Drinking From a Firehose While Herding Cats

A Search for Accountability in the Rust Open Source Community

NANDINI SHARMA, School of Computer Science, Carnegie Mellon University, USA
NARAYAN RAMASUBBU, School of Business, University of Pittsburgh, USA
JAMES D. HERBSLEB, School of Computer Science, Carnegie Mellon University, USA
BOGDAN VASILESCU, School of Computer Science, Carnegie Mellon University, USA

Accountability promotes responsible behavior, particularly in fulfilling obligations, achieving planned outcomes, and serving the collective good. While systems of accountability are well-structured in corporate environments, their role in open-source communities remains less understood. Existing research has explored requirements-gathering processes and motivations for contributing to open source, focusing on individual benefits. However, there is limited research on how these motivations and processes translate into accountability that guides collective actions to meet the needs of diverse stakeholders. This study examines accountability within the Rust community, particularly during the transition from roadmap-based planning to a goal-oriented process with designated owners within various teams. We provide a rich description, based on meeting notes, project documents and blog posts, as well as 21 interviews with a variety of project participants, to draw lessons about accountability in open source. We found that the Rust roadmapping process facilitated various forms of accountability but lost its effectiveness because (a) unfinished work items on the yearly roadmaps were unavoidable and made the process feel disingenuous; (b) the community culture shifted and facilitated a gradual decline of the roadmapping process; (c) external factors such as the pandemic and layoffs at Mozilla led to changes in the community priorities. Our findings suggest that accountability in open source is an emergent phenomenon that arises from participatory work processes around community engagement and that moral accountability, rather than transactional accountability, is crucial to fostering these processes.

CCS Concepts: • **Human-centered computing**; • **Software and its engineering** → **Collaboration in software development**; • **Social and professional topics** → **Project and people management**;

Additional Key Words and Phrases: Open Source Software Communities, Emergent Accountability, Collaborative Work, Community Engagement

ACM Reference Format:
Nandini Sharma, Narayan Ramasubbu, James D. Herbsleb, and Bogdan Vasilescu. 2025. Drinking From a Firehose While Herding Cats: A Search for Accountability in the Rust Open Source Community. *Proc. ACM Hum.-Comput. Interact.* 9, 7, Article CSCW404 (November 2025), 25 pages. https://doi.org/10.1145/3757585

1 Introduction

Accountability is generally seen as a prerequisite for responsible behavior, for fulfilling obligations, and serving a larger community. In many contexts, e.g., the corporate world, worker accountability is built into the structure and practices of management authority (e.g., [18]). However, the ways in which accountability mechanisms apply in an open source community has been a bit of a mystery.

Authors' Contact Information: Nandini Sharma, nandini2@andrew.cmu.edu, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; Narayan Ramasubbu, narayanr@pitt.edu, School of Business, University of Pittsburgh, Pittsburgh, Pennsylvania, USA; James D. Herbsleb, herbsleb@cmu.edu, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA; Bogdan Vasilescu, vasilescu@cmu.edu, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

ACM 2573-0142/2025/11-ARTCSCW404
https://doi.org/10.1145/3757585

There has been extensive research examining the motivation for contributing to open source. This research has largely focused on "joint products," or individual benefits that open source provides to those who build and maintain it (e.g., [24, 26, 36]). Other strands of research investigate the processes of gathering requirements for open source systems (e.g., [55, 57]). Relatively little is known about when, how, and why these motivations and processes bring about (or fail to bring about) the specific collective action that would satisfy the existing and emerging needs of various stakeholders of the open source software communities. While definitions may vary, an organization's stakeholders are generally considered to be "groups and individuals who can affect or are affected by it." [50] (p. 3). The most widely used projects play an infrastructure-like role, and their success requires understanding and meeting the needs of a diverse, dynamic, and evolving set of stakeholders.

The open decentralized style of decision-making, a typical characteristic of open-source communities, facilitates innovation but also adds to the uncertainty about accountability mechanisms given the lack of traditional hierarchical management tools and practices that would support accountability. In corporate settings, such tools and practices are often well-funded, but, in open source communities, lack of consistent and sufficient funding hamper community members' ability to establish a structure that would encourage accountability on both individual and group level. However, funding alone does not address issues related to ensuring accountability in open source. Attempts to enforce accountability may have undesirable side effects such as demotivating volunteers and shrinking the workforce. While several different hybrid arrangements involving both commercial entities and open-source communities have proven successful (e.g., RedHat's approach to Linux, and other examples of funded contributors supported by commercial firms), such arrangements have a number of inherent limitations, and pose significant risks, as recently evidenced in the Log4Shell and Heartbleed vulnerabilities. Even though these vulnerabilities have not been explicitly linked to accountability failures, lack of funding and insufficient number of maintainers for widely and extensively used open source software projects do merit raising accountability-related concerns.

