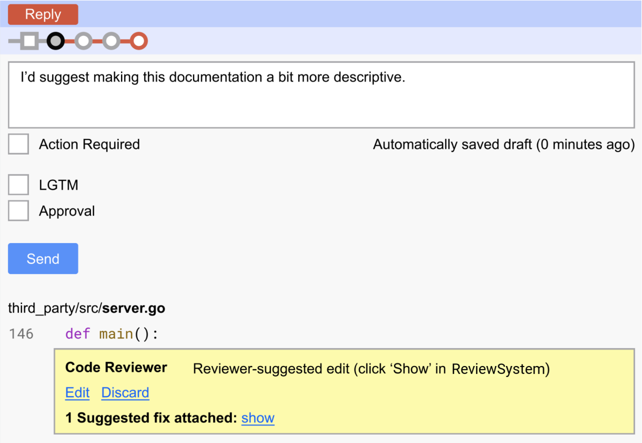
CL 2345678 by the code author LGTM Reply

□ third_party/src/**server.go**

```
def main():
  """Generates and prints a random
string."""

  length = random.randint(1, 100)
  random_string =
generate_random_string(length)
  print(random_string)
```

Figure 1: The file view in the pre-GenderMag version of Critique. To invoke the Suggest Edit feature, the user pressed the pencil icon by the file name.



Reply

I'd suggest making this documentation a bit more descriptive.

□ Action Required Automatically saved draft (0 minutes ago)

□ LGTM

□ Approval

Send

third_party/src/**server.go**

```
146  def main():
```

Code Reviewer Reviewer-suggested edit (click 'Show' in ReviewSystem)

Edit Discard

1 Suggested fix attached: show

Figure 2: The reviewer about to send the Suggested Edit back to the author.

to send the changelist back to the author (Figure 2). The author could then preview any Suggested Fixes, and could apply them to the changelist with a single click.

3.2 How We Applied GenderMag

We applied the GenderMag method to find usability issues in Suggest Edit. We used GenderMag's Tim and Abi personas, which are at the opposite ends of each facet spectrum, over two workshop sessions that roughly lasted 3.5 hours in total. As the method literature recommends, we modified the personas to fit with this context, such that both Tim and Abi were Google software engineers. We kept the default profile pictures, as prior work found that the persona's appearance is not an important differentiator [15]. Also congruent with the literature, the sessions included a facilitator, to ensure reasonable adherence to the GenderMag process; a recorder, to write down findings the group encountered; a driver, to step through the use case on a shared-display computer; and three to four evaluators, to discuss how Tim or Abi might have difficulty at each step of the use case.

After conducting these workshops, we had clearly identified issues we wanted to address. We then ran a two-day design workshop where we took these issues, and re-designed the feature to address them. The outcome of this design workshop were UI mocks of the re-designed workflow, some of which were implemented in the new version of Critique.

Table 1: Number of findings uncovered with GenderMag. The number that were fixed are in parentheses.

	Tim persona	Abi persona
Inclusion Findings	6 (2)	16 (4)
Other Usability Findings	3 (1)	5 (3)

3.3 What We Found Applying GenderMag

Table 1 counts the number of findings we uncovered with GenderMag, and the number that were fixed (in parentheses) by the time the new version of Critique was deployed. Consistent with prior literature [4, 28], the Abi persona produced the most findings. Also as in prior literature [24], we distinguish between findings related to gender inclusion and other usability findings; inclusion findings are those that can be tied to GenderMag's five facets.

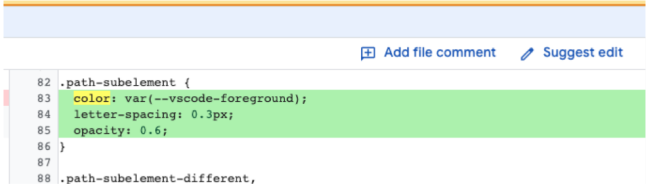
Notably, we fixed only some of the inclusion and other usability findings uncovered with GenderMag. The remainder were not fixed for a several reasons:

- The fix would have been too difficult with limited apparent benefits. For instance, one fix would have entailed integrating a new styled code editor into the code review tool.
- Fixing the issue was of unclear importance. For instance, two issues were likely encountered rarely enough that they seemed not important enough to fix.
- The fix would entail tradeoffs that may not have been worthwhile. For instance, one fix would have worked better for users with a certain mental model, but would have been confusing to users with a different mental model.
- The fix would have required changes to a conceptually separate part of the software. For instance, one fix would have changed Critique's "reply" dialog, but the change would have impacted a variety of other use cases, with unclear implications.

We next enumerate the six inclusion findings that we encountered with GenderMag, and that were fixed. With the Abi persona, we found:

- **F1**: Suggesting an edit is not encouraged by the UI (as opposed to commenting, which has an affordance upon highlighting a piece of code). The feature is hidden as a small icon on the file view. Low self-efficacy/risk averse users may be deterred from using a seemingly uncommon feature.
- **F2**: The Suggest Edit feature is potentially hard to find in the Critique manual for process-oriented learners, as it only appears in the "how do I…" section.
- **F3**: Non-tinkering, process-learning users do not realize that they are making progress towards their goal of suggesting an edit, as the manual starts with "Click on the pencil button…" but that is nowhere to be seen on the main page and having to open the file view is not clearly described as part of the process.
- **F4**: Risk-averse users may be reluctant to press "save" in the editor overlay, as the language can imply that code changes will be permanently saved or sent to the author immediately.

