Climate Coach: A Dashboard for Open-Source Maintainers to Overview Community Dynamics

Huilian Sophie Qiu
sophie.qiu@kellogg.northwestern.edu
Northwestern University; Carnegie Mellon University
Evanston, IL, USA

Anna Lieb
al117@wellesley.edu
Wellesley College
Wellesley, MA, USA

Jennifer Chou
jcchou@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Megan Carneal
megancarneal@gmail.com
The University of Alabama
Tuscaloosa, AL, USA

Jasmine Mok
jasminem@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Emily Amspoker
eamspoke@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Bogdan Vasilescu
vasilescu@cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Laura Dabbish
dabbish@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT

Open-source software projects have become an integral part of our daily life, supporting virtually every software we use today. Since open-source software forms the digital infrastructure, maintaining them is of utmost importance. We present Climate Coach, a dashboard that helps open-source project maintainers monitor the health of their community in terms of team climate and inclusion. Through a literature review and an exploratory survey (N=18), we identified important signals that can reflect a project's health, and display them on a dashboard. We evaluated and refined our dashboard through two rounds of think-aloud studies (N=19). We then conducted a two-week longitudinal diary study (N=10) to test the usefulness of our dashboard. We found that displaying signals that are related to a project's inclusion help improve maintainers' management strategies.

ACM Reference Format:
Huilian Sophie Qiu, Anna Lieb, Jennifer Chou, Megan Carneal, Jasmine Mok, Emily Amspoker, Bogdan Vasilescu, and Laura Dabbish. 2023. Climate Coach: A Dashboard for Open-Source Maintainers to Overview Community Dynamics. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3544548.3581317

1 INTRODUCTION

While by and large, our society has come to recognize the immense value and importance of open-source software for our digital economy [27], maintaining all this digital infrastructure remains challenging [43]. With much open-source software being developed and maintained by volunteers [14, 65, 66], a core issue affecting the sustainability of the whole ecosystem remains to be attracting and retaining contributors to different projects. Many important, heavily downloaded open-source projects are maintained by only one or two developers [16] and sometimes by no one at all [4]. Turnover rates are generally high in open-source projects [28, 51]. Moreover, there are many documented socio-technical barriers that newcomers face when trying to join open-source projects [47, 77], often disproportionally affecting women and members of underrepresented groups [40, 85, 88].

Researchers have made considerable progress in the last decade towards understanding the factors that affect the health and sustainability of open-source projects, *e.g.*, see Franco-Bedoya et al. [31], Linåker et al. [43], and Trinkenreich et al. [88] for recent surveys. This knowledge is only beginning to make its way back into practice. In addition to technical aspects, such as code quality and development process, and project governance aspects, such as licensing, social (community) aspects is an indispensable dimension of open source health. Nevertheless, there is a surprising scarcity of evidence-based interventions for improving open-source community health in terms of team climate and inclusion. Although some code hosting platforms begin to provide some community-oriented design elements, such as GitHub's checklists of items associated with promoting inclusion and community health, *e.g.*, contributing guidelines and codes of conduct, only two academic studies designed and evaluated community health interventions: Steinmacher et al. [78] designed a portal helping newcomers to navigate an open source project, and Guizani et al. [35] designed a dashboard assisting maintainers to overview the joining, activity, and retention trends of the newcomers to their projects.

In this paper, we take another step in the direction of open-source community health interventions. Grounded in the literature and interviews with open-source maintainers, we start by identifying factors and measures indicative of community health that do not currently have associated signals in the GitHub UI or other

CHI '23, April 23–28, 2023, Hamburg, Germany
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3581317

dedicated monitoring infrastructure, such as indicators of responsiveness to issues [60], pushback in code reviews [26], and toxicity in pull request and issue discussions [64]. Next, we iteratively design and evaluate a dashboard, Climate Coach, that tracks these indicators for a given project over time and in comparison to a group of 'peer' projects. Results from a diary study with project maintainers show that our dashboard can increase the maintainers' confidence in supporting community health. Compared to prior studies [35, 78], our work goes beyond newcomers and is more broadly focused on project climate, but can be seen as complementary to those efforts on exploring indicators of open-source community health [33, 35].

Our work also has several broad contributions to the Human-Computer Interaction (HCI) community. **An intervention that integrates prior literature**: our work builds on a vast body of literature that studied how to improve a project's climate and inclusion and implements various methods or actions suggested in those works. We turned many of the practices or suggestions developed from empirical studies into a usable intervention. **A product ready to use**: our dashboard design has been refined by two rounds of interviews with active open-source maintainers and a two-week field study. Our source code is publicly available.[1] **Proof of idea**: the positive feedback we have received from maintainers on the usefulness of our dashboard shows that the general idea of turning important yet hidden metrics into plain, observable, and quantifiable signals can help users better assess their projects' status and make informed decisions. **Transferrable results**: some of our findings are not exclusively applicable to the open-source context and can be adapted to other similar team settings such as remote collaboration or volunteer communities.

2 RELATED WORK

2.1 Healthy Open-Source Communities

With open-source software becoming ubiquitous and powering applications in virtually every domain, much of the research attention has shifted from understanding how and why this mode of production functions [52] and what motivates people to contribute to it [39], to understanding what are the risk factors impacting the health of open source and how to sustain this digital infrastructure on which so much of our society relies [16, 27, 33, 90]. Many dimensions of project health have been identified as important, ranging from organizational and legal (*e.g.*, what are appropriate governance structures [44, 56] and licenses [48, 84]), through technical (*e.g.*, how to ensure code quality in a rapidly-paced, distributed software development setting [20, 76]), to social (*e.g.*, how to attract and retain contributors [60, 79] and how to maintain healthy conversations [49, 64]).

A key challenge related to an open-source project's social dimension is attracting and retaining contributors. Much of open source is developed and maintained by volunteers [14, 65, 66], who typically have a choice of where in the ecosystem to spend their efforts (which projects to join, which tasks to work on, *etc.*) and how long to stick around. Coupled with a constant need for work to maintain and evolve open-source software systems (*e.g.*, fixing bugs and security vulnerabilities, developing new features, improving documentation) and generally high rates of turnover among all contributors to open source [28, 51, 70], the volunteer-based community makes it hard to sustain a steady stream of contributors to one's open-source project.

The literature has identified a wealth of factors that could impact a project's ability to attract and retain contributors, affecting all stages of the contribution process, ranging from choosing a project to join [60] and overcoming initial barriers to entry [82], to ensuring long-term sustained participation and engagement [61]. Many of these factors are cultural. Whether the project is perceived as open to new contributors and whether it has enough scaffolding in place to facilitate their onboarding [60], how the project acknowledges contributions [13, 95], whether the maintainers are responsive to requests [26, 62] and constructive and reasonable to other contributors in their feedback [80, 82], whether the tone of project-related discussions is perceived as polite or, on the contrary, toxic [29, 30, 49, 64], and whether a code of conduct is in place [42, 73, 74] are all seen as contributing to creating an inclusive environment and a healthy community. In addition, these issues are known to disproportionately affect women and other groups that are severely underrepresented in open source [53, 59, 85], which further reduces the size of a project's potential contributor pool; it also has broader negative consequences beyond open source, as contributing to open source is for many a launching ramp for professional careers in the technology world [32, 45].

2.2 Transparency and Signaling

Our discussion above leaves implicit the impression formation process through which the different perceptions of project climate attributes are formed. In fact, social coding platforms rely heavily on transparency and signaling to facilitate impression formation [22, 46, 89]. On GitHub, many signals (visible cues) are available by default as part of the UI for all projects hosted on the platform, *e.g.*, the number of stars a repository has received, the number of commits recorded in its history, or the number of followers an individual user has accumulated. Other signals, *e.g.*, repository badges embedded in a project's top-level README file [89], still can be defined and customized by project maintainers to communicate attributes of interest including code quality, adherence to testing and dependency management best practices. Prior work has shown that all these various signals play a role in a diversity of decisions users on the platform make, including which repositories to watch [71], which to trust [89], which to contribute to [60], which pull request contributions to accept [46], which developers to follow [7, 41], and even which developers to recruit for positions in the offline world [11, 45].

At the same time, not all attributes indicative of a healthy project climate and community have dedicated signals or monitoring tools. This leaves maintainers without a direct way to monitor the health of their open-source communities and, if needed, intervene. However, as we argue in this paper, there is an opportunity to further leverage the high level of transparency afforded by the GitHub platform, which recorded and made public available many dimensions of activities and communication histories, to design new signals indicative of open-source community health.

[1]https://doi.org/10.5281/zenodo.7592079

2.3 Measurement and Dashboards

Unsurprisingly, mining data from software repositories to measure and understand the activity in open source is an idea almost as old as the domain itself. By now, researchers have proposed a plethora of measures of project activity, success, or health in terms of project quality [2, 3, 16, 18, 20, 21, 28, 52, 87, 90, 92], including measures of code quality, popularity, team size, team productivity, turnover, contribution inequality, and risk of becoming abandoned, just to name a few.

Researchers have also proposed tools to extract and visualize information from open-source repositories, including various dashboards designed to help with project management tasks. Typically, these have focused on technical aspects, *e.g.*, visualizing distributions and trends in basic activity metrics such as the number of commits and commit authors across projects in an ecosystem [57, 94], managing package dependencies and possible conflicts arising from dependency version upgrades [15], and visualizing statistics about reported issues [37].

There are also a few notable examples of efforts focused on social aspects such as identifying expertise [1], visualizing the structure of the various socio-technical networks that form between developers, communications, and software artifacts [54, 69, 86, 97], visualizing collaboration patterns [58], visualizing trends in demographic diversity attributes such as gender and geographic location [67, 96], and raising awareness of team members' current activities when working on a shared code base [5]. More generally, there are now multiple mature data analytics toolsets for software repository data [19, 24, 25, 75], as well as efforts to standardize the relevant measures indicative of open source health as part of the CHAOSS project [33, 34].

In contrast, there has been very little work to design dashboards and monitoring infrastructure explicitly for health in open source and sustainability indicators, particularly along the project climate dimension related to attracting and retaining contributors. As exceptions, we note, first, an earlier work by Steinmacher et al. [78] of a portal for newcomers that helps demistify the joining process. Second, Guizani et al. [35] designed a dashboard for maintainers to monitor statistics about the joining, activity, and retention trends of the newcomers to their projects. Finally, we note work by Goggins et al. [33] to "implement CHAOSS metrics and present them in ways that enable maintainers, contributors, and other stakeholders to draw inferences about the relative health and sustainability of their projects."

Our current work builds most directly on insights from the latter two. Similarly to both Guizani et al. [35] and Goggins et al. [33], the target audience for our dashboard is project maintainers and community managers interested in monitoring open-source community health. Similarly to Goggins et al. [33], we also include a project comparison element in our design, to allow maintainers to benchmark their community's metrics against a subset of their peers. However, unlike either of them, we focus our dashboard on measures of project climate indicative of a healthy, inclusive culture, including subtle indicators of pushback in code reviews and toxicity in issue discussions, that currently lack associated signals in the GitHub UI or dedicated repository badges (see details below). Moreover, we focus this paper on the design of the dashboard and explore, using an iterative user-centered design process, different media through which maintainers can interact with the dashboard, different sets of metrics to track, and many other design decisions. Finally, we report on a diary study to evaluate the dashboard in practice. Overall, we see our work as complementary to that of Guizani et al. [35] (focused on newcomers rather than project climate more broadly) and Goggins et al. [33] (focused on standardized metrics and measurement infrastructure rather than dashboard design). We expect future work in this space to combine elements of all three.

Beyond open source health monitoring, dashboards have long been used for team management in other domains. We highlight in particular the work of Samrose et al. [68], which inspired our study design. Samrose et al. [68] created MeetingCoach, a wireframe dashboard to facilitate more inclusive online meetings. The authors first conducted an initial survey, from which they collected feedback on what features can help create a more inclusive meeting, such as speaking turns. Then they created a wireframe and iterated on the design with interviews and think-aloud studies with in-situ meetings. Finally, they showed that the dashboard improves meeting attendees' awareness of meeting dynamics that have implications for inclusion.

3 METHODS

This section provides an overview of our three-phase study design. The process is illustrated in Figure 1. Following the guidance by Bjögvinsson et al. [6] on participatory design and the study on the MeetingCoach dashboard by Samrose et al. [68], our design process is iterative and collaborative. We invite active open-source maintainers to participate starting at a very early stage of the design process, and we attempt to include a diverse set of maintainers to gain different perspectives.

Overall, Phase 1 consists of email interviews with active maintainers to **collect signals** that can help improve an open-source project's climate and inclusion. In Phase 2, we use the signals from Phase 1 to **design our dashboard** and evaluate the usability with active open-source maintainers. Phase 3 is a field study to **evaluate the dashboard**: we conducted a two-week diary study with maintainers to see its effectiveness.