Extant literature explicates the tools, practices, and flows of information in open source software communities but does not provide much insight into open-source accountability, e.g., the identification and engagement of stakeholders, the obligations to stakeholders as perceived by maintainers , the evaluation and prioritization of stakeholders' needs, the expression of development priorities to the maintainer community, and the extent to which selected priorities are honored in the development work. In this paper, we address these gaps through an examination of the Rust community's transition from roadmaps-based planning to an ownership-based allocation of work. Previous research has examined the roadmapping process in Rust, and found that despite the efforts devoted to the roadmapping process, a relatively small proportion of the development effort was devoted to the items in the roadmap [30]. Motivated by this work, we examine the phasing out of roadmapping in Rust. Specifically, we ask:

RQ1. *What does the phasing out of roadmaps in the Rust community demonstrate about the emergence of accountability in such contexts?*

We found that the Rust roadmaps facilitated various forms of accountability. However, regardless, the use of yearly roadmaps in the Rust community lost its effectiveness for the project members for three primary reasons: unfinished work items on the yearly roadmaps were unavoidable and made the process feel disingenuous, the community culture shifted and facilitated a gradual decline of the roadmapping process, and external factors such as the pandemic and layoffs at Mozilla led to changes in the community priorities. We discuss the implications of these findings and argue that (a) accountability in open source is an emergent phenomenon that is demonstrated through participatory work processes rather than work outcomes; and (b) moral accountability, as opposed to transactional accountability, is crucial for cultivating healthy participatory processes.

In the following sections, we provide a literature review discussing the economic importance of open source software, the need for considering accountability in open source, and the various ways in which accountability in open source software communities has been conceptualized previously. We conclude the literature review with a brief discussion of prior work on the roadmapping process in the Rust community and its relevance for exploring issues of accountability in open source software communities.

2 Literature Review

Since much open-source software is widely relied upon by business, government, and other organizations, there seems to be a clear need for accountability. Yet there is a fundamental tension between the nature of open source and dimensions of accountability. The desire to understand how open source communities manage and perhaps transcend this tension drives our research question.

2.1 The Economic Importance of Open-Source Software

While a few decades ago open-source software was mostly developed by volunteers and was used by a limited number of niche users, today, open source software is also developed by multi-billion dollar corporations and plays a central role in virtually every domain. Economists have begun to refer to open source, depended upon by billions of people, as “digital dark matter” [21], to signify both its invisibility and importance. A recent study commissioned by the European Union [5] estimated that companies located in the EU invested approximately a billion Euros in open source in 2018 (in the form of contributed engineering effort), which resulted in 65–95 billion Euros worth of impact on the EU economy. Economists [27] recently estimated the value of open source software worldwide to be more than eight trillion dollars. Open-source software also powers 94% of mobile devices, at least 96% of the most popular web domains, and 92% of the public cloud market.

Given the enormity of such economic reliance on open source software, one might think that developers, projects, and ecosystems are in some way accountable to the firms, governments, and citizens that rely on them. However, this is not necessarily the case. First, we address what we mean by accountability in the context of open source.

2.2 The Need for Accountable Open-Source Software

In its broadest sense, accountability refers to the “giving and demanding of reasons for conduct” [54]. Literature argues that accountability is a social practice that manifests itself in both individualistic and socializing forms, often examined as “systems of accountability” that embody a moral order of reciprocal rights and obligations [53]. Such systems of accountability have been well examined in the contexts of businesses and markets, where division of labor among individuals is accompanied by communication of a set of values and ideals of expected conduct. Accountability in commercial software production and supply chains is achieved through established organizational and institutional mechanisms. For example, participants can be subjected to a set of control mechanisms that aim to ensure individuals and teams conduct themselves in a manner that is consistent with meeting organizational objectives [29]. In addition, organizations delivering and utilizing enterprise software applications are subject to compliance to regulations enacted by government agencies, and they are held accountable for any weaknesses in these systems [37]. The importance of open-source software raises questions about the applicability of such systems of accountability to open-source environments.