And with the Tim persona, we found:



```
.path-subelement {
  color: var(--vscode-foreground);
  letter-spacing: 0.3px;
  opacity: 0.6;
}

.path-subelement-different,
```

Figure 3: The Suggest Edit button after a GenderMag-based redesign.

- **F5**: While a tinkerer will eventually discover Suggest Edit, it is still not easy to find. This may cause a severe distraction from the original task.
- **F6**: Especially for a tinkering user who does not read documentation, having to first go to the file page is a concern, as there is no indication of progress towards this goal from the main page (while scrolling and figuring out where to go). This could cause distraction or loss of focus during tinkering.

In short, we interpret these findings to mean that users like Abi will be less likely to discover Suggest Edit than users like Tim (though users like Tim will be distracted while using Suggest Edit).

3.4 How Suggest Edit is Used After GenderMag

Based on the findings about Suggest Edit that emerged from GenderMag, we proposed several mockups for the next version of Critique. After our redesign and the Critique team's subsequent implementation, we made the following modifications to how Suggest Edit works, relative to the prior implementation (Section 3.1):

- Addressing **F1**, **F3**, **F5**, and **F6** above, the Suggest Edit button now appears in both the file view and the in-place file view, now has equal prominence to the "Add file comment" button, and has text added next to the icon (Figure 3).
- Addressing **F2**, a help document about Suggest Edit now appears alongside six other major reviewer tasks in the Critique User Guide.
- Addressing **F4**, the "Save" button was relabeled as "Save suggestion", striking a more appropriate, welcoming, and tentative tone.

4 EVALUATION METHODS

How effective was our application of GenderMag in practice? We examine this question in two ways.

First, we investigate to what extent the original Suggest Edit feature in Critique was less usable for women than for men. We expect this gender-based difference because one of our most significant findings from the GenderMag session was that the Suggest Edit feature was difficult to find. The theory behind GenderMag suggests that low discoverability would disproportionately affect women. Indeed, we found more gender issues using the Abi persona than the Tim persona. Therefore, we formulate this mechanism as a testable hypothesis:

> **Hypothesis 1:** In the version of Critique that was not designed using GenderMag, discoverability is significantly higher for men than for women.

Second, assuming Hypothesis 1 is supported, we investigate whether the GenderMag-based redesign increased discoverability, especially for women:

> **Hypothesis 2:** In the version of Critique that was redesigned using GenderMag, discoverability of Suggest Edit increased, especially for women.

Large-scale logs data on user interactions with Critique over time and employee gender data were both available at Google. However, we still had important decisions to make about which outcome variables to measure and how to filter and aggregate the data such that we can more confidently attribute any changes in the discoverability of Suggest Edit between the two versions of Critique to the GenderMag-based redesign, as opposed to possible confounding factors. In this section, we describe the methodology choices we made to statistically test our two hypotheses.

The proposal for this research was reviewed by the company's employee privacy working group, which is somewhat similar to an Institutional Review Board.

4.1 Discoverability Metric: "Time" Until Suggest Edit

We model the discoverability of Suggest Edit primarily as a function of interacting with Critique. Thus, we expect that the more comments one writes across all their different code reviews (i.e., the more they interact with the tool), the more chances they get to discover Suggest Edit. We chose this operationalization over one based on wall clock time as a way to normalize speed by reviewing activity, since we expect that code review activity varies widely between engineers over some fixed period of wall clock time.

Still, this left a choice of whether to measure interaction with Critique in terms of number of code reviews or number of review comments. We chose to measure the discoverability of Suggest Edit as the total number of Critique *comments* made before the first-ever use of Suggest Edit, to better approximate the amount of interaction one had with Critique – more comments implies more chances to suggest edits. The lower this number, the more discoverable it is.

Limitations. We acknowledge the following limitations (threats to *construct validity*) of our operationalization. First, reviewers may discover Suggest Edit also for exogenous reasons, e.g., the feature being advertised explicitly at some point [22], or a colleague recommending it, while we only capture discoverability as a function of interacting with Critique. While such cases are possible, we have no reason to suspect that any such exogenous reasons impacted men and women disproportionately, so this should not substantially affect our conclusions. Second, it is also possible that our discovery marker event – the first use of Suggest Edit – does not accurately reflect discovery. For instance, if a user discovers Suggest Edit, but gets frustrated with it, makes a comment instead, and then never uses it again, our measure would not record a discovery event at all. More research is needed to disentangle possible gender differences in the usability of Suggest Edit *conditioned on being aware of it* from the discoverability of Suggest Edit in the first place, as understood here. GenderMag might be applicable again, but that goes beyond the scope of this work. Finally, we only considered one measure of Suggest Edit's discoverability, the time to discover the feature, while a variety of other effectiveness measures exist for GenderMag, as we explained in the related work section. Future researchers may extend our analysis to include those as well.

4.2 Modeling Strategy: Survival Analysis

To test our hypotheses, we collected and analyzed data from internal company code review logs, which are available for both the original and redesigned versions of Critique and span multiple years.