3.1 Phase 1: Collect Signals - Email Interviews

During Phase 1, we conducted email interviews with maintainers to learn about what signals would help them manage their project's health in terms of welcoming contributors and community inclusion. Our goal was to determine what signals could be included in our dashboard to assist maintainers in monitoring their project's health. Because of this, we were interested in what strategies maintainers employed to manage newcomers. Onboarding newcomers is a significant burden on maintainers but also an important source of community growth and influences how inclusive and welcoming a project seems to outsiders. The outcome of this phase was a list of signals to include in our Climate Coach dashboard (Table 1). Given the vast literature on open-source management, maintainers' ongoing experience allowed us to identify and determine the most critical signals to include in our dashboard. Below we present the method of how we collected signals for our dashboard.

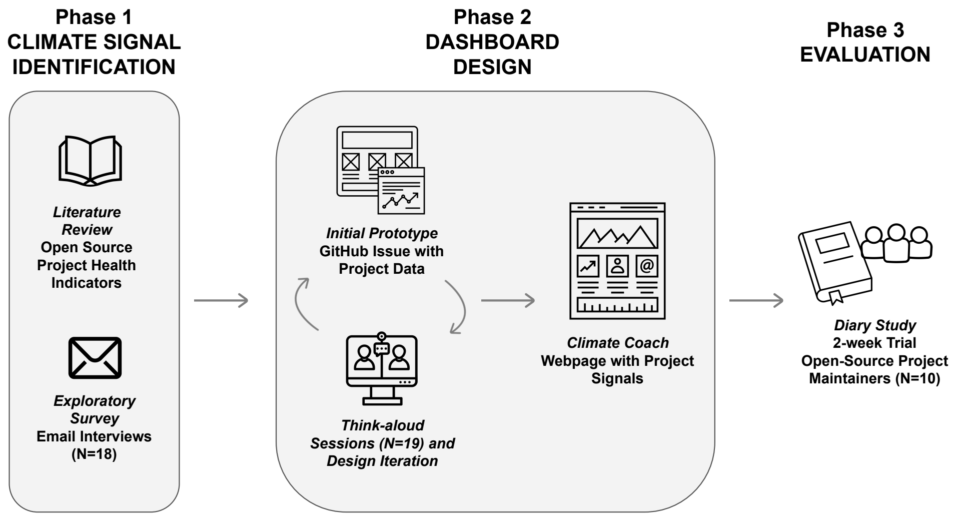

Phase 1
CLIMATE SIGNAL IDENTIFICATION
Literature Review
Open Source Project Health Indicators
Exploratory Survey
Email Interviews (N=18)
Phase 2
DASHBOARD DESIGN
Initial Prototype
GitHub Issue with Project Data
Think-aloud Sessions (N=19) and Design Iteration
Climate Coach
Webpage with Project Signals
Phase 3
EVALUATION
Diary Study
2-week Trial
Open-Source Project Maintainers (N=10)

Figure 1: Three-phase study process

Protocol. The email interview was primarily unstructured. It started with a single question: *what their project thinks about new contributors*. Instead of directly asking what the maintainers have done to attract new contributors, we decided to ask this broad question to avoid leading the project owners to provide socially desirable answers about project diversity and inclusion. In addition to their initial responses to our prompt, we engaged in follow-up conversations with many of the maintainers so that they could fully express their thoughts in response to our query.

Recruitment. We used the GitHub API to identify 100 projects that had commits in the past week and owners that displayed their emails on their profile pages. Projects with fewer than three people were excluded because small projects were more likely to be personal or private projects rather than open-sourced ones and less likely to have dealt with newcomers or contributions from nonmembers.

We tried to collect a sample that consists of projects of various sizes and whether they have at least one woman contributor. We consider projects with at least one woman or non-binary contributor as gender-diverse. We had to choose this loose definition of gender-diverse projects lest our pool of potential participants would be too small. Because men contributors take up more than 90% of the open-source population [63], the number of projects containing women or non-binary people is already very minimum. Our sample was further limited because not all maintainers disclose their contact information on GitHub.

To identify gender-diverse projects, we first used the computational tool Namsor [12, 63] to infer contributors' genders automatically based on their names. This inference served only as a guideline to point us to projects that probably have women contributors. We then manually verified if there were indeed women or non-binary contributors to the project. When trying to detect whether there are women or non-binary contributors, we acknowledge the limitation in our method that we mostly used their namesand profile pictures as an approximation. Knowing that it is a relatively unreliable method than asking the contributors to self-identify their genders, we only consider those with strong signals, *e.g.*, commonly used female names or clearly labeled gender pronouns, as contributors of a gender other than men. We further discuss the limitation in Section 7.

We sent out emails to 100 project owners and received 18 responses. Among them, ten projects did not have any women or non-binary contributors, and the other 8 had at least one. Unfortunately, there were not enough women or non-binary maintainers for us to contact, so the gender distribution of our interviewees was heavily skewed toward men. The projects' sizes we collected ranged from 4 to 5000+. We refer to participants in our email interviews as R0Px. The breakdown of the projects' characteristics is shown in Table 1 in our supplemental material.[1]

Data analysis. Two of the authors conducted a thematic analysis on the responses we received from maintainers [8]. As a validation of our literature review, we focused on the themes that were present in prior studies while paying attention to new themes. We first identified instances of different themes in the first ten responses. For each response analyzed, we identified owners' attitudes towards new contributors and actions they described taking to handle new contributors. Based on the themes we identified from our first round of open coding, we developed a set of initial codes and then continued open coding the rest of the responses, comparing each response with previously examined ones, adding new codes when a new theme emerged, and grouping codes to form higher level categories. When possible, we assigned codes to categories we identified from the literature. We repeatedly discussed the codes and categories in a highly collaborative and iterative process. We present the results in Section 4.

3.2 Phase 2: Design Dashboard - Think-aloud Interviews

The goal of Phase 2 was to use the signals we collected in Phase 1 to develop our dashboard and conduct usability interviews with maintainers. We calculated and visualized signals collected in the previous phase to put on our dashboard. After we finished our initial prototype design (shown in Section 1 in our supplemental material),[1] we started conducting think-aloud interviews with open-source maintainers to test its usability. The outcome of this phase was a refined dashboard that we used for Phase 3, a two-week diary study with open-source maintainers to further test the usability and its effectiveness in helping maintainers manage their project climate.

We used PyGithub[2] to mine GitHub data for our dashboard. Given a project's slug (*owner/repo_name*), we wrote a Python program that could automatically pull a project's data for generating the dashboard, including issues and PRs, along with all their comments. After removing bots [23] and their posts, we performed data aggregation, analysis, and visualization. We present the details of our design in Section 4.

After we produced our initial dashboard prototype, we conducted two rounds of detailed semi-structured interviews and think-aloud studies with active GitHub maintainers to test the usability of our dashboard and guide later stages of development. We conducted two rounds of interviews, modifying the prototype design after the first round of interviews and performing more interviews to test the updated design. We also used this opportunity to understand better how maintainers assess their community health and approach issues related to inclusion.

Protocol. Our interview protocol consists of two major parts. During the first part, we asked participants questions regarding their project community, their perception of the health of their communities, and their methods of managing their communities. The second part adopted the think-aloud approach to understanding how participants used the dashboard. Before each interview, we generated an individualized dashboard for the participant based on data from their repository. We asked the participants to browse through the dashboard. If they had any questions during the think-aloud, we answered them after they finished browsing the dashboard. After participants finished browsing the dashboard, we asked several follow-up questions regarding what signals were important, unnecessary, or missing. In the end, we ask them several demographic questions such as gender, age, and race. Our interview protocol is presented in our supplemental material.[1]

Recruitment. To recruit participants, we searched on GitHub for a stratified range of stars, which serves as an approximation of a project's popularity or size. We also filtered projects based on whether they had recent activity.

We identified GitHub projects with recent activities and contacted the project maintainers, *i.e.*, owner of the project or the top two contributors of projects owned by organizations, if they provided emails or Twitter handles on their GitHub page. Although we strived to recruit women or non-binary maintainers, we were not very successful due to the low representation of women and non-binary people among maintainers. After we interviewed ten men maintainers, we paused the interview process and made changes to the dashboard. Therefore, we call the first ten interviews as the first round and refer to each of them as R1Px. After we redesigned our dashboard, we conducted the second round of interviews and made new changes when new feedback emerged so that we could test new designs immediately. We talked to a total of 9 maintainers (including one woman maintainer). We refer to each of them as R2Px.

Our participants' projects had from 11 to 20.6K stars. The team sizes range from 8 to 100+. Five projects have at least one woman or non-binary contributors. The breakdown of the think-aloud interview participants is in Table 2 in our supplemental material.[1]

Data analysis. Our coding process aimed to identify two major categories: maintainers' perception of community health and feedback on our dashboard. We first performed open coding on interview transcripts. Two of the authors first coded two interviews independently. They then met to discuss their codes through a constant comparison method: they consolidated codes into a shared set of codes by combining overlapping codes or developing new codes. The two authors independently coded another four interview transcripts with the preliminary code book before convening again to discuss the generated codes. After the two authors coded the rest of the interviews, they met again to discuss all the codes and coded paragraphs. We continued conducting interviews while coding the transcripts and concluded the first round of interviews when we reached theoretical saturation, *i.e.*, no new themes emerged from new interviews. Then the two authors conducted axial coding on the full set of codes: we considered the relationship among the codes and assigned them to one of the two major categories or created a new category. We followed the same coding procedure for our second round of think-aloud interviews. The codebook is available in our supplemental material.[1]

3.3 Phase 3: Evaluate Dashboard - Diary Study

After incorporating changes to the dashboard based on feedback from the think-aloud interviews, we evaluated the final dashboard design in a two-week diary study [55].

Protocol. Our diary study lasted two weeks and followed the structure described in Figure 2, including an initial survey, onboarding session, weekly survey, and exit survey. Participants were compensated $50. Below we describe each of the study components in more detail.

1) Initial Survey + Onboarding Session (30 minutes)

Initial survey (20 minutes). We provided an initial survey for participants to fill out via a Google form to gather background information about their project and their perceptions of community health. We also provided participants with the consent form and information about the study structure. In the survey, we asked for background information about the maintainer's identity, habits, and project dynamics. More importantly, the survey also asked questions about *maintainers' workflow*, *their perception of their community's health*, and *projects they want to be compared with*.

When asking about maintainers' workflow, we asked about whether they were seeking new contributors, the importance of increasing demographic or technical diversity, and how confident they were in managing their community. We also asked them to rate the priorities of several management actions, including "fast response time to issues," "fast response time to PR," "creating a

[2]https://github.com/PyGithub/PyGithub

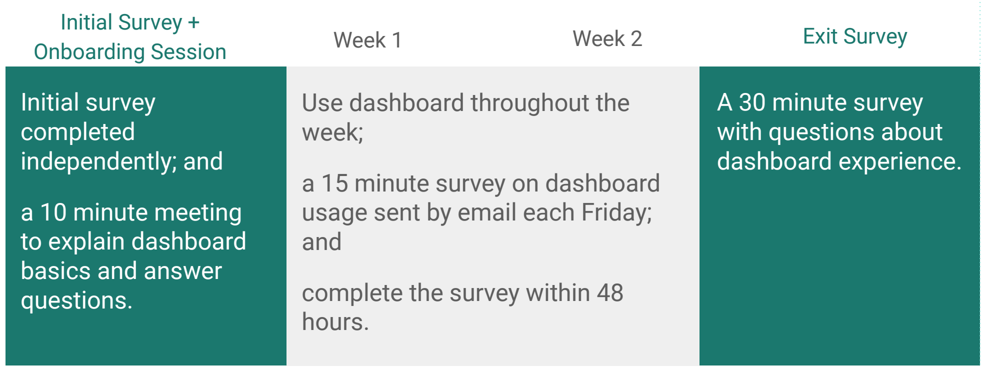

Initial Survey + Onboarding Session
Week 1
Week 2
Exit Survey
Initial survey completed independently; and
a 10 minute meeting to explain dashboard basics and answer questions.
Use dashboard throughout the week;
a 15 minute survey on dashboard usage sent by email each Friday; and
complete the survey within 48 hours.
A 30 minute survey with questions about dashboard experience.

Figure 2: Diary Study Logistics

welcoming environment," "attracting new contributors," and "attracting a diverse group of contributors." We then asked them how often they responded to issues and PRs each week and their goals for their project community.

Then, we asked participants about their understanding of community health with three open-ended questions: How would you describe your project's community health? How would you define diversity in open-source software? How would you define inclusion in open-source software?

Our dashboard compares the project with similar projects to provide maintainers a context and reference of how well they are doing. To make our comparison more relevant to the maintainers, we asked participants to enter projects they wanted to compare with. At the end of the survey, we asked them to sign up for a time slot for a Zoom call for an onboarding session, as well as several demographic questions such as gender, age, and race.

Onboarding session - Zoom call (10 minutes). During the Zoom call, we explained the logistics of the study and weekly survey. Then we showed them the dashboard and ensured they understood the basic setup and answered any questions.

2) Weekly usage (30 minutes each week $\times$ 2 = 1 hour total)

Participants could freely use the Climate Coach dashboard as little or as much as they wanted during the two-week study period. Each Friday, we sent an email asking participants to complete a brief weekly survey about how they used the dashboard that week. The survey itself took about 15 minutes. Participants were asked to complete the survey within 48 hours.