2.3 The Elusive Concept of Accountability in Open Source

Contibutors in open-source software projects are generally free to make decisions about their contributions as they see fit. Such individual decisions bring about an emergent allocation of effort across projects. Besides this form of decision-making by individual contibutors, there is no obvious mechanism to influence direction of effort towards greatest needs of the project. Moreover, given the lack of financial compensation, the traditional organizational controls, market incentives, and regulatory scrutiny mechanisms cannot contribute to a system of accountability for open-source software in the same ways as they do for commercial software. Extant literature offers insights into this form of unfunded and unincentivized skilled work in open-source projects .

Ye and Kishida [64] and Wu et al. [63] found that participation in open source was often driven by a desire to learn. Shah [56] further elaborates that while participation was initially driven by personal needs, the few who continued to contribute became more deeply involved “hobbyists” and played a critical role in projects’ long-term viability. Hippel and Krogh [26] proposed a “private-collective” model which explains such volunteer participation in terms of the private returns that volunteers receive, including solutions to their own problems. Others have expanded the notion of private returns to include benefits such as signaling skills to potential employers [35, 65]. Given that “joint products,” or individual benefits explain participation in development and maintenance work, it highlights limited sources of external accountability in open source software communities [16].

David [13] suggests “forfeit[ing] our understanding of [accountability] as punishability” as implied by traditional commercial arrangements which are at odds with the very nature of open source. Instead, literature suggests that, in such settings, there is scope for alternative forms of domain-specific accountability to emerge.

2.3.1 Accountability Through Reputation. One mechanism through which open-source contributors might create accountability is reputation. Open-source communities are typically organized as meritocracies [35]. New developers progress through “the ranks” by demonstrating engagement, making high-quality contributions, building reputation, and earning trust as they advance to be included in a project’s “core” membership [15]. When this happens, contributors typically earn unrestricted commit rights to the code repository and have greater authority over a project’s direction. Researchers have argued that this desire to build and maintain reputation holds open-source developers accountable to their peers [13, 22, 41]. It remains unknown, however, how the same mechanism creates external accountability to open-source users, if at all.

2.3.2 Accountability Through Transparency. Another mechanism through which open-source communities might create accountability is transparency. Since the beginning of the open-source movement, there has been a great degree of openness not just with regards to the source code, but also the process through which source code of a project was developed. Such open and transparent processes have been theorized as “serve[ing] the need of answerability” [13]. More recently, open-source hosting platforms like GitHub have provided a multitude of visible cues about project activity (e.g., number of followers, issues, contributors etc.). This level of transparency enables people to make rich inferences about each other’s technical expertise and level of commitment [11, 40]. While transparency seems to be a key facilitator of overcoming barriers to accountability in open source, this literature frames accountability as auditability. There is still limited understanding of how transparency can contribute to creating a process or a culture in which maintainers perceive being answerable to external audiences.

2.3.3 Accountability Through Hybrid Models. Yet another aspect of accountability mechanisms can be understood from the perspective of hybrid open source models where private firms engage with open source communities. The focus of commercial firms partnering with open-source communities has been primarily to leverage community work for innovation and value-added services [51]. Firms producing and supporting commercial open-source products have varied business models that emphasize support and consulting services (e.g., Red Hat), split or dual licensing (e.g., MySQL), and value-added closed-source products with open-source components (e.g., SugarCRM, Microsoft, Oracle). Such hybrid arrangements allow commercial firms to provide accountability to users of open source as they try to compete with traditional proprietary software vendors [61]. Red Hat, for example, monitors fixes and features added to the open source Linux distribution Fedora and incorporates them in its Enterprise Linux only when those fixes and features are stable. Other firms that hire developers to work on open-source software create mechanisms of self-accountability. However, potential misalignments between the business models of corporate entities and the open-source community practices, including volunteers' perspectives and community licensing choices, may cause friction in establishing a system of accountability [38]. That is, corporate entities may not always obey norms, values, and rules established by communities [12] encouraging volunteer participants to adopt an "anti-capitalist commons" perspective [9]. Given these challenges, it is unclear how such hybrid models could institute systems of accountability.

2.4 Requirements Engineering, Governance Mechanisms, and Accountability

Literature on requirements engineering and open source governance mechanisms has the potential to provide the vocabulary to frame questions of accountability in open source settings. For example, prior research highlights complicated processes through which open-source maintainers elicit users' requirements through analysis of communication and surveys [28], personal experience with the open-source product [46], online forums [34], issue trackers [45], and a variety of online document genres sometimes called "informalisms" [55]. Projects also make use of techniques adapted from the commercial world [33], such as roadmapping [30]. Text analysis tools have been developed to partially automate the process of discovering requirements through online forums, feature requests, and other sources, e.g., by Vlas and Robinson [58]. Several studies have focused on the role of social interactions among stakeholders, suggesting, e.g., that communication centrality is associated with vagueness in requirements [19], and that certain stakeholders are particularly prolific contributors of requirements [4]. From the perspective of requirements engineering, open source ecosystem affords novel ways of conceptualizing accountability.