A naive data analysis approach would compare discovery times of men and women, with both versions of Suggest Edit, using the outcome variable above. For example, in analyzing usage of the pre-GenderMag version of Critique, if we found that the median women discovered Suggest Edit after commenting 50 times, but the median man after 40 times, Hypothesis 1 could be considered supported.

However, this approach leaves out important data – people who never discovered Suggest Edit at all. Continuing the example, if it turned out that 90% of women eventually discovered Suggest Edit but only 50% of men did, that suggests that women actually discover Suggest Edit more easily than men, disconfirming Hypothesis 1. An additional challenge of this binary discovery data – Suggest Edit is discovered or not discovered – is that in practice, we cannot determine whether someone will never discover Suggest Edit, or simply has not discovered it yet.

Instead, we use survival analysis to account for both binary discovery data and discovery time data. Survival analysis is a statistical technique that is typically used to estimate the survival of patients with a disease [1]. It can be used to compare patient survival outcomes between different patient types, such as men and women, and those with and without a treatment.

In our survival analyses, how long it takes to discover Suggest Edit is analogous to how long a patient survives. Our data about a user who has not yet discovered Suggest Edit is analogous to a patient who has not died, typically called a *right censored* event. Users who have used the versions of Critique with and without a GenderMag design session are analogous to users with and without treatment for disease.

While the problem of software features not being discovered is well-studied in the literature [10, 12, 17, 21], we have not yet seen a survival analysis of the problem. Other research has used survival analysis in software engineering, including to understand defect proneness [30], developer turnover [18, 26], library migration [11], and project abandonment [31] in open source.

In our analysis, we use two well-worn tools from the survival analysis toolbox. The first is a Kaplan-Meier plot [2], which is useful for visualizing survival curves, or in our case, how Suggest Edit tends to get discovered over time under various conditions. The second is a Cox proportional hazards regression [8], which enables us to separate out the effects of various independent variables (gender and Critique version) on discoverability, while controlling for confounding variables.

Like any observational study, our ability to isolate the effect of interest from confounding variables is limited. Nonetheless, we expected and attempted to control for two main confounds:

Experience with other code review tools. To our knowledge, the only other code review tool that supports a similar feature is GitHub's, which began supporting it in October 2018. Thus, we expected that prior experience with this GitHub feature could cause Critique users to look for the analogous feature, increasing the rate of Suggest Edit discovery. Conversely, we expected that significant prior experience with code review systems that do not have Suggest Edit-like functionality would decrease Suggest Edit discovery in Critique. Unfortunately, there was no way for us to reliably measure prior code review tool experience at scale, so we resorted to a proxy measure, the user's level as an employee, such as "junior" or "senior staff"-level. In short, we expected that more junior engineers may be more likely to have experience with the recently added Suggest Edit-like feature in GitHub. Likewise, controlling for level should also account for prior work's findings suggesting that "junior [software engineers] have more free time to explore" their programming environments [21].

Job role. We also expected different roles may be more or less likely to discover Suggest Edit, because they may have different expectations about the existence of similar functionality. For example, word processors typically have "track changes" functionality, which functions similarly to Suggest Edit, so users who were previously familiar with tracking changes may likewise expect to be able to Suggest Edits in Critique. Again as a proxy measure, and following prior quantitative work about code review at Google [20], we modeled users' official role as Software Engineer (the bulk of Critique users), Site Reliability Engineer, Other Engineer (for example, Research Scientist Engineer), and Other.

Testing the two hypotheses involves two (sets of) comparisons – between the discoverability of the original and redesigned versions of Critique, and between women and men employees. The Cox proportional hazards regression framework allows us to carry out both simultaneously, when specified as:

```
Surv(last_event_num, discovered) ~
    job_role + job_level + version * gender,
```

where the pair (`last_event_num`, `discovered`) represents the outcome variable, the overall index of one's review comment when Suggest Edit was first used (if `discovered = true`) or the data became right censored (`discovered = false`); `job_role` and `job_level` are the two control variables above; and `version * gender` is an interaction term between the version of Critique used and gender – considering men users of the original version of Critique as the baseline, the resulting three estimated coefficients describe how the discovery probability varies for women users of the original version, men users of the redesigned version, and women users of the redesigned version.

We implemented our quantitative analysis in a combination of SQL and R. All analysis code was written by one of the co-authors of this paper and formally code reviewed by another co-author. When interpreting the Cox regression, we consider p-values below .05 to be statistically significant and report the estimated hazard ratios as measures of effect size.