The weekly survey consisted of two parts, *maintainers activity* and *dashboard engagement*. The questions in the maintainers' activity portion included the types of contributions they receive, the amount of time they spend on maintaining, and the tone of conversations in their community. In the dashboard engagement portion, we asked participants questions regarding the usefulness of the dashboard, such as how often they checked the dashboard, which parts were most useful, which tips were more helpful, and how the signals were.

3) Exit Survey (30 minutes)

After two weeks, we sent participants an exit survey to get feedback on the dashboard and compare responses from the initial survey. We repeated questions from *maintainers' workflow* and *perception of their community's health* in the initial survey and added questions regarding *the usefulness of our dashboard.* We asked them to rate their level of agreement with a list of statements regarding whether the dashboard is useful for them and other maintainers. To test if our dashboard had any effect on their management strategies, we asked if they made any changes after viewing our dashboard. Lastly, we asked them how likely they were to continue to use this dashboard after the study ended. The survey concluded with an open-ended question for feedback on the dashboard.

We provided 5-point Likert scales for participants to measure their level of agreement with statements regarding their perception of their community's health and the usefulness of our dashboard, with 5 being "strongly agree" and 1 being "strongly disagree." To test if they were paying attention to the statement rather than clicking "strongly agree" or "agree" for all statements, we reverse-coded some of the statements as an attention check. When analyzing responses to these statements, we first reversed the responses, *i.e.*, "strongly agree" as "strongly disagree."

We include the diary study protocol in our supplemental material.[1]

Recruitment. For the diary study, we explicitly recruited maintainers from big and active projects. From the two prior interviews, we learned that big projects could benefit more from our dashboard (R2P1) because there are a number of things to keep track of that can exceed maintainers' capability. Moreover, since our diary study's survey frequency is weekly, less active projects will not have generated sufficient activities to appear on the dashboard. Therefore, for the diary study, we searched on GitHub for projects with at least 1K stars, followers, or 100 to 200 forks.[3] From the search results, we picked the projects with activities (issues or PRs) within the last week and with at least ten contributors. Similar to previous interviews, we contacted only the maintainers who left their emails, or Twitter handles on their profile pages.

In the end, we recruited 10 participants for our diary study. Two of them reached out to us after seeing our advertisement on Twitter. The rest of them accepted our email invitation. We sent out 128 emails, and 8 of them were accepted. The summary of all the participants' projects is shown in Table 3 in our supplemental material.[1] We refer to each of the participants as R3Px.

[3]https://docs.github.com/en/search-github/searching-on-github/searching-for-repositories

Data Analysis. We analyzed the responses to the 2 weekly surveys and the exit survey, using participants as the unit of analysis. One of the researchers performed open coding on open-ended questions in the surveys. We used affinity diagrammed codes generated from the open-ended responses to identify themes in participants' uses of and reactions to the dashboard.

4 DESIGN

This section presents the result from Phase 1, how they informed our design, and the think-aloud interview results from Phase 2.

4.1 Signals Extracted From Phase 1

From the email interview with maintainers, we compiled a set of strategies they have used when managing their projects. The overview of the strategies and how they link to the literature is provided in Table 1. Below we describe how these signals were grounded in email interviews and literature.

Team growth. Overall, the projects in our sample welcomed new contributors. For example, R0P5 expressed that *"one of [their] objectives is to convert more users to contributors to make the project more open and sustainable."* However, at the same time, they admitted that new contributors imposed a cost in terms of the effort required to manage contributions and socialize them. As one owner concluded, *"I welcome newcomers but fear them,"* because while they make contributions, more often than not, they also *"break the architectural vision or have bad coding practices"* (R0P1). Some projects noted that they did not actively search for new contributors, but some *"still arrive spontaneously"* (R0P12). Our dashboard uses `the number of active and new contributors` to reflect the growth of the projects.

In addition to growing a team, retaining current contributors is equally important to a project's sustainability. Some maintainers publicly recognize newcomers' efforts. For example, some maintainers put newcomers' names on a contributor list in the README (R0P1). Some invite contributors to become maintainers of the project and recognize their contributions (R0P5). We display on our dashboard `the list of active and new contributors` to the project to help maintainers recognize their efforts.

Social capital. Social capital is a concept in Sociology that measures the benefits one can gain from their social networks [10], either through strong connections, *e.g.*, long-time collaborators, or through connections to diverse groups. Previously, Qiu et al. [61] studied contributors' sustained participation on GitHub and found that high social capital is associated with contributors' prolonged engagement [61]. Therefore, we include measurements that can approximate a project's social capital: `the number of recurring contributors` and `the average months of contributors' tenure` for the strength of connections among contributors , and `the number of new contributors` to approximate connections to new groups.

Responsiveness. Our email interviews confirmed the importance of being responsive to contributors. Some maintainers pointed out that *fast reply* is a vital signal because ignoring contributions (even bad ones) may create ill will (R0P1), and contributors may *"feel spurned"* (R0P2). This observation echoes what Egelman et al. [26] described as "pushback": a delay in a code review that can cause negative feelings among contributors. We use the `average close time` as the signal on our dashboard to reflect fast replies. The dashboard also points maintainers to `the conversations (issues or PRs) that have been opened for the longest time`.

Some owners told us they tried to signal their accessibility and try to resolve issues or PRs within a specific period of time (R0P1, R0P2, R0P11). R0P2 told us he changed his profile status to be *"Merging your PR"* (R0P2). Our dashboard uses `the number of issues or PRs closed` and `the number of issues or PRs still open` to reflect how quickly maintainers conclude issue discussions or code reviews in PRs.

Some maintainers noted that they *"respect new contributors' bandwidth and often help them to refine contributions collaboratively"* (R0P4) by commenting back and forth on a design in a GitHub issue (R0P1), which is reflected by the signal `the number of comments` and `the number of conversations (issues or PRs) closed with 0 comments` in our dashboard.

However, too many comments can give contributors a feeling that maintainers are too picky or even unwilling to merge their contributions [26]. We try to help maintainers eliminate providing contributors such impressions by listing open issues and PRs with `the most comments`.

Conversation tone. Some maintainers mentioned that they try to show friendliness to newcomers, encourage contributions (R0P15), and signal inclusiveness (R0P4). They hope that the users of their libraries will feel welcome to contribute to them (R0P10). Some maintainers keep a Code of Conduct so that *"potential contributors have the feeling of a safety net"* (R0P10). We use Google's Perspective API[4] to measure comments' `toxicity score` and `identity attack score` (both in the range of [0,1]) to reflect a project's language inclusiveness. In addition, we provide links to the conversations with scores higher than a threshold (we set it to 0.7).

Onboarding material. Another way that maintainers welcome newcomers is to provide a beginner's guide or relevant documentation. Some of the actions they took to welcome newcomers include providing onboarding materials *"to show them the entire journey"* (R0P6). Some mentioned using a contributing guideline and issue tags (R0P1). Nevertheless, they also mentioned that using the "newcomer-friendly" tag was not practical because many of the issues were not newcomer-friendly (R0P1). Some maintainers recognized the importance of documentation but also admitted that their testing process was not well documented, which may scare away potential newcomers (R0P10). However, we did not include these in our dashboard because GitHub's Insight page consists of a checklist of all these recommended documentation.

Contribution process management. Maintainers varied in their internal coordination processes or methods to manage teams, and these activities influenced how they, in turn, tried to help newcomers. Some tried to use continuous integration (CI) tools to automate the process and save maintainers' time (R0P1 and R0P11). They tried to speed up the process by having bots check if the submission had passed CI before notifying owners to review. However, at the same time, they also admitted that using an *"CI can introduce too many rules and conventions newcomers need to learn,*

[4]https://perspectiveapi.com

Table 1: Phase 1 results: signals and their references.

Category	Strategy	Reference
Team management	Team growth	Email
	Recognize contribution	
Social capital	Bonding social capital	[61]
	Bridging social capital	
Responsiveness	Fast response	[26], Email
	Provide help	[36, 80, 82], Email
Conversation tone	Toxic conversation	[29, 50, 62, 64, 83], Email
Gamification	Compare with peers	[33]
Contributors' negative feelings	Avoid pushback	[26, 62]
	Avoid toxic language	

which can be discouraging" (R0P11). Our dashboard did not include any signals to reflect the use of CI or similar tools either because it is currently difficult to automatically detect using the GitHub API.

4.2 Creating the Dashboard

Next, we present how we design and create the dashboard.[5] Since GitHub maintainers have been heavily involved throughout the study and we refine our design iteratively based on their feedback, we present the entire design of our final, complete dashboard and note the parts informed by think-aloud findings. This paper includes screenshots of our final design. Our supplemental material[1] contains our first design.

4.2.1 Types of Signals.

Summarized signals: Repository's basic statistics in the recent past. The section, `Basic Stats`, displays signals from the Community and Responsiveness categories shown in Table 1 as numeric values. It includes `the number of new contributors` and `the number of active contributors` in the past month. For responsiveness, the dashboard reports `the number of issues and PRs closed` in the past month and `the average close time of issues and PRs`, as well as `the number of open issues and PRs` and the `average time they have been open`.

Temporal signals: Trends in the past period of time. This dashboard provides line charts of the trends of signals shown in the `Basic Stats` section as a context of how their projects have developed, for example, in the past month or the past half a year.

Indicative signals: Conversation tone analysis. Inspired by a study by Raman et al. [64] and Qiu et al. [62], we added a signal for conversation tone, including the number of potentially problematic conversations and their links, identified using the Perspective API developed by Google. Before the 2nd round of interviews, we also added signals that can help identify contributors' negative feelings: excessive rounds of reviews and long shepherding time, *i.e.*, the time that "the author spent actively viewing, responding to reviewer comments, or working on the selected CR, between requesting the code review and merging the change into the code base" [26].

Comparative signals: Comparison with other projects. We compare the project with similar projects on the signals shown in the `Basic Stats` section. We identify comparable projects by the range of stars and topics set by projects.

We map our signals and their types in Table 2.

4.2.2 Computing Signals.

To construct the dashboard, we used PyGithub to pull projects' data in a certain past period. In our experiment setting, the period is four weeks. We used data from the most recent week for basic stats, indicative signals, and comparative signals, and data from all four weeks for temporal signals. We wrote a Python program that automatically pulls data for the last four weeks of a project, computes the signals, and generates a dashboard. The code is available in our supplemental material.[1] Here we describe how we defined and computed our signals.

Team management We considered all users who posted an issue or a PR as an `active contributor`. For each active contributor, we queried all their issues and pull requests posted to this project. A contributor's `experience in contributing issues/PRs` was calculated by the number of months since they posted the first issue/PR. Note that we considered issues and PRs as two different types of contribution: PRs are usually more involved because one needs to submit code or document with them. If it was the first time a contributor submitted an issue or a PR, we considered them a `new contributor`.

Social capital. We took the `average month of experience` of active contributors and the number of `recurring contributors`, *i.e.*, those who were not new contributors, to approximate bonding social capital, the benefit one could gain from a tightly connected social network [17]. We used `the number of new contributors` to approximate bridging social capital, a benefit one could gain from information diversity [10].

Responsiveness. `Close time` is defined as the time difference between the `created_at` timestamp and the `closed_at` timestamp we retrieve using the GitHub API. The `number of comments` includes only comments from human (non-bot) users.

When counting the number of issues/PRs that are `closed` or `still open`, due to the API rate limit,[6] for medium to large size projects, we could only retrieve issues and PRs created in the past four weeks and check each one for its `closed_at` timestamp.

Contributors' negative feelings. This information can help maintainers identify issues or PRs that may cause negative feelings

[5] See an example of our dashboard here: https://www.sophiehsqq.com/climate_coach/index_id.html

[6] https://docs.github.com/en/rest/overview/resources-in-the-rest-api?apiVersion=2022-11-28#rate-limiting

Table 2: Dashboard signals and their mappings to management categories (Phase 1 results)

Category	Signals	Types
Team management	`Number of new contributors`	Summarized, Temporal
	`Number of active contributors`	
	`List of active and new contributors`	Indicative
Social capital	`Number of recurring contributors`	Summarized
	`Average months of experience in the project`	
	`Number of new contributors`	Summarized, Temporal
Responsiveness	`Average close time`	Summarized, Temporal
	`Average num comments for closed conversations`	
	`Number of issues or PRs closed`	
	`Number of issues or PRs still open`	
	`Num conversations closed with 0 comments`	
Conversation tone	`Perspective API`	Temporal
	`Problematic conversations`	Indicative
Gamification	`Comparison to Similar Repositories`	Comparative
Contributors' negative feelings	`Open conversations with the most comments`	Indicative
	`Conversations opened for the longest time`	
Contribution management	`Labels of conversations and distribution`	Indicative

among contributors. We reported the top 5 open issues or PRs with `the most comments`, *i.e.*, potentially excessive rounds of reviews, and the top 5 issues or PRS that were `opened for the longest time`, *i.e.*, potentially long shepherding time [26].