4.3 Data

While the historical code review data available to us captures when employees used Critique to make comments and use Suggest Edits,

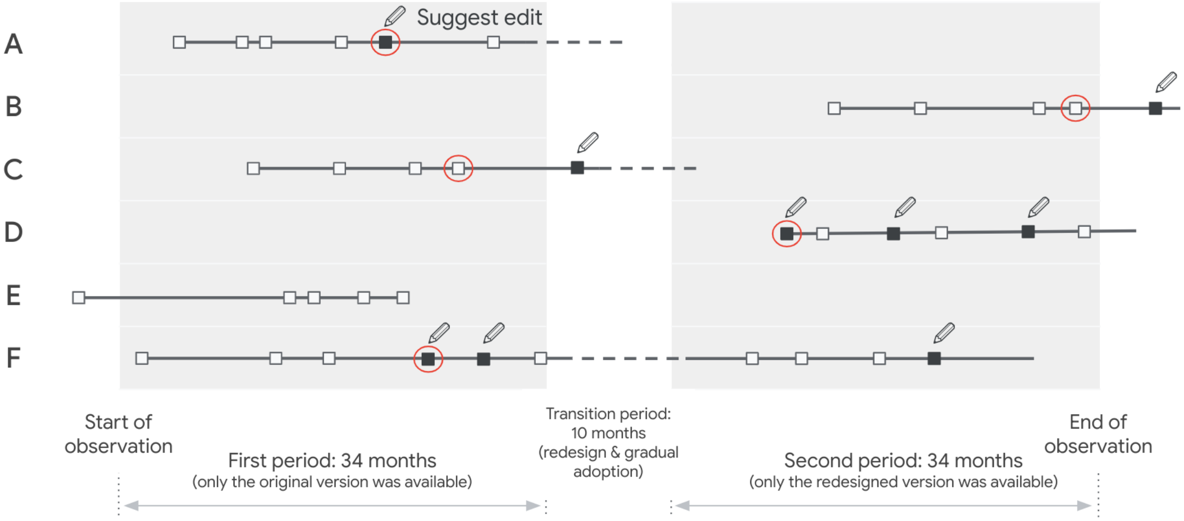

Suggest edit
A
B
C
D
E
F
Start of observation
First period: 34 months
(only the original version was available)
Transition period: 10 months (redesign & gradual adoption)
Second period: 34 months
(only the redesigned version was available)
End of observation

Figure 4: Data filtering approach to create two groups (Critique with the original vs. redesigned Suggest Edit feature) that can be compared fairly. The times when the employee first used Suggest Edit or the data becomes right-censored are circled.

comparing the original and redesigned versions is not straightforward, for many reasons. For example, there is more historical data available for the original version compared to the redesigned one, simply because the redesign is relatively recent, resulting in potentially more right-censored observations for the latter group. In addition, the redesign and transition to the new version did not happen overnight, or even at the same time for all employees, e.g., some employees continued to use the old Critique while the new Critique reached feature parity. This means that there is no single clear transition point. Finally, some employees were exposed to and used both versions of Critique, while others joined the company after only the redesigned version was available, or left the company before the redesign.

To allow for a sound comparison, we first filter the historical code review data to create two distinct, non-overlapping but comparable, equal-length periods (the `version` flag in our Cox regression distinguishes between them):

- The most recent (Second) period is bounded by a start date on which the version of Critique containing the redesigned Suggest Edit was fully deployed to production. The end point was around the date this analysis was performed. The period is 34 months long and only includes employees whose first-ever Critique comment or Suggest Edit was recorded during this interval. That is, we excluded people that used the original Critique and, therefore, the remaining people are not biased in some way by their previous code review experience with the original version. In Figure 4, employees B and D started reviewing any code directly with the redesigned Critique. D suggested an edit on their first comment (and other times thereafter), while B is right censored and is recorded as not yet having discovered Suggest Edit, even though they perhaps discovered it shortly after the end of the observation period. Employee F is not considered for this period, even though they also used the redesigned Suggest Edit, because they have prior experience with the original version.
- The prior (First) period is bounded by an end date on which the original Critique, designed without GenderMag, was used by all employees except a small fraction (for example, early dogfooders of the revised Critique). The start date was defined as 34 months before that date, to ensure the two periods have equal length. In Figure 4, employees A, C, and F started reviewing any code during this period and are considered, while E is "too old" to be considered for this period. A and F eventually used Suggest Edit during this period (the first use is the one that counts), while C is recorded as right censored – on the last comment of the period, the comment that counts, they had yet to use Suggest Edit, even though they used it shortly thereafter.

Note that we deliberately captured two equal length periods, so that discovery times would be directly comparable. In addition, we deliberately excluded the time between the two periods, about 10 months in total, to account for uncertainty during the period in which both versions of Critique co-existed. Moreover, a single user cannot contribute data to both periods.

For comment histories (the `last_event_num` variable) we calculated the number of comments up to and including the first use of Suggest Edit, or, if the user never used Suggest Edit, then number of comments made before the end of the period. While rare, we excluded users from our analysis who used both the older and newer versions of Critique. All users were full-time Google employees.

We gathered gender data for our sample from internal human resources data, which is 99% complete for code reviewers. Although gender is not a binary, the data available to us only included male and female as labels. More research is needed to capture and understand the experience of non-binary software developers.

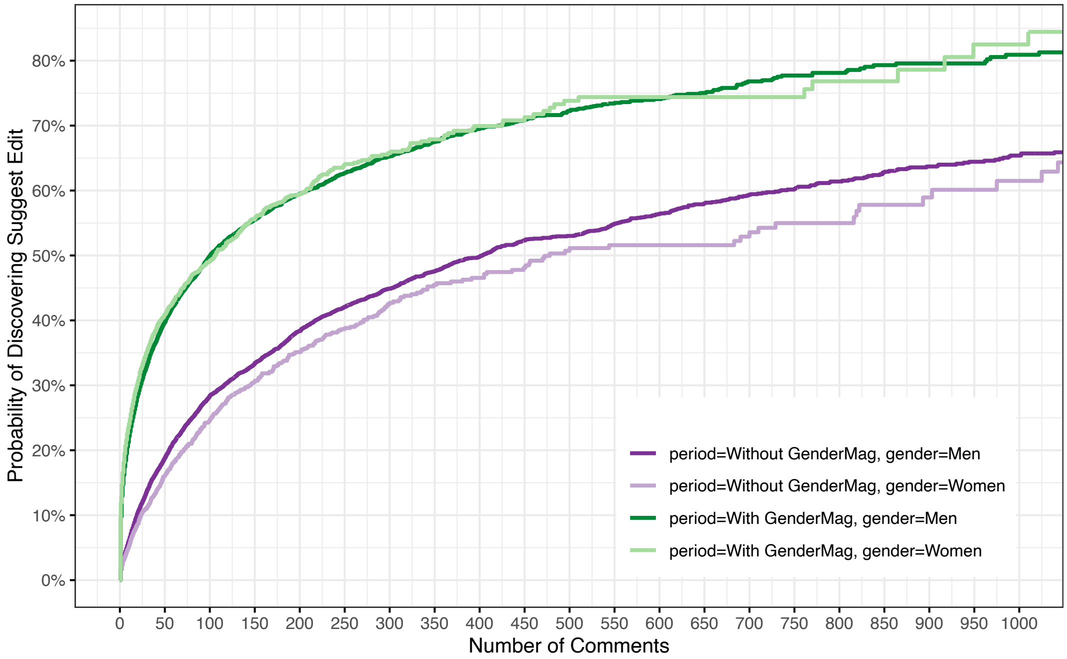


Figure 5: Kaplan-Meier plot showing the probability of discovering Suggest Edit over time, for four different groups.

In sum, we included data from 21,109 users for Critique without GenderMag (of which 4,087 were women), and for the period with GenderMag, we included 31,646 users (7,274 women).

Limitations. Taken together, our modeling strategy and sampling approach have several limitations (giving rise to several threats to *internal validity*), with respect to how confidently we can conclude that GenderMag is the cause of the observed results. Since the two observation periods occurred at two distinct periods in time, some event other than GenderMag may have influenced discovery rates. For example, it is possible that the COVID19 pandemic and resulting lockdown may have somehow increased discoverability for the post-GenderMag version of Suggest Edit. Similarly, after looking at the discovery data for the post-GenderMag period, we were surprised to find that many Critique users used Suggest Edit before they made a comment (technically, as their first comment in our operationalization). We were concerned that perhaps the new engineer training – where engineers complete a tutorial where they review each others’ code – was modified recently to recommend Suggest Edit. Fortunately, we found no such evidence, but it is still possible that other sources of causal interference are present. A more conclusive experiment would have deployed both pre- and post-GenderMag versions of Critique simultaneously, randomly assigning users to one or the other. Such an experiment was infeasible in this case, because maintaining two separate versions of the same software, especially over a multi-year period, was impractical.

It is also worth noting a threat to *external validity* that tempers our ability to generalize to other contexts. Namely, our study was conducted at just one company, for just one feature, in just one piece of software, and by just one team. Other companies, features, software, or teams may experience different results. However, the literature described in the Related Work Section shows that GenderMag, broadly speaking, is applicable in a variety of other organizations for a variety of types of features and software.

5 RESULTS

Before GenderMag, 23% of users of Critique discovered Suggest Edit; of those discoverers, the median number of comments made before Suggest Edit was invoked was 37. In comparison, after GenderMag, 35% of users discovered Suggest Edit; of those discoverers, the median number of comments made before Suggest Edit was invoked was 5. These numbers suggest that GenderMag’s redesign increased discoverability, and we next show how discoverability varied by gender (Section 5.1) and then test our hypotheses formally (Section 5.2).

5.1 Suggest Edit Discovery Curves

Figure 5 shows a Kaplan-Meier plot for our data, which compares the number of comments on the x-axis to the probability of discovering Suggest Edit on the y-axis. We have truncated the x-axis to 1,000 comments, because few users made so many comments, reducing the certainty of estimates at the far right. In the plot, four different lines are shown:

- Men using the Critique designed without GenderMag, in dark purple (🟪).
- Women using the Critique without GenderMag, in light purple (🟪).
- Men using the Critique designed with GenderMag, in dark green (🟩).
- Women using the Critique with GenderMag, in light green (🟩).

To understand Figure 5, let us consider each series at x=1,000 comments, at the far right of the plot. Moving from bottom to top, we see that the model estimates that about 62% of women using the code review system designed without GenderMag find Suggest Edit by their 1,000th comment; in comparison, about 65% of men using the same system find Suggest Edit by their 1,000th comment. Moving upwards to the green lines describing the system designed with GenderMag, we next see that the model estimates 81% of men discover Suggest Edit by their 1,000th comment, compared to 83% of women.[7]

Examining Figure 5, we can make the following observations:

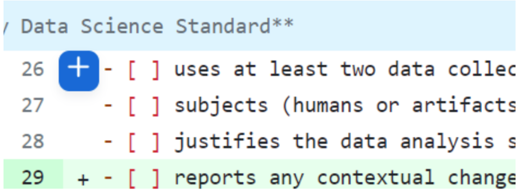
- The largest discovery differences are between With and Without GenderMag versions of the code review system. This suggests that all users benefited, in terms of discovery, from GenderMag's cognitive walkthrough.
- Given the uniformly lower line, women tend to discover Suggest Edit more slowly than men using the system without GenderMag. This suggests a confirmation of Hypothesis 1, but we provide further confirmatory evidence below.
- The gender gap in discovery appears to be closed for the code review system designed with GenderMag, providing some evidence in support of Hypothesis 2; we examine this further below.