Conversation tone. We fed all comments to Perspective API for toxicity and identity attack scores (both with range [0,1]). If one of the scores was higher than the threshold (we used 0.7), we reported it as a potentially `problematic conversation` on the dashboard.

Gamification. We identified projects similar to the focal project based on the number of stars, functionalities, or the number of contributors. When deploying the dashboard, we asked the maintainers to identify projects they considered similar to their project. For each project, we pulled their data using GitHub API for the past week and computed the following signals: `Number of Active Users`, `Number of Issues Closed`, `Number of PRs Closed`, `Average Time to Close Issues`, and `Average Time to Close PRs`.

Contribution management. We parsed the `labels` acquired from GitHub API and counted the number of issues, or PRs tagged with each label. We sorted the labels by the number of issues or PRs and plotted them as a bar chart.

4.2.3 Format and Layout. We first designed our dashboard as a GitHub issue. Our Python program output a markdown as a GitHub issue. The screenshots are shown in our supplemental material.[1] This initial design contains a subset of our signals.

After completing the first round of think-aloud studies with 10 participants, we rewrote our dashboard using JavaScript and turned it into an interactive webpage. We used `Bootstrap` to organize the layout and the `Chart.js` library to plot the charts. Using the format of a webpage instead of a GitHub issue can avoid *"off-putting"* maintainers (R1P9) because a GitHub issue *"doesn't mean good things for them"* (R1P9), *"needs to be closed"* (R1P4), and has a *"goal of reading it"* (R1P4). Moreover, using a JavaScript library can address participants' requests for high-resolution and interactive graphs (R1P2). This complete version of the webpage dashboard contains all features we present in Table 2.

Overall, we grouped our signals based on their types. Within each type, we put similar signals together, *e.g.*, `number of comments` and `average close time` were both under *How was the response* heading. After an overview of the dashboard (Figure 3), it first displays summarized signals that are numeric values (Figure 4). Then the dashboard displays indicative signals that point maintainers to conversations that might cause negative feelings among contributors (Figure 4). This is followed by temporal signals, showing several line charts of various signals' trends (Figure 5). These temporal signals are divided into two headings: *How big is your community* and *How was the response.* Beneath is the conversation tone analysis (Figure 6), a combination of temporal trends and indicative links. We put conversation tone signals close to the bottom of the dashboard because we did not want to show the negative signals too upfront. Following the conversation tone signals are the bar charts showing the usage of labels (Figure 7) and the comparison with similar projects (Figure 8). Lastly, the dashboard presents the comparative signals and concludes our dashboard with a summarization of methods and a list of references (Figure 9).

As R1P3 suggested, the default setting consists of line charts for a subset of the signals and we added drop-down buttons on the sides to allow users to select other signals to display, such as the median instead of the average.

4.2.4 Reinforce Inclusion Goals. After the first round of interviews, the dashboard only contained the open-source project's signals. However, Goggins et al. [33] described the importance of transparency and context with analytical signals. Therefore, we added `tips` throughout our dashboard to help maintainers improve their management strategies. These tips display results from prior studies on OSS management strategies, such as avoiding pushback [26, 62] in code review and adding a Code of Conduct [42]. The full list of tips is shown in Table 4 in our supplemental material.[1]

Moreover, we added sections `Methods` and `References` (Figure 9) for transparency, so our users could see our sources and the way we created the dashboard. We also added `Prior Research Results` section (Figure 9), which included `Features Affecting Project Attractiveness` to provide maintainers actionable suggestions in addition to presenting numerical signals.

4.3 Interview Results

4.3.1 Perceptions of Community Health. When asked about the criteria of community health, we found two major points of view. Similar to our findings in prior studies [20, 21], many maintainers thought of technical aspects. For example, R1P3 mentioned continuous integration (ci) as an indicator of community health, including *"how often is it being overwritten"* and *"build times"* (R1P3). Usage is also mentioned as a health indicator by several maintainers, including their dependencies and customers (R1P2), applications (R1P9), and the number of downloads (R1P6 and R1P10).

Another commonly mentioned perception concerns the social aspects. One of them is the community's sustained participation (R1P2, R1P3, R1P4, R1P5, R1P6, and R1P9). P4 pointed out that the `number of new contributors` indicates that their community is growing, which is a good sign. R1P2 also mentioned that the way they build their community is *"by engaging with groups of students who are going to implement new standalone tools that might be published as separate packages."* P9 commented on the same point, *"one big thing in terms of the developer community is like, [...] how do we figure out things that make people want to contribute and want to keep working on the project."*

Another health indicator concerning social aspects is the help maintainers can provide to the community, *i.e.*, responsiveness. Help includes maintainers' response to issues or pull requests (R1P2 and R1P9), documentation (R1P1, R1P4, and R1P9), and office hours (R1P1, R1P5, and R1P9). P2 acknowledged that *"a really bad way to ruin a community is by ignoring pull requests."* He further commented that the `number of pull requests that are still open` *"should probably be zero."* R1P1 even set a strict timeline of getting a response within one or two days. When looking at the summary of the `number of comments`, R1P4 pointed out that having good commentary indicates good health. R1P1, R1P5, and R1P9 all mentioned that a healthy community should have *"scheduled office hours that happen on a regular basis"* (R1P1) so that contributors *"can get help"* (R1P9). These points of view echoed the findings by Steinmacher et al. [81] that barriers newcomers face include the lack of responses from maintainers. The type of project health we focus on in this study aligns well with maintainers' concern for a project's social aspects – their community growth and sustainability.

4.3.2 Attitudes Towards Diversity and Inclusion. When asked about diversity, some commented that it was hard for them to know the level of diversity in their community (R1P6 and R1P10) because *"generally the only thing I see is their* GITHUB *username"* (R1P6).

Although some maintainers admitted that they cared about diversity and even desired more diversity (R1P1, R1P2, R1P4, R1P5, and R1P6), they were limited by their environment. For example, P1 told us that *"in <country> there's not a lot of diversity [in terms of race and ethnicity],"* especially since they mostly hire locals *"in a small town that's 70,000 people."* Hence, most of their members are white males. This idea is shared by R1P10, who listed several countries he interacted with and felt the ratio of women was lower in some of the countries. This observation aligns with some research results [59, 67]. On the contrary, being in a university, R1P2 experienced several occasions *"where all the students who were working on the project in our group were0 women."* When there was a lack of demographic diversity, maintainers considered diversity as a diversity of thoughts (R1P1).

Maintainers have generally taken action to improve the diversity of their community (R1P3, R1P4, R1P6, and R1P8). For example, with about 20,000 followers on Twitter, P6 tried to advocate diversity on social media. Some tried to *"sourcing people from different paths to provide programs to help educate people into the space better"* (R1P4). Several participants told us they try to improve diversity by being welcoming (R1P3, R1P4, R1P6, and R1P8).

4.3.3 Signals' Usefulness.

Conversation analysis. Three maintainers considered the conversation analysis to be the most important and useful feature of our dashboard (R2P5, R2P6, and R2P9). They found the tone analysis to be the *"the big selling point"* (R2P6) that could *"be highlighted much earlier in your reporting"* (R2P5). As R2P6 summarized, links to potentially problematic conversations were actionable items,

> *"[...] with these actionable things, you know, you can go actually take some sort of action to address concerns and anything that has a negative sentiment. Try to squash right away and make it more straightforward"* (R2P6).

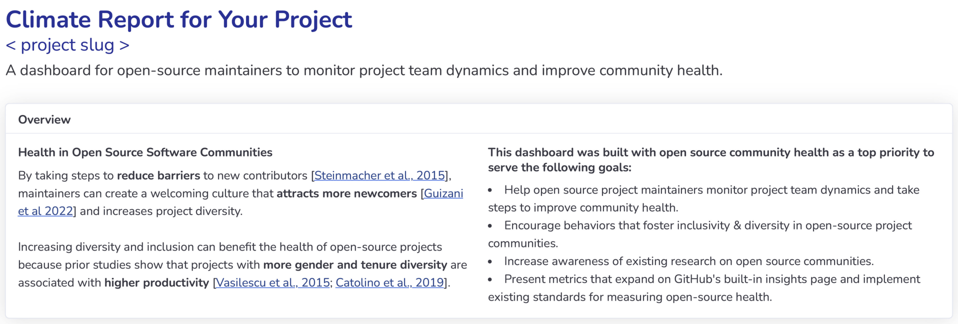
Potential pushback conversations. Links to conversations with long open time or many comments were considered to be useful by many maintainers (R2P4, R2P5, R2P6, R2P7, R2P8, and R2P9). Although some maintainers told us that some conversations were left open on purpose (R2P8), others told us that those conversations were the *"things [they] can look at and take action on"* (R2P6) and would even like to *"go and actually address these right now"* (R2P7). R2P5 echoed the findings of pushback in code reviews [26, 62] and pointed out that these conversations *"can almost directly correlate potentially to anything that's, you know, negative"* (R2P5). During our interview, the links even helped R2P8 identify a thread that waits for his reply while he thought he *"was waiting for her reply there"* (R2P8). As R2P6 nicely summarized, the links are

> *"sort of a daily dashboard where I can say, Oh, you know, here's my in-tray for the week, here's stuff that needs attention, here's stuff that may have fallen through the cracks, is something I need to pay attention to"* (R2P6).

This feature is more beneficial for big communities. R2P9, the maintainer of a project with more than 100 contributors, told us that our links helped them identify conversations that needed immediate attention because *"there are probably 50 parallel semi-active conversations going at any time, and we certainly can't track that"* (R2P9).

The number of closed issues and PRs. R2P4 told us that the number of closed issues and PRs is handy because they are a research institute, and they can put the data in their grant report:

> *"Knowing the pull request stats is very valuable, too, like the new authors. That one probably would be the most useful for us as far as reporting to our granting*



Climate Report for Your Project

< project slug >

A dashboard for open-source maintainers to monitor project team dynamics and improve community health.

Overview

Health in Open Source Software Communities

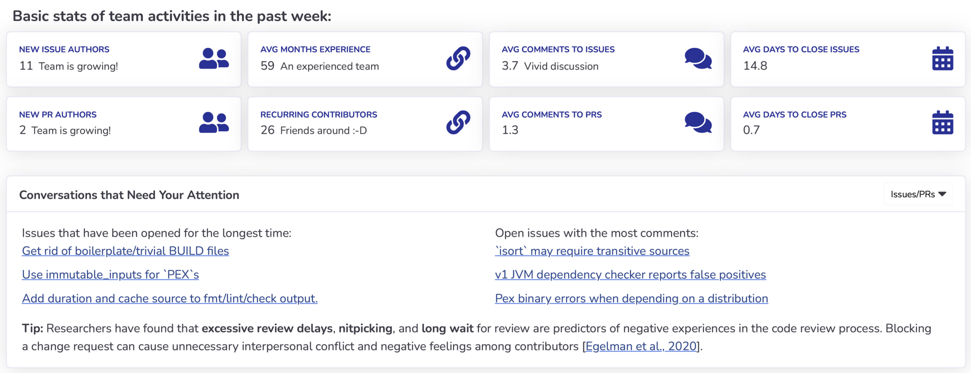
By taking steps to **reduce barriers** to new contributors [Steinmacher et al., 2015], maintainers can create a welcoming culture that **attracts more newcomers** [Guizani et al 2022] and increases project diversity.

Increasing diversity and inclusion can benefit the health of open-source projects because prior studies show that projects with **more gender and tenure diversity** are associated with **higher productivity** [Vasilescu et al., 2015; Catolino et al., 2019].

This dashboard was built with open source community health as a top priority to serve the following goals:

- Help open source project maintainers monitor project team dynamics and take steps to improve community health.
- Encourage behaviors that foster inclusivity & diversity in open-source project communities.
- Increase awareness of existing research on open source communities.
- Present metrics that expand on GitHub's built-in insights page and implement existing standards for measuring open-source health.

Figure 3: Version 2: Overview. The top of the dashboard shows an overview of our study and the goals we aim to achieve.