While the above provides some evidence for our hypothesis, we next test the hypotheses directly.

5.2 Testing Hypotheses

We estimate the Cox regression model described above on the same survival data to formally test our hypotheses. After controlling for job role and level, we find that, compared to men using the code review tool designed without GenderMag:

- Women using the system designed without GenderMag were 18% (95% confidence interval is 11-24%) less likely to discover Suggest Edit than men at any given time ($\beta = -0.19752, HR = 0.82076, SE = 0.03977, z = -4.966, p = 6.83e - 07$). This finding quantifies the gap between purple lines in Figure 5, and supports Hypothesis 1.
- Men using the system designed with GenderMag were 2.4 times (CI=2.27-2.46) more likely to discover Suggest Edit than when using the system designed without GenderMag ($\beta = 0.86180, HR = 2.36741, SE = 0.01944, z = 44.327, p < 2e - 16$).
- Women using the system designed with GenderMag were 2.4 times (CI=2.28-2.53) more likely to discover Suggest Edit than Men using the system designed without GenderMag ($\beta = 0.876632, HR = 2.402793, SE = 0.025823$). Given the practically equivalent estimates and confidence intervals for men and women's discovery of Suggest Edit after GenderMag, we conclude that Hypothesis 2 is supported.



```
/ Data Science Standard**
26    - [ ] uses at least two data collec
27    - [ ] subjects (humans or artifacts
28    - [ ] justifies the data analysis s
29  + - [ ] reports any contextual change
```

Figure 6: The "+" button on a GitHub pull request, indicating that a user can add a comment to line 26.

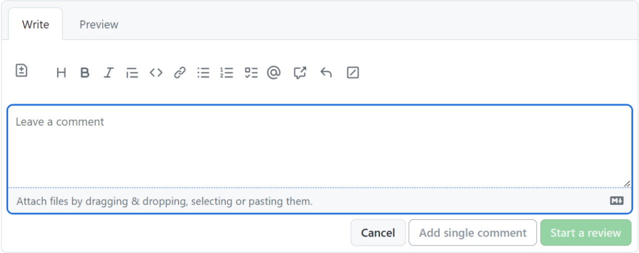


Figure 7: Once the "+" button is clicked in Figure 6, this dialog appears on a GitHub pull request. Pressing the "±" page icon on the toolbar invokes the Suggest Edit-like feature.

6 DISCUSSION

While the evidence presented here suggests that indeed discoverability for Suggest Edit was increased by GenderMag, an astute reader might wonder whether we could have just as easily increased the discoverability of Suggest Edit with some other method, such as a general cognitive walkthrough. Moreover, looking back over this neatly-told usability story, it seems obvious that discoverability was a major usability barrier, considering that the Suggest Edit button was hidden away in an unexpected place and that it used a cryptic pencil icon. We argue that the discoverability barrier was only obvious in retrospect. As evidence, consider GitHub's Suggest Edit functionality, which exhibits a very similar discoverability problem:[8]

- When a reviewer on GitHub wants to suggest an edit to the file, they must first open a view indicating that they want to make a comment (Figure 6).
- Once the view to make a comment is open, the reviewer then clicks on the cryptic "±" page icon to suggest an edit (Figure 7, top left).

So to the extent that we are guilty of missing an obvious usability problem with the Critique code review tool, at least we are in good company.

Although this paper's contribution is largely quantitative, we observed the following qualitative experiences applying GenderMag at Google:

- The GenderMag method was new to session participants, so having an OSU GenderMag expert describe the method, its

[7] The reader might question how to reconcile this 81-83% discovery rate with the 35% discovery rate reported at the beginning of Section 5. The answer is that most users in our dataset never reached 1000 comments; this is evident in Figure 5, where the curves are smooth at the left of the figure, indicating many data points, but relatively rough at the right of it, indicating few data points.

[8] Accordingly, commentary notes that this feature of GitHub is "not that well-known". (https://haacked.com/archive/2019/06/03/suggested-changes/) and "for some reason a lot of reviewers are not familiar or bother" with the feature (https://news.ycombinator.com/item?id=38518473#38522993).

research basis, and a short demo of how to run a GenderMag session helped make our GenderMag sessions productive.

- A deep appreciation of the method and its potential impact really only became apparent once our sessions were underway, and we dived deep into a use case using the personas.
- The detailed analysis framework that GenderMag provides allowed us to identify actionable issues in a systematic way, which we could then tackle via design changes.
- Initially the method seemed to have a high cost in terms of time, and the outcomes potentially seemed a bit abstract to the workshop participants (due to the initial lack of familiarity with the method). However, by the end of the two-day design workshop, after seeing the full journey from identifying issues using GenderMag to seeing improved designs being sketched out and come to life, a number of workshop participants expressed that any initial reservations they may have had with the method were gone.
- Reflections after the workshops on what could be improved for future iterations surfaced reducing the overall time cost of the method. For example, as also suggested in prior work [14], they suggested potentially having smaller groups go through the detailed feature analysis, and instituting some time keeping so that the entire use case can be examined in the time set (since it can be easy to go deep on details).