Basic stats of team activities in the past week:

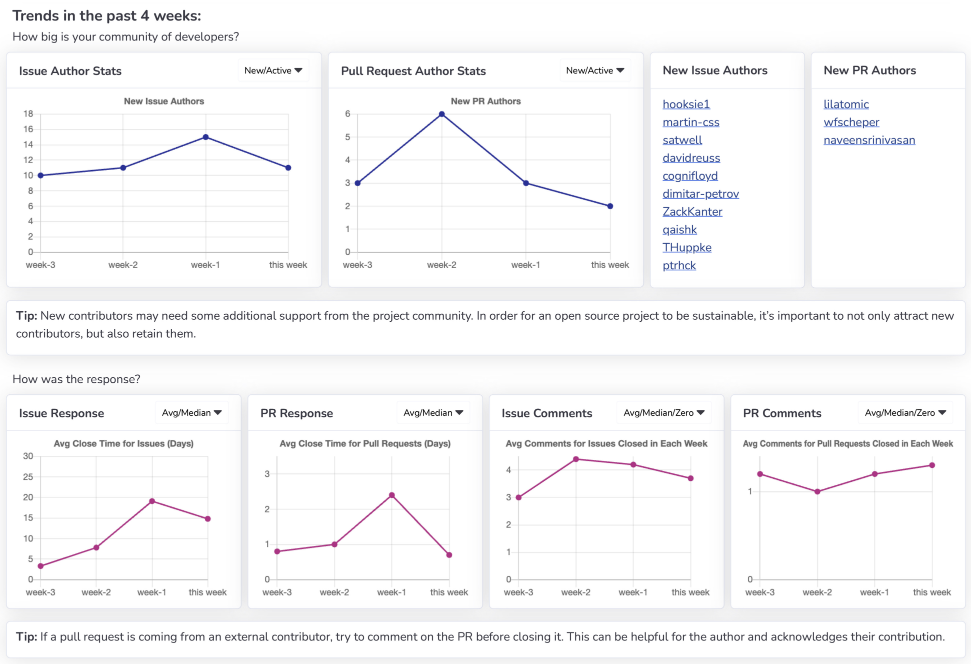
NEW ISSUE AUTHORS 11 Team is growing!	AVG MONTHS EXPERIENCE 59 An experienced team	AVG COMMENTS TO ISSUES 3.7 Vivid discussion	AVG DAYS TO CLOSE ISSUES 14.8
NEW PR AUTHORS 2 Team is growing!	RECURRING CONTRIBUTORS 26 Friends around :-D	AVG COMMENTS TO PRS 1.3	AVG DAYS TO CLOSE PRS 0.7

Conversations that Need Your Attention Issues/PRs

Issues that have been opened for the longest time:	Open issues with the most comments:
Get rid of boilerplate/trivial BUILD files	`isort` may require transitive sources
Use immutable_inputs for `PEX`s	v1 JVM dependency checker reports false positives
Add duration and cache source to fmt/lint/check output.	Pex binary errors when depending on a distribution

Tip: Researchers have found that **excessive review delays**, **nitpicking**, and **long wait** for review are predictors of negative experiences in the code review process. Blocking a change request can cause unnecessary interpersonal conflict and negative feelings among contributors [Egelman et al., 2020].

Figure 4: Version 2: Basic Stats. The dashboard shows some basic statistics regarding the activity level of the project.


Issue Author Stats New/Active
New Issue Authors
week-3 week-2 week-1 this week
Pull Request Author Stats New/Active
New PR Authors
Issue Response Avg/Median
Avg Close Time for Issues (Days)
PR Response Avg/Median
Avg Close Time for Pull Requests (Days)
Issue Comments Avg/Median/Zero
Avg Comments for Issues Closed in Each Week
PR Comments Avg/Median/Zero
Avg Comments for Pull Requests Closed in Each Week

Trends in the past 4 weeks:

How big is your community of developers?

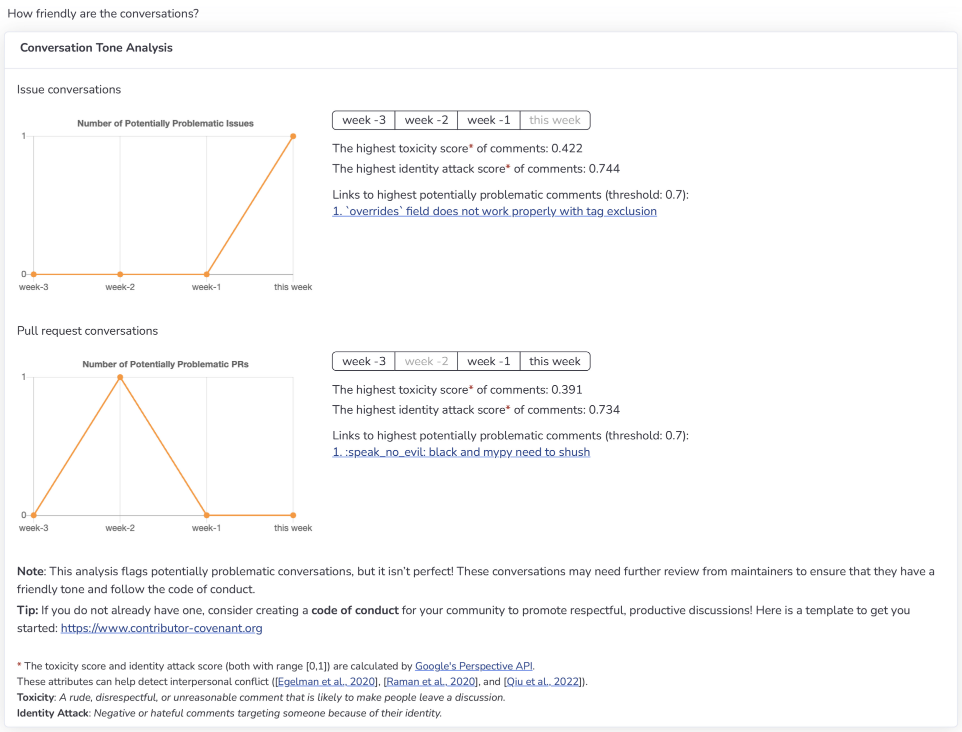
New Issue Authors	New PR Authors
hooksie1	lilatomic
martin-css	wfscheper
satwell	naveensrinivasan
davidreuss	
cogniflоyd	
dimitar-petrov	
ZackKanter	
qaishk	
THuppke	
ptrhck	

Tip: New contributors may need some additional support from the project community. In order for an open source project to be sustainable, it's important to not only attract new contributors, but also retain them.

How was the response?

Tip: If a pull request is coming from an external contributor, try to comment on the PR before closing it. This can be helpful for the author and acknowledges their contribution.

Figure 5: Version 2: Trends. This section of the dashboard displays trends of some useful metrics in the past four weeks to provide maintainers some context of how the project has been doing. This section also displays the logins of new contributors and active contributors with links to their profile pages.


How friendly are the conversations?

Conversation Tone Analysis

Issue conversations

The highest toxicity score* of comments: 0.422

The highest identity attack score* of comments: 0.744

Links to highest potentially problematic comments (threshold: 0.7):
1. `overrides` field does not work properly with tag exclusion

Pull request conversations

The highest toxicity score* of comments: 0.391

The highest identity attack score* of comments: 0.734

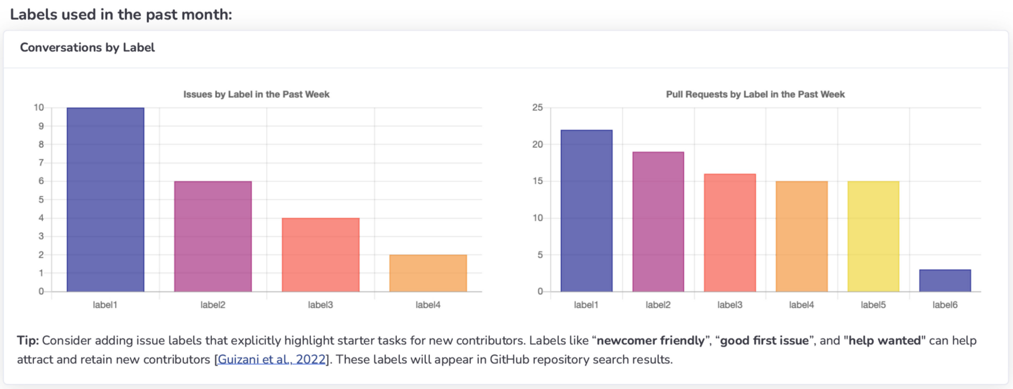
Links to highest potentially problematic comments (threshold: 0.7):
1. :speak_no_evil: black and mypy need to shush

Note: This analysis flags potentially problematic conversations, but it isn't perfect! These conversations may need further review from maintainers to ensure that they have a friendly tone and follow the code of conduct.

Tip: If you do not already have one, consider creating a **code of conduct** for your community to promote respectful, productive discussions! Here is a template to get you started: https://www.contributor-covenant.org

* The toxicity score and identity attack score (both with range [0,1]) are calculated by Google's Perspective API.
These attributes can help detect interpersonal conflict ([Egelman et al., 2020], [Raman et al., 2020], and [Qiu et al., 2022]).
Toxicity: *A rude, disrespectful, or unreasonable comment that is likely to make people leave a discussion.*
Identity Attack: *Negative or hateful comments targeting someone because of their identity.*

Figure 6: Version 2: Conversation Tone Analysis. This section shows the trends of the number of issues or pull requests that are labeled by the Perspective API as potentially problematic and provides the link to the posts. At the bottom, we explained how the scores were calculated and tips.


Labels used in the past month:

Conversations by Label

Tip: Consider adding issue labels that explicitly highlight starter tasks for new contributors. Labels like "**newcomer friendly**", "**good first issue**", and "**help wanted**" can help attract and retain new contributors [Guizani et al., 2022]. These labels will appear in GitHub repository search results.

Figure 7: Version 2: Labels Used by Issues and PRs. This section shows the labels the project used in the past week and the number of issues or pull requests with that label.

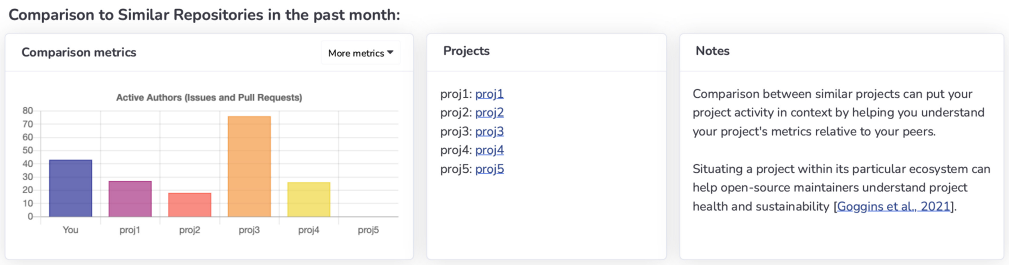
> *agencies and yearly reports where you just say like, Oh, this last year we closed like 300 tickets, and we opened like 6,000 or something"* (R2P4).

Labels. R2P6 and R2P7 mentioned that the numbers of issues or PRs under different labels are useful. R2P6 told us that they used *"labels to categorize pull requests for the change log"* so *"these labels actually matter to us"* (R2P6).

Average response time. R2P7 told us that the average response time is useful, especially since he oversees many GitHub repositories. He said it could make him aware that *"sometimes, [in] some repos, [...] people see [there is] an issue, and no one even responds to it"* (R2P7).

4.3.4 Dashboard Design Feedback and Changes.

Goals. There was some uncertainty about the title of our dashboard (R2P5) and the dashboard's purpose. Several interviewees


Comparison to Similar Repositories in the past month:

Comparison metrics

Projects

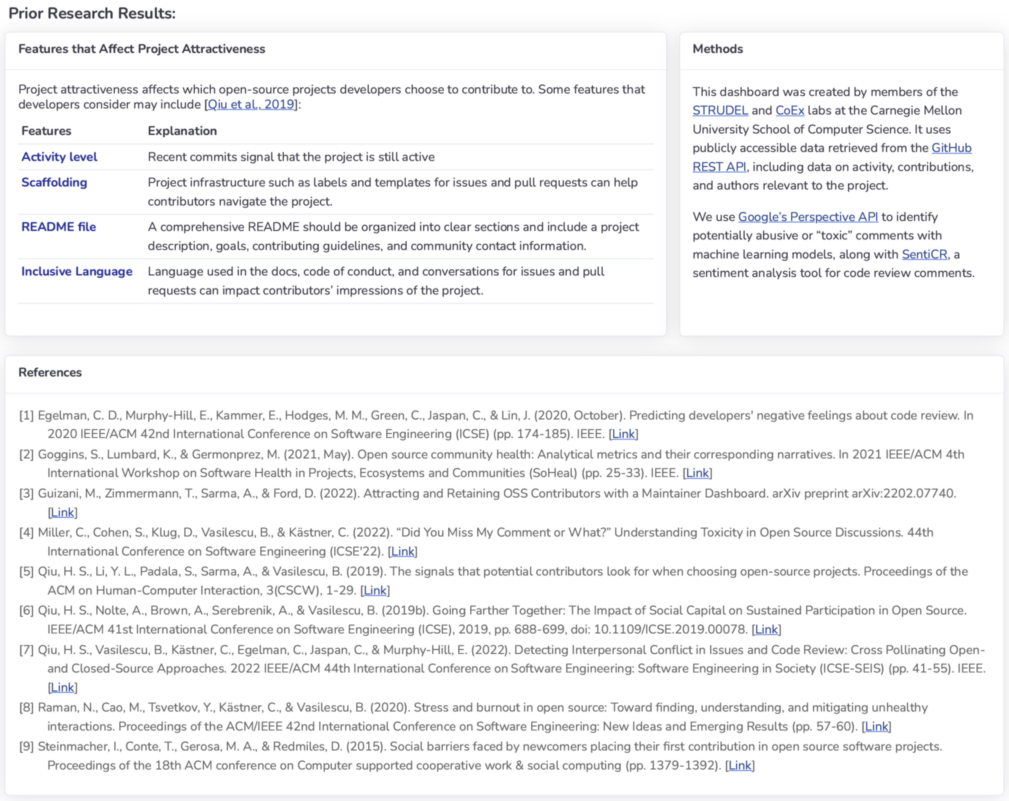
proj1: proj1
proj2: proj2
proj3: proj3
proj4: proj4
proj5: proj5

Notes

Comparison between similar projects can put your project activity in context by helping you understand your project's metrics relative to your peers.