Another observation of using GenderMag was that although many findings emerged, few findings were actually fixed (see Table 1), and the glib summary is that our fixes entailed just copying a button and adding some text. On one hand, the number of findings or fixes that GenderMag yields is not as important as whether the fixes actually make a tangible improvement. On the other hand, if we had implemented more fixes, perhaps we could have achieved even more tangible increases to discoverability and usability overall. But implementing even more fixes would have been challenging, for several reasons: additional user interface improvements can have diminishing returns; interfaces that look great in mockups can be difficult to implement in deployed software; and existing users of a system expect consistency from release to release. These challenges illustrate some practical limitations of design methods like GenderMag.

Finally, we believe that our quantitative analysis technique – applying survival analysis to feature usage, then disaggregated by gender – should be more widely used to find gender disparities in software usage. Doing so manually is a nearly impossible task, because in order to know what features work disproportionately poorly for some genders requires researchers to run GenderMag (or a more expensive technique, like a lab study) on every feature. Instead, we envision that a broader use of quantitative, logs-based techniques like the one we presented here can find the most underdiscovered features for some identity group, whether that be women or Latinx+ users, for instance. More intense and time-consuming methods, like GenderMag or its broader-scoped cousin InclusiveMag [19] which considers more facets of identity than just gender, could then be used as a follow-up to diagnose and remedy these statistically problematic features. Logs-based techniques like ours could complement attempts to automate applying GenderMag [6], and more broadly contribute to ongoing "quantitative personas" research [23, 27].

7 CONCLUSION

In this paper, we describe an application of the GenderMag design methodology, showing that it helped improve the usability of a feature called Suggest Edit. In particular, our quantitative evaluation showed that, prior to GenderMag, it took women more time to discover the feature than men. It also showed that, after redesigning with the help of GenderMag, we were able to eliminate this gender gap entirely. More broadly, we envision a future in which practitioners can quantitatively hone in on problematic software features at scale, a step on the journey towards more software that works for a more diverse set of users.

8 DATA AVAILABILITY

The raw data used in this paper is sensitive – especially gender data – and sharing it is not permitted.

ACKNOWLEDGEMENTS

Thanks to the users of Critique for their continued feedback and support. For their assistance on this research and early drafts of this paper this paper, thanks to Anita Sarma, Danny Berlin, Kristóf Molnár, Margaret Burnett, Ole Rehmsen, and Rayven Plaza.

REFERENCES

[1] Peter C Austin. 2017. A tutorial on multilevel survival analysis: methods, models and applications. *International Statistical Review* 85, 2 (2017), 185–203.

[2] J Martin Bland and Douglas G Altman. 1998. Survival probabilities (the Kaplan-Meier method). *Bmj* 317, 7172 (1998), 1572–1580.

[3] Margaret Burnett, Robin Counts, Ronette Lawrence, and Hannah Hanson. 2017. Gender HCI and Microsoft: Highlights from a longitudinal study. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 139–143.

[4] Margaret Burnett, Anicia Peters, Charles Hill, and Noha Elarief. 2016. Finding gender-inclusiveness software issues with GenderMag: A field investigation. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 2586–2598.

[5] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, and William Jernigan. 2016. GenderMag: A method for evaluating software's gender inclusiveness. *Interacting with Computers* 28, 6 (2016), 760–787.

[6] Amreeta Chatterjee, Mariam Guizani, Catherine Stevens, Jillian Emard, Mary Evelyn May, Margaret Burnett, and Iftekhar Ahmed. 2021. AID: An automated detector for gender-inclusivity bugs in OSS project pages. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1423–1435.

[7] Sally Jo Cunningham, Annika Hinze, and David M Nichols. 2016. Supporting gender-neutral digital library creation: A case study using the GenderMag Toolkit. In *Digital Libraries: Knowledge, Information, and Data in an Open Access Society: 18th International Conference on Asia-Pacific Digital Libraries, ICADL 2016, Tsukuba, Japan, December 7–9, 2016, Proceedings 18*. Springer, 45–50.

[8] John Fox and Sanford Weisberg. 2002. Cox proportional-hazards regression for survival data. *An R and S-PLUS companion to applied regression* 2002 (2002).

[9] Jeremy Freese and David Peterson. 2017. Replication in social science. *Annual Review of Sociology* 43 (2017), 147–165.

[10] Marko Gasparic, Andrea Janes, Francesco Ricci, Gail C Murphy, and Tural Gurbanov. 2017. A graphical user interface for presenting integrated development environment command recommendations: Design, evaluation, and implementation. *Information and Software Technology* 92 (2017), 236–255.

[11] Mathieu Goeminne and Tom Mens. 2015. Towards a survival analysis of database framework usage in Java projects. In *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 551–555.

[12] Tovi Grossman, George Fitzmaurice, and Ramtin Attar. 2009. A survey of software learnability: metrics, methodologies and guidelines. In *Proceedings of the sigchi conference on human factors in computing systems*. 649–658.