Situating a project within its particular ecosystem can help open-source maintainers understand project health and sustainability [Goggins et al., 2021].

Figure 8: Version 2: Comparison. This section compares the project with peer projects on several metrics, such as the number of active users. The information serves as context for maintainers to interpret if their projects' metrics are relatively good or bad.


Prior Research Results:

Features that Affect Project Attractiveness

Project attractiveness affects which open-source projects developers choose to contribute to. Some features that developers consider may include [Qiu et al., 2019]:

Features	Explanation
Activity level	Recent commits signal that the project is still active
Scaffolding	Project infrastructure such as labels and templates for issues and pull requests can help contributors navigate the project.
README file	A comprehensive README should be organized into clear sections and include a project description, goals, contributing guidelines, and community contact information.
Inclusive Language	Language used in the docs, code of conduct, and conversations for issues and pull requests can impact contributors' impressions of the project.

Methods

This dashboard was created by members of the STRUDEL and CoEx labs at the Carnegie Mellon University School of Computer Science. It uses publicly accessible data retrieved from the GitHub REST API, including data on activity, contributions, and authors relevant to the project.

We use Google's Perspective API to identify potentially abusive or "toxic" comments with machine learning models, along with SentiCR, a sentiment analysis tool for code review comments.

References

[1] Egelman, C. D., Murphy-Hill, E., Kammer, E., Hodges, M. M., Green, C., Jaspan, C., & Lin, J. (2020, October). Predicting developers' negative feelings about code review. In 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE) (pp. 174-185). IEEE. [Link]
[2] Goggins, S., Lumbard, K., & Germonprez, M. (2021, May). Open source community health: Analytical metrics and their corresponding narratives. In 2021 IEEE/ACM 4th International Workshop on Software Health in Projects, Ecosystems and Communities (SoHeal) (pp. 25-33). IEEE. [Link]
[3] Guizani, M., Zimmermann, T., Sarma, A., & Ford, D. (2022). Attracting and Retaining OSS Contributors with a Maintainer Dashboard. arXiv preprint arXiv:2202.07740. [Link]
[4] Miller, C., Cohen, S., Klug, D., Vasilescu, B., & Kästner, C. (2022). "Did You Miss My Comment or What?" Understanding Toxicity in Open Source Discussions. 44th International Conference on Software Engineering (ICSE'22). [Link]
[5] Qiu, H. S., Li, Y. L., Padala, S., Sarma, A., & Vasilescu, B. (2019). The signals that potential contributors look for when choosing open-source projects. Proceedings of the ACM on Human-Computer Interaction, 3(CSCW), 1-29. [Link]
[6] Qiu, H. S., Nolte, A., Brown, A., Serebrenik, A., & Vasilescu, B. (2019b). Going Farther Together: The Impact of Social Capital on Sustained Participation in Open Source. IEEE/ACM 41st International Conference on Software Engineering (ICSE), 2019, pp. 688-699, doi: 10.1109/ICSE.2019.00078. [Link]
[7] Qiu, H. S., Vasilescu, B., Kästner, C., Egelman, C., Jaspan, C., & Murphy-Hill, E. (2022). Detecting Interpersonal Conflict in Issues and Code Review: Cross Pollinating Open- and Closed-Source Approaches. 2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS) (pp. 41-55). IEEE. [Link]
[8] Raman, N., Cao, M., Tsvetkov, Y., Kästner, C., & Vasilescu, B. (2020). Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results (pp. 57-60). [Link]
[9] Steinmacher, I., Conte, T., Gerosa, M. A., & Redmiles, D. (2015). Social barriers faced by newcomers placing their first contribution in open source software projects. Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing (pp. 1379-1392). [Link]

Figure 9: Version 2: Methods and References. At the bottom of our dashboard, we explain our methods and list the references.

mentioned they felt that this could be created by GitHub (R2P1, R2P6, and R2P7).

This feedback pointed out that our dashboard did not clearly convey its objective to maintainers. Due to this, our final design included an `Overview` (Figure 3) section that contained the background and goals for the dashboard.

Formatting and Design Decisions. We received feedback on the formatting and some design decisions, such as the use of colors and some features were missing. We made adjustments when two or more participants pointed out the same problem.

Feature Suggestions. Participants had a couple of suggestions of interesting features they would like to have in a dashboard: From a front-end functionality perspective, a few participants mentioned that they wanted a more interactive dashboard. One participant wanted to be able to drag the different dashboard sections around to customize it to their preference (R2P9). We could not address this feedback at the moment, but we did take note of which sections most maintainers felt were more important and should have been highlighted at the top of the dashboard. Additionally, participants wanted to be able to change the date ranges for the data (R2P1 and R2P4) to have a better idea of how their project developed over time. Unfortunately, we are not able to add this feature at the moment.

5 EVALUATION

After we finalized our design, we recruited a new group of 10 active GitHub maintainers to participate in a two-week diary study (Phase 3) so that we could further test the dashboard's usability and potential effectiveness. This section presents the findings from our two-week diary study.

5.1 Participant Information

We recruited a diverse group of participants. Four out of ten participants were women. Five of them have at least one woman or non-binary contributor. The years of experience ranged from less than a year to 10+ years (see Figure 1 in our supplemental material.[1] Most of them were involved in more than one OSS project (M = 5.9, SD = 5.22). The projects also varied in terms of popularity (23 to 18.7K stars) and team size (7 to 100+) (see Table 3 in our supplemental material).[1] Three of the participants were the sole maintainer of their project; the rest were either one of the maintainers or a lead maintainer with other specialized sub-maintainers.

5.2 Findings

5.2.1 Dashboard Effect. Overall, most participants agreed that the dashboard was useful to them (M = 3.75, SD = 1.16). Except for 2 participants, the rest expressed that they would continue to use this dashboard after the study. Most of them agreed that the dashboard would be helpful to most maintainers (M = 4.33, SD = 1.21).

Comparing participants' responses to the initial and exit surveys, we found that participants became more confident in supporting and encouraging a healthy community after using the dashboard. Overall, participants showed higher agreement with the statement *"I feel confident in supporting the community of contributors in my project"* (initial: M = 4.44, SD = 0.53; exit: M = 4.63, SD = 0.52). Three participants provided higher ratings in the exit survey than in the initial one. The other participants provided the same rating in both surveys. The exit survey also showed an improvement in the agreement with the statement *"I am sure about how to encourage a healthy project community"* (initial: M = 3.33, SD = 1; exit: M = 3.88, SD = 0.64).

While most participants acknowledged the usefulness of the dashboard, R3P7, the maintainer of a relatively small project, commented that, because his project is not very active, the dashboard would be more useful if the signals were aggregated by months rather than weeks.

5.2.2 Maintainer workflow. In both the initial and exit surveys, we asked maintainers to rate the importance of five goals, including fast response and recruiting new contributors. Participants were asked to give a rating between 1 (lowest priority) and 5 (highest priority) for each of the five goals. We ranked the average ranking of all participants and found that the priority order of the five factors did not change between the initial and exit surveys. We suspect that the diary study duration was too short for the maintainers' priority to change.

In both the initial and exit survey, most of the participants placed "attracting new contributors" as a lower priority (initial: M = 3, SD = 1.22; exit: M = 3, SD = 1.26). "Attracting a diverse group of contributors" has an even lower priority (initial: M = 2.44, SD = 1.42; exit: M = 2.17, SD = 1.47). Tasks with the highest priority are "fast response time to issues" (initial: M = 3.78, SD = 1.20; exit: M = 4.33, SD = 0.52) and to "PRs" (initial: M = 4; SD = 1; exit: M = 4, SD = 1.10). They are followed by "creating a welcoming environment" (initial: M = 3.78, SD = 1.09; exit: M = 3.5, SD = 1.22).

In the open-ended questions, maintainers added various goals they would like to achieve. These goals can be categorized into three groups: expanding the community (R3P2 and R3P7), accelerating response (R3P1, R3P3, and R3P7), and improving communication (R3P5 and R3P7).

Participants had extreme diversity in their frequency of responding to issues and PRs each week. Almost half of them indicated that they responded to issues and PRs 1-3 times per week, whereas some other participants indicated that they responded 10+ times per week. We have yet to discover a clear difference between the initial and exit survey in terms of the frequency of responding to issues and PRs.

5.2.3 Dashboard Feedback.

Useful signals. In each weekly survey, we asked participants to list out dashboard signals that they viewed more often than others. We found that participants paid attention to various signals. R3P1 and R3P9 paid more attention to the `Basic Stats` section at the top of the dashboard as they provide an overview of the project's status. R3P2 cared more about the `time to respond to issues and PRs` as he considered "fast response" to issues and PRs much more important than the other three goals. R3P4 and R3P7 mentioned that the trends are useful. The signal that is mentioned the most is `Conversations that Need Your Attention` (R3P4, R3P5, and R3P7) because it provides maintainers actionable items.

Although in the previous two rounds of interviews, we found that few R1s and R2s participants approved of the `comparison` section, it was considered valuable by some diary study participants (R3P2, R3P3, and R3P5). The comparison signals became helpful probably because they were being compared with projects they chose to be peers or competitors (by reporting them in the initial survey). However, R3P11 pointed out that comparison was difficult among projects because some projects have full-time contributors, whereas others do not.

Confusing signals. While participants agreed that most of the signals are *"self-explanatory"* (R3P2), some of them pointed out that the `Conversation Tone Analysis` part was confusing (R3P1, R3P2, and R3P5). R3P1 reported to us that he *"wanted to learn more about what the numeric score was. First, it would make more sense if it were just a percent (0%-100%), [but] it's currently a unitless number."* On top of the confusion on the measurement, we suspect the lack of toxic conversations made the `Conversation Tone Analysis` section empty and thus useless. None of our diary study participants had any conversations flagged by the Perspective API. However, we report the highest toxicity and identity attack scores regardless of the presence of any potentially toxic conversations, *i.e.*, toxicity or identity attack scores > 0.7. Future researchers can explore other ways of reporting toxicity or other tools for detection.

Helpful tips. The majority of the participants considered the tips in `Conversations that Need Your Attention` to be useful (R3P2, R3P3, R3P4, R3P5, R3P7, R3P9, and R3P11). Some participants also pointed out some tips that helped them improve specific parts of their projects. R3P7 told us that after viewing our tips on adding a Code of Conduct, he planned to add one soon. Several other participants mentioned tips of `Features that Affect Project Attractiveness` to be useful (R3P1, R3P2, R3P4, R3P5, and R3P11). R3P4 and R3P12 thought the tip in the section `Conversations by Label` was helpful. Unfortunately, although those maintainers considered some of the tips applicable, except for R3P7, who would add a code of conduct, none of them made adjustments by the time they completed the exit survey. Our diary study may need to be longer for maintainers to take concrete actions.

In summary, many of our diary study participants found this dashboard useful for themselves or other maintainers, and their level of confidence in supporting community health increased. However, our dashboard has yet to affect the maintainers' actual workflow.

6 DISCUSSION

Our study takes the first step towards visualizing signals related to the climate and the inclusion of open-source software projects but are hard to observe on current social coding platforms. From the user studies, we received positive feedback on our dashboard's usefulness. This section discusses some implications of our study and ideas for future research.

6.1 Contributions to Open-Source Software and HCI Communities

Our work directly contributes to improving open-source communities' community health and contributes to the broad HCI community in several ways. Firstly, our dashboard is one of the few efforts to provide an intervention to improve community health in open-source and is based on scientifically validated evidence. Our work builds on a wide variety of prior studies related to community health. We took inspiration from literature such as signals that newcomers use to select a project [60], barriers new contributors face [78, 80, 82], contributors' negative feelings [26, 50, 62], and contributors' sustained participation [61]. The literature provides strong support for the effectiveness of this intervention.

Our dashboard differs from the two prior works that built intervention tools, *i.e.*, Steinmacher et al. [78]'s portal for newcomers and Guizani et al. [35]'s dashboard for attracting and retaining newcomers. In addition to the fact that our design has all groups of contributors in mind, our dashboard also emphasizes the coaching aspect. The tips we provide in our dashboard are intended to help maintainers improve their management skills and increase their confidence in building a healthier community.

Secondly, our dashboard is mature after several rounds of design iterations and has proven to have effects on maintainers' confidence in improving project health. We disclose our code on GitHub so everyone can download and use it locally or share it with team members. We hope to see social coding platforms incorporating some of our signals to reach a broader audience. Researchers can thus collect more data to test their usability and effectiveness.

Thirdly, our findings from our diary study show that visualizing hard-to-observe signals is a promising strategy to increase maintainers' awareness of the status of their projects and improve their management strategies. Future studies can build on this idea and expand the set of signals to capture more nuanced project characteristics, making the management process more transparent and straightforward. For example, although some maintainers pointed out the usefulness of CIs and bots, we could not compute a signal based on the current trace data provided by GitHub. A collaboration with the social coding platforms would provide opportunities for further exploration.

However, we also like to highlight the flip side of being highly transparent. Constantly monitoring the signals can create extra stress on maintainers. Downward slopes in temporal signals can make maintainers worry about their performance. At the same time, if we make the dashboard publicly available, current and potential contributors can use the data to evaluate and judge maintainers' productivity and efficiency. Nevertheless, by knowing the signals' mechanism, maintainers can game the system to make the data look attractive.

Lastly, our more general contribution to the HCI community is that our design process and the findings can be widely transferred to other domains. Although the multi-phase iterative design process we adopted from Samrose et al. [68] was long and laborious, the outcome was effective and promising. More importantly, some of our signals are transferrable to other contexts that have similar settings and processes. Proprietary software development or remote collaboration can benefit from our results as well. For example, measures for pushback and toxicity in code review and other communications are equally important [62] for corporate software development. Pointers to conversations that might need further investigation can help managers distribute their attention and energy more wisely. Being able to provide timely responses also accelerates the project's progress.

6.2 Implications for design and future research

6.2.1 *Provide actionable feedback.*

Future work can explore the balance between simply displaying information that reflects the project's status and providing specific tips or instructions for maintainers to follow or implement. During our interviews, some participants appreciated that, in the GitHub issue version, we only provided maintainers with information and did not ask them to perform specific actions. However, some other maintainers reported that many of the `tips` in the web page version and `the links to the conversations that might need more attention` were useful. We argue that displaying only information limits the effectiveness of our dashboard if we do not also provide possible interventions backed by rigorous empirical studies. The number of tips we should provide can be very nuanced and needs further investigation.

6.2.2 *More signals.*

Future studies can also explore ways to incorporate more signals. When designing the dashboard, we ensured that our features were not redundant with the ones GitHub is providing. For example, GitHub already checks (on each project's `Insights -> Community Standards`) if a project has a README, among other forms of documentation, such as contributing guidelines and codes of conduct — all of which have been found to associate with higher project attractiveness to new contributors [60]. However, there are still a tremendous number of potential signals that we did not explore. For example, from our interviews, we also collected many signals that maintainers consider important but were hard for us to measure, such as the status of continuous integration (CI) builds. There are, however, many standard badges to reflect CI status [89], and these could be further integrated into a dashboard like ours.

Given the prevalence of bots [23, 93], interactions between humans and bots are also essential to consider in a maintainer dashboard. Our dashboard excluded bots' activities. Future studies can treat them as a separate group different than human contributors and analyze their behavior. A dashboard like Climate Coach could help maintainers assess where and how to utilize bots to support contributors. It could be helpful for project owners to understand how the use of various bots is associated with other participation signals, *e.g.*, contributors could be deterred by interaction with certain bots.

6.2.3 Long-term evaluation. Although our diary study shows promising results that maintainers consider themselves more confident in building a healthier community after two weeks of usage, we need more time to observe any substantial changes maintainers make in reaction to our dashboard. Long-term evaluation has been used in HCI [38, 91], especially some visualization projects [72]. Researchers can observe the changes in team size or the overall climate of a project in the long term and collect data on which signals are more valuable. For example, do temporal signals provide maintainers enough intuition on how their projects have been evolving? Researchers can also track changes that maintainers make and study if any of those changes are inspired by our dashboard, thus quantifying this intervention's effectiveness.

6.2.4 Incorporate our signals to the social coding platforms. Our work would be much more impactful if social coding platforms incorporate some of our signals into their design. Such integration can avoid context switching. Platforms also have the resource to conduct a larger-scale and longer-term user study, even an A/B test to compare the differences between the group of projects that adopt our dashboard and the group that do not. These data are valuable for improving the design and making the dashboard more helpful for maintainers.

7 LIMITATIONS

As with many studies, our paper has several limitations we would like to discuss here.

The diversity of our participants was limited by the low gender diversity in our participant pool. Although we managed to recruit four women maintainers for our diary study, we only had one woman maintainer in our Phase 2 interviews, which were essential to our dashboard design, and no non-binary participants throughout the entire study. Therefore, it is possible that we failed to incorporate some concerns that are unique to these marginalized groups. Future studies can recruit a more diverse user group to make the dashboard more inclusive. Another possible future work is to adopt the GenderMag approach [9], a cognitive walkthrough that helps software developers discover features that unintentionally exclude certain user groups.

Another limitation, as we admitted when describing our methods, is the gender inference process when recruiting participants. We acknowledge that there are more reliable methods than this one. However, since we had to review many projects to achieve a balanced sample in terms of gender diversity, we had to rely on certain heuristics to speed up the process. During the entire process, we did not use our assumed gender to address any individual contributors. We only relied on signals such as commonly used women's names, profile pictures, or self-reported pronouns. To lower the errors, we had more than one researcher browse the contributor list. If during our email interviews, a maintainer informed us that their project lacked gender diversity or had women or non-binary contributors, we corrected our record. For Phases 2 and 3, we reported participants' self-reported genders.

Lastly, the effectiveness of our dashboard on maintainers' actions could have been improved if the duration of our diary study were longer. Ideally, we would have liked to conduct a longer-term diary study to examine whether and how maintainers could integrate our dashboard into their process. As discussed above, we encourage future works to explore this possibility.

8 CONCLUSION

This paper presents Climate Coach, a dashboard we designed to improve the health of open-source communities. We first identified signals reflecting team inclusion by email interviews with maintainers. Based on the signals we identified, we designed a dashboard prototype and iteratively improved it with maintainers through think-aloud interviews. We tested the effectiveness of our refined dashboard through a two-week diary study with maintainers. Our results show that displaying signals that reflect various dimensions of teams' social aspects can increase maintainers' awareness of their community health and help them improve their management strategies.

REFERENCES

[1] Omar Alonso, Premkumar T Devanbu, and Michael Gertz. 2008. Expertise identification and visualization from CVS. In *International Conference on Mining Software Repositories (MSR)*. 125–128.

[2] Hirohisa Aman, Aji Ery Burhandenny, Sousuke Amasaki, Tomoyuki Yokogawa, and Minoru Kawahara. 2017. A Health Index of Open Source Projects Focusing on Pareto Distribution of Developer's Contribution. In *2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP)*. 29–34. https://doi.org/10.1109/IWESEP.2017.14

[3] Joop Aué, Michiel Haisma, Kristín Fjóla Tómasdóttir, and Alberto Bacchelli. 2016. Social Diversity and Growth Levels of Open Source Software Projects on GitHub. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*. ACM, 1–6.

[4] Guilherme Avelino, Eleni Constantinou, Marco Tulio Valente, and Alexander Serebrenik. 2019. On the abandonment and survival of open source projects: An empirical investigation. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–12.

[5] Jacob T Biehl, Mary Czerwinski, Greg Smith, and George G Robertson. 2007. FASTDash: a visual dashboard for fostering awareness in software teams. In *SIGCHI Conference on Human Factors in Computing Systems (CHI)*. 1313–1322.

[6] Erling Bjögvinsson, Pelle Ehn, and Per-Anders Hillgren. 2012. Design things and design thinking: Contemporary participatory design challenges. *Design issues* 28, 3 (2012), 101–116.

[7] Kelly Blincoe, Jyoti Sheoran, Sean Goggins, Eva Petakovic, and Daniela Damian. 2016. Understanding the popular users: Following, affiliation influence and leadership on GitHub. *Information and Software Technology* 70 (2016), 30–39.

[8] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[9] Margaret Burnett, Anicia Peters, Charles Hill, and Noha Elarief. 2016. Finding gender-inclusiveness software issues with GenderMag: a field investigation. In *Proceedings of SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 2586–2598.

[10] Ronald Burt. 1992. *Structural Holes: The Social Structure of Competition.* Harvard University Press.

[11] Andrea Capiluppi, Alexander Serebrenik, and Leif Singer. 2013. Assessing technical candidates on the social web. *IEEE Software* 30, 1 (2013), 45–51.

[12] Elian Carsenat. 2019. Inferring gender from names in any region, language, or alphabet. *Unpublished* 10 (2019).

[13] Amanda Casari, Katie McLaughlin, Milo Z Trujillo, Jean-Gabriel Young, James P Bagrow, and Laurent Hébert-Dufresne. 2021. Open source ecosystems need equitable credit across contributions. *Nature Computational Science* 1, 1 (2021), 2–2.

[14] Maëlick Claes, Mika Mäntylä, Miikka Kuutila, and Umar Farooq. 2018. Towards automatically identifying paid open source developers. In *International Conference on Mining Software Repositories (MSR).* 437–441.

[15] Maëlick Claes, Tom Mens, and Philippe Grosjean. 2014. maintaineR: A web-based dashboard for maintainers of CRAN packages. In *International Conference on Software Maintenance and Evolution (ICSME).* IEEE, 597–600.

[16] Jailton Coelho and Marco Tulio Valente. 2017. Why modern open source projects fail. In *Joint Meeting on the Foundations of Software Engineering (ESEC/FSE).* 186–196.

[17] James S Coleman. 1990. *Foundations of social theory.* Belknap.

[18] Valerio Cosentino, Javier Luis Cánovas Izquierdo, and Jordi Cabot. 2015. Assessing the bus factor of Git repositories. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER).* 499–503. https://doi.org/10.1109/SANER.2015.7081864

[19] Valerio Cosentino, Javier Luis Cánovas Izquierdo, and Jordi Cabot. 2018. Gitana: A software project inspector. *Science of Computer Programming* 153 (2018), 30–33.

[20] Kevin Crowston. 2003. Defining Open Source Software Project Success. In *International Conference on Information Systems (ICIS).*

[21] Kevin Crowston, James Howison, and Hala Annabi. 2006. Information systems success in free and open source software development: theory and measures. *Software Process: Improvement and Practice* 11, 2 (2006), 123–148.

[22] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW).* ACM, 1277–1286.

[23] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. 2020. *Detecting and Characterizing Bots That Commit Code.* ACM, New York, NY, USA, 209–219. https://doi.org/10.1145/3379597.3387478

[24] Santiago Dueñas, Valerio Cosentino, Jesus M Gonzalez-Barahona, Alvaro del Castillo San Felix, Daniel Izquierdo-Cortazar, Luis Cañas-Díaz, and Alberto Pérez García-Plaza. 2021. GrimoireLab: A toolset for software development analytics. *PeerJ Computer Science* 7 (2021), e601.

[25] Santiago Dueñas, Valerio Cosentino, Gregorio Robles, and Jesus M Gonzalez-Barahona. 2018. Perceval: software project data at your will. In *International Conference on Software Engineering: Companion Proceeedings.* 1–4.

[26] Carolyn D Egelman, Emerson Murphy-Hill, Elizabeth Kammer, Margaret Morrow Hodges, Collin Green, Ciera Jaspan, and James Lin. 2020. Predicting developers' negative feelings about code review. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE).* IEEE, 174–185.

[27] Nadia Eghbal. 2016. *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure.* Ford Foundation.

[28] Fabio Ferreira, Luciana Lourdes Silva, and Marco Tulio Valente. 2020. Turnover in Open-Source Projects: The Case of Core Developers. In *Brazilian Symposium on Software Engineering (SBES).* 447–456.

[29] Isabella Ferreira, Jinghui Cheng, and Bram Adams. 2021. The "Shut the f** k up" Phenomenon: Characterizing Incivility in Open Source Code Review Discussions. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–35.

[30] Anna Filippova and Hichang Cho. 2015. Mudslinging and manners: Unpacking conflict in free and open source software. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing.* 1393–1403.

[31] Oscar Franco-Bedoya, David Ameller, Dolors Costal, and Xavier Franch. 2017. Open source software ecosystems: A Systematic mapping. *Information and Software Technology* 91 (2017), 160–185.

[32] GitHub. 2017. Open Source Survey. http://opensourcesurvey.org/2017/.

[33] Sean Goggins, Kevin Lumbard, and Matt Germonprez. 2021. Open source community health: Analytical metrics and their corresponding narratives. In *International Workshop on Software Health in Projects, Ecosystems and Communities (SoHeal).* IEEE.

[34] Sean P Goggins, Matt Germonprez, and Kevin Lumbard. 2021. Making open source project health transparent. *Computer* 54, 08 (2021), 104–111.

[35] Mariam Guizani, Tom Zimmermann, Anita Sarma, and Denae Ford Robinson. 2022. Attracting and Retaining OSS contributors with a Maintainer Dashboard. In *International Conference on Software Engineering, Software Engineering in Society Track (ICSE SEIS 2022).* ACM.

[36] Jack Jamieson, Eureka Foong, and Naomi. Yamashita. 2022. Maintaining Values: Navigating Diverse Perspectives in Value-Charged Discussions in Open Source Development. In *Proceedings of the ACM on Human-Computer Interaction (CSCW).*

[37] Joselito Mota Jr, Railana Santana, and Ivan Machado. 2021. GrumPy: an automated approach to simplify issue data analysis for newcomers. In *Brazilian Symposium on Software Engineering (SBES).* 33–38.