[13] Mariam Guizani, Lara Letaw, Margaret Burnett, and Anita Sarma. 2020. Gender inclusivity as a quality requirement: Practices and pitfalls. *IEEE Software* 37, 6 (2020), 7–11.

[14] Claudia Hilderbrand, Christopher Perdriau, Lara Letaw, Jillian Emard, Zoe Steine-Hanson, Margaret Burnett, and Anita Sarma. 2020. Engineering gender-inclusivity into software: ten teams' tales from the trenches. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 433–444.

[15] Charles G Hill, Maren Haag, Alannah Oleson, Chris Mendez, Nicola Marsden, Anita Sarma, and Margaret Burnett. 2017. Gender-inclusiveness personas vs. stereotyping: Can we have it both ways?. In *Proceedings of the 2017 chi conference on human factors in computing systems*. 6658–6671.

[16] Clayton Lewis and Cathleen Wharton. 1997. Cognitive walkthroughs. In *Handbook of human-computer interaction*. Elsevier, 717–732.

[17] Wei Li, Tovi Grossman, and George Fitzmaurice. 2012. GamiCAD: a gamified tutorial system for first time autocad users. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 103–112.

[18] Bin Lin, Gregorio Robles, and Alexander Serebrenik. 2017. Developer turnover in global, industrial open source projects: Insights from applying survival analysis. In *International Conference on Global Software Engineering (ICGSE)*. IEEE, 66–75.

[19] Christopher Mendez, Lara Letaw, Margaret Burnett, Simone Stumpf, Anita Sarma, and Claudia Hilderbrand. 2019. From GenderMag to InclusiveMag: An inclusive design meta-method. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 97–106.

[20] Emerson Murphy-Hill, Jillian Dicker, Amber Horvath, Maggie Morrow Hodges, Carolyn D Egelman, Laurie R Weingart, Ciera Jaspan, Collin Green, and Nina Chen. 2023. Systemic Gender Inequities in Who Reviews Code. *Proceedings of the ACM on Human-Computer Interaction* 7, CSCW1 (2023), 1–59.

[21] Emerson Murphy-Hill, Da Young Lee, Gail C Murphy, and Joanna McGrenere. 2015. How do users discover new tools in software development and beyond? *Computer Supported Cooperative Work (CSCW)* 24 (2015), 389–422.

[22] Emerson Murphy-Hill, Edward K Smith, Caitlin Sadowski, Ciera Jaspan, Collin Winter, Matthew Jorde, Andrea Knight, Andrew Trenk, and Steve Gross. 2019. Do developers discover new tools on the toilet?. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 465–475.

[23] Lene Nielsen, Joni Salminen, Bernard J Jansen, Jisun An, Haewoon Kwak, and Soon-Gyo Jung. 2019. Automatic persona generation for online content creators: Conceptual rationale and a research agenda. *Personas-User focused design* (2019), 135–160.

[24] Hema Susmita Padala, Christopher Mendez, Felipe Fronchetti, Igor Steinmacher, Zoe Steine-Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Simpson, Margaret Burnett, et al. 2020. How gender-biased tools shape newcomer experiences in OSS projects. *IEEE Transactions on Software Engineering* 48, 1 (2020), 241–259.

[25] John Pruitt and Jonathan Grudin. 2003. Personas: practice and theory. In *Proceedings of the 2003 conference on Designing for user experiences*. 1–15.

[26] Huilian Sophie Qiu, Alexander Nolte, Anita Brown, Alexander Serebrenik, and Bogdan Vasilescu. 2019. Going farther together: The impact of social capital on sustained participation in open source. In *International Conference on Software Engineering (ICSE)*. IEEE, 688–699.

[27] Joni Salminen, Kathleen Guan, Soon-gyo Jung, Shammur A Chowdhury, and Bernard J Jansen. 2020. A literature review of quantitative persona creation. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[28] Arun Shekhar and Nicola Marsden. 2018. Cognitive Walkthrough of a learning management system with gendered personas. In *Proceedings of the 4th Conference on Gender & IT*. 191–198.

[29] Simone Stumpf, Anicia Peters, Shaowen Bardzell, Margaret Burnett, Daniela Busse, Jessica Cauchard, Elizabeth Churchill, et al. 2020. Gender-inclusive HCI research and design: A conceptual review. *Foundations and Trends® in Human–Computer Interaction* 13, 1 (2020), 1–69.

[30] Mark D Syer, Meiyappan Nagappan, Bram Adams, and Ahmed E Hassan. 2014. Replicating and re-evaluating the theory of relative defect-proneness. *IEEE Transactions on Software Engineering* 41, 2 (2014), 176–197.

[31] Marat Valiev, Bogdan Vasilescu, and James Herbsleb. 2018. Ecosystem-level determinants of sustained activity in open-source projects: A case study of the PyPI ecosystem. In *Joint Meeting on the Foundations of Software Engineering (ESEC/FSE)*. 644–655.

[32] Mihaela Vorvoreanu, Lingyi Zhang, Yun-Han Huang, Claudia Hilderbrand, Zoe Steine-Hanson, and Margaret Burnett. 2019. From gender biases to gender-inclusive design: An empirical investigation. In *Proceedings of the 2019 CHI Conference on human factors in computing systems*. 1–14.