[38] Helena Karasti, Karen S Baker, and Florence Millerand. 2010. Infrastructure time: Long-term matters in collaborative development. *Computer Supported Cooperative Work (CSCW)* 19, 3 (2010), 377–415.

[39] Karim R Lakhani and Robert G Wolf. 2003. Why hackers do what they do: Understanding motivation and effort in free/open source software projects. https://ssrn.com/abstract=443040.

[40] Amanda Lee and Jeffrey C Carver. 2019. FLOSS participants' perceptions about gender and inclusiveness: a survey. In *International Conference on Software Engineering (ICSE).* IEEE, 677–687.

[41] Michael J Lee, Bruce Ferwerda, Junghong Choi, Jungpil Hahn, Jae Yun Moon, and Jinwoo Kim. 2013. GitHub developers use rockstars to overcome overflow of news. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI).* ACM, 133–138.

[42] Renee Li, Pavitthra Pandurangan, Hana Frluckaj, and Laura Dabbish. 2021. Code of conduct conversations in open source software projects on Github. *Proceedings of the ACM on Human-computer Interaction* 5, CSCW1 (2021), 1–31.

[43] Johan Linåker, Efi Papatheocharous, and Thomas Olsson. 2022. How to characterize the health of an Open Source Software project? A snowball literature review of an emerging practice. In *International Symposium on Open Collaboration (OpenSym).* 1–12.

[44] M Lynne Markus. 2007. The governance of free/open source software projects: monolithic, multidimensional, or configurational? *Journal of Management & Governance* 11, 2 (2007), 151–163.

[45] Jennifer Marlow and Laura Dabbish. 2013. Activity traces and signals in software developer recruitment and hiring. In *ACM Conference on Computer Supported Cooperative Work (CSCW).* ACM, 145–156.

[46] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. 2013. Impression formation in online peer production: activity traces and personal profiles in GitHub. In *ACM Conference on Computer Supported Cooperative Work (CSCW).* ACM, 117–128.

[47] Christopher Mendez, Hema Susmita Padala, Zoe Steine-Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Simpson, Nupoor Patil, Anita Sarma, and Margaret Burnett. 2018. Open source barriers to entry, revisited: A sociotechnical perspective. In *International Conference on Software Engineering (ICSE).* 1004–1015.

[48] Vishal Midha and Prashant Palvia. 2012. Factors affecting the success of Open Source Software. *Journal of Systems and Software* 85, 4 (2012), 895–905.

[49] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian Kästner. 2022. "Did You Miss My Comment or What?" Understanding Toxicity in Open Source Discussions. In *International Conference on Software Engineering (ICSE).*

[50] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian Kästner. 2022. "Did You Miss My Comment or What?" Understanding Toxicity in Open Source Discussions. In *International Conference on Software Engineering (ICSE).* ACM.

[51] Courtney Miller, David Gray Widder, Christian Kästner, and Bogdan Vasilescu. 2019. Why do people give up flossing? a study of contributor disengagement in open source. In *International Conference on Open Source Systems (OSS).* Springer, 116–129.

[52] Audris Mockus, Roy T Fielding, and James D Herbsleb. 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 11, 3 (2002), 309–346.

[53] Dawn Nafus. 2012. 'Patches don't have gender': What is not open in open source software. *New Media & Society* 14, 4 (2012), 669–683.

[54] Michael Ogawa, Kwan-Liu Ma, Christian Bird, Premkumar Devanbu, and Alex Gourley. 2007. Visualizing social interaction in open source software projects. In *International Asia-Pacific Symposium on Visualization.* IEEE, 25–32.

[55] Sandra Ohly, Sabine Sonnentag, Cornelia Niessen, and Dieter Zapf. 2010. Diary studies in organizational research: An introduction and some practical recommendations. *First publ. in: Journal of Personnel Psychology 9 (2010), 2, pp. 79-93* 9 (01 2010).

[56] Siobhán O'Mahony. 2007. The governance of open source initiatives: what does it mean to be community managed? *Journal of Management & Governance* 11, 2 (2007), 139–150.

[57] Javier Pérez, Romuald Deshayes, Mathieu Goeminne, and Tom Mens. 2012. Seconda: Software ecosystem analysis dashboard. In *European Conference on Software Maintenance and Reengineering (CSMR).* IEEE, 527–530.

[58] Jessica Perrie, Jing Xie, Maleknaz Nayebi, Marios Fokaefs, Kelly Lyons, and Eleni Stroulia. 2019. City on the river: visualizing temporal collaboration. In *International Conference on Computer Science and Software Engineering.* 82–91.

[59] Gede Artha Azriadi Prana, Denae Ford, Ayushi Rastogi, David Lo, Rahul Purandare, and Nachiappan Nagappan. 2021. Including everyone, everywhere: Understanding opportunities and challenges of geographic gender-inclusion in OSS. *IEEE Transactions on Software Engineering* (2021).

[60] Huilian Sophie Qiu, Yucen Lily Li, Susmita Padala, Anita Sarma, and Bogdan Vasilescu. 2019. The signals that potential contributors look for when choosing

open-source projects. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–29.

[61] 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. https://doi.org/10.1109/ICSE.2019.00078

[62] Huilian Sophie Qiu, Bogdan Vasilescu, Christian Kästner, Carolyn Egelman, Ciera Jaspan, and Emerson Murphy-Hill. 2022. Detecting Interpersonal Conflict in Issues and Code Review: Cross Pollinating Open-and Closed-Source Approaches. In *International Conference on Software Engineering, Software Engineering in Society (ICSE-SEIS)*. IEEE, 41–55.

[63] Huilian Sophie* Qiu, Zihe H* Zhao, Tielin Katy Yu, Justin Wang, Alexander Ma, Hongbo Fang, Laura Dabbish, and Bogdan Vasilescu. 2023. Gender Representation Among Contributors to Open-Source Infrastructure - An Analysis of 20 Package Manager Ecosystems. In *International Conference on Software Engineering - Software Engineering in Society (ICSE SEIS)*. ACM.

[64] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. 2020. Stress and Burnout in Open Source: Toward Finding, Understanding, and Mitigating Unhealthy Interactions. In *International Conference on Software Engineering, New Ideas and Emerging Results (ICSE)*. ACM, 57–60.

[65] Dirk Riehle, Philipp Riemer, Carsten Kolassa, and Michael Schmidt. 2014. Paid vs. volunteer work in open source. In *Hawaii International Conference on System Sciences (HICSS)*. IEEE, 3286–3295.

[66] Gregorio Robles, Jesus M Gonzalez-Barahona, and Martin Michlmayr. 2005. Evolution of volunteer participation in libre software projects: evidence from Debian. In *International Conference on Open Source Systems (OSS)*. 100–107.

[67] Davide Rossi and Stefano Zacchiroli. 2022. Worldwide gender differences in public code contributions: and how they have been affected by the COVID-19 pandemic. In *International Conference on Software Engineering, Software Engineering in Society (ICSE-SEIS)*. 172–183.

[68] Samiha Samrose, Daniel McDuff, Robert Sim, Jina Suh, Kael Rowan, Javier Hernandez, Sean Rintel, Kevin Moynihan, and Mary Czerwinski. 2021. Meetingcoach: An intelligent dashboard for supporting effective & inclusive meetings. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–13.

[69] Anita Sarma, Larry Maccherone, Patrick Wagstrom, and James Herbsleb. 2009. Tesseract: Interactive visual exploration of socio-technical relationships in software development. In *International Conference on Software Engineering (ICSE)*. IEEE, 23–33.

[70] Pratyush N Sharma, John Hulland, and Sherae Daniel. 2012. Examining turnover in open source software projects using logistic hierarchical linear modeling approach. In *IFIP International Conference on Open Source Systems (OSS)*. Springer, 331–337.

[71] Jyoti Sheoran, Kelly Blincoe, Eirini Kalliamvakou, Daniela Damian, and Jordan Ell. 2014. Understanding watchers on GitHub. In *Proceedings of the International Conference on Mining Software Repositories (MSR)*. ACM, 336–339.

[72] Ben Shneiderman and Catherine Plaisant. 2006. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*. 1–7.

[73] Dan Sholler, Igor Steinmacher, Denae Ford, Mara Averick, Mike Hoye, and Greg Wilson. 2019. Ten simple rules for helping newcomers become contributors to open projects. *PLoS Computational Biology* 15, 9 (2019), e1007296.

[74] Vandana Singh, Brice Bongiovanni, and William Brandon. 2022. Codes of conduct in Open Source Software—for warm and fuzzy feelings or equality in community? *Software Quality Journal* 30, 2 (2022), 581–620.

[75] Davide Spadini, Maurício Aniche, and Alberto Bacchelli. 2018. Pydriller: Python framework for mining software repositories. In *Joint Meeting on the Foundations of Software Engineering (ESEC/FSE)*. 908–911.

[76] Ştefan Stănciulescu, Likang Yin, and Vladimir Filkov. 2022. Code, quality, and process metrics in graduated and retired ASFI projects. In *Joint Meeting on the Foundations of Software Engineering (ESEC/FSE)*. 495–506.

[77] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. Social barriers faced by newcomers placing their first contribution in open source software projects. In *ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW)*. 1379–1392.

[78] Igor Steinmacher, Tayana Uchoa Conte, Christoph Treude, and Marco Aurélio Gerosa. 2016. Overcoming open source project entry barriers with a portal for newcomers. In *International Conference on Software Engineering (ICSE)*. 273–284.

[79] Igor Steinmacher, Marco Aurélio Gerosa, and David Redmiles. 2014. Attracting, onboarding, and retaining newcomer developers in open source software projects. In *Workshop on Global Software Development in a CSCW Perspective*.

[80] Igor Steinmacher, Gustavo Pinto, Igor Scaliante Wiese, and Marco Aurélio Gerosa. 2018. Almost there: A study on quasi-contributors in open-source software projects. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE, 256–266.

[81] 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.

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

[83] Igor Steinmacher, Igor Scaliante Wiese, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2014. The hard life of open source software project newcomers. In *Proceedings of the 7th international workshop on cooperative and human aspects of software engineering*. 72–78.

[84] Chandrasekar Subramaniam, Ravi Sen, and Matthew L Nelson. 2009. Determinants of open source software project success: A longitudinal study. *Decision Support Systems* 46, 2 (2009), 576–585.

[85] 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 Comp Sci* 3 (2017), e111.

[86] Ferdian Thung, Tegawende F Bissyande, David Lo, and Lingxiao Jiang. 2013. Network structure of social coding in GitHub. In *European Conference on Software Maintenance and Reengineering (CSMR)*. IEEE, 323–326.

[87] Marco Torchiano, Filippo Ricca, and Alessandro Marchetto. 2011. Is My Project's Truck Factor Low? Theoretical and Empirical Considerations about the Truck Factor Threshold. In *Proceedings of the 2nd International Workshop on Emerging Trends in Software Metrics* (Waikiki, Honolulu, HI, USA) *(WETSoM '11)*. Association for Computing Machinery, New York, NY, USA, 12–18. https://doi.org/10.1145/1985374.1985379

[88] Bianca Trinkenreich, Igor Wiese, Anita Sarma, Marco Gerosa, and Igor Steinmacher. 2022. Women's participation in open source software: A survey of the literature. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 4 (2022), 1–37.

[89] Asher Trockman, Shurui Zhou, Christian Kästner, and Bogdan Vasilescu. 2018. Adding Sparkle to Social Coding: An Empirical Study of Repository Badges in the npm Ecosystem. In *Proceedings of the International Conference on Software Engineering (ICSE)*. ACM, 511–522.

[90] 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.

[91] Jari Varsaluoma and Farrukh Sahar. 2014. Usefulness of long-term user experience evaluation to product development: Practitioners' views from three case studies. In *Proceedings of the 8th Nordic conference on human-Computer interaction: Fun, Fast, Foundational*. 79–88.

[92] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark G.J. van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. 2015. Gender and Tenure Diversity in GitHub Teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3789–3798. https://doi.org/10.1145/2702123.2702549

[93] Mairieli Wessel, Bruno Mendes De Souza, Igor Steinmacher, Igor S Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A Gerosa. 2018. The power of bots: Characterizing and understanding bots in oss projects. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–19.

[94] Diego Winter, Guilherme Avelino, and Charles Miranda. 2022. HealthyEnv: a tool to assist in health assessment of software repositories. In *Brazilian Symposium on Software Engineering (SBES)*. 382–387.

[95] Jean-Gabriel Young, Amanda Casari, Katie McLaughlin, Milo Z Trujillo, Laurent Hébert-Dufresne, and James P Bagrow. 2021. Which contributions count? Analysis of attribution in open source. In *International Conference on Mining Software Repositories (MSR)*. IEEE, 242–253.

[96] Stefano Zacchiroli. 2020. Gender differences in public code contributions: a 50-year perspective. *IEEE Software* 38, 2 (2020), 45–50.

[97] Jierui Zhang, Liang Wang, Zhiwen Zheng, and Xianping Tao. 2022. Social Community Evolution Analysis and Visualization in Open Source Software Projects. In *International Conference on Web Information Systems Engineering*. Springer, 38–45.