Similarly, prior literature on open-source governance has been useful in tackling collective action and coordination problems [2, 44, 47] and facilitating a conducive environment for sustaining community membership [56]. According to this literature, open source governance can be understood as structures of roles and responsibilities and formal and informal rules with respect to coordination and control of various project activities [39]. Surveys of open-source governance mechanisms reveal a diverse set of configurations that vary in their degree of openness, transparency, and allocation of authority [14, 48, 60]. However, these governance mechanisms are limited to the roles of core and peripheral members, focus on internal decision making processes, and do not constitute a robust system of reciprocal rights and obligations for enabling external accountability towards a broader set of stakeholders.

We conclude our literature review with a brief summary of Klug et al. [30], which provides background for the current work.

2.5 Prior Work on Roadmapping in Rust

The Rust community has followed a roadmapping process from 2016 through 2021, the purpose of which has been to formulate and align the work of the community with a set of shared technical and non-technical goals. The process has varied over the years but often included three primary stages:

gathering feedback from the community through blog posts and surveys, distilling the feedback into a set of goals, discussing the shortlisted goals among project members through the RFC (Request for Comments) process, and publishing those goals to the broader community. Over the years, the roadmapping process has changed to accommodate the needs of the project members. Overall, the project executed the roadmapping process five times and eventually stopped using roadmaps in 2021. Table 4 in the Appendix describes the primary differences across the five roadmaps.

Klug et al. [30] investigated the construction and use of roadmaps (primarily the 2018 roadmap) in the Rust community as a mechanism for coordinating technical work. Stakeholders were engaged through an invitation to write blog posts, through a survey, and through incorporating member suggestions into the Request for Comments (RFC) process. The core team then created and released the roadmap, which incorporated a subset of these suggestions. Looking at the technical work actually performed while the roadmap was current, the authors found that just over 20% of the lines of code contributed, by both team members and others, were related to roadmap items. The authors concluded that the roadmap provided some community focus, but was not strictly followed.

A few years after this paper was published, the roadmap process was eventually discontinued. Our interest was drawn by the apparent decision that roadmapping was not serving adequately to engage stakeholders and direct community efforts toward addressing their needs. We set out to find out why, and learn what we could about accountability in this context. We ask:

RQ1. *What does the phasing out of roadmaps in the Rust community demonstrate about the emergence of accountability in such contexts?*

We provide a rich description of community processes based on meeting notes, project documents and blog posts, as well as 21 interviews with a variety of project participants, to draw lessons about accountability in open source as roadmapping proved ineffective and fell out of favor. We discuss the process that replaced roadmapping in further details later in the paper. In the following sections, we provide a brief overview of our field site i.e, Rust, focusing on its relevance in the open source ecosystem and its organizational hierarchy.

3 Rust

Rust is currently the fastest growing programming language in terms of usage, with a 40% increase year over year.[1] It was recognized as "the most admired language" for the eighth year in a row by the 2023 Stack Overflow developer survey,[2] and is being included in both the Linux and Windows kernels.[3]

We chose to study Rust for several reasons apart from its relevance in open-source software development: Rust is an interesting case of a fast-growing open-source infrastructure project that is likely experiencing growing pains and a growing need for accountability; Rust has an elaborate governance structure; and Rust is heavily used in the software industry and is contributed to by the members of the software industry. Collectively, Rust makes for an interesting case of a hybrid open source model highlighted earlier in the Literature Review. In the remainder of this section we give a brief overview of how the Rust project is organized along with its roadmapping process.

3.1 The Organization of Rust's Leadership and Community

Currently, Rust has upwards of 5,000 contributors to its GitHub software repository.[4] This project has specific teams that focus on the language aspects of Rust, language compiler, relevant development

[1]https://github.blog/news-insights/research/the-state-of-open-source-and-ai/
[2]https://survey.stackoverflow.co/2023/#section-admired-and-desired-programming-scripting-and-markup-languages
[3]https://www.zdnet.com/article/rust-in-linux-where-we-are-and-where-were-going-next/
[4]https://github.com/rust-lang/rust

Table 1. Rust's team structure.

Team type	Team count
leadership-council team	1
top-level teams	7
sub-teams	48
project-groups	14
working-groups	45
marker-teams	33

tools and infrastructure, and has various other kinds of technical working groups and project groups that focus on issues cross-cutting across teams. There are around 300 project members that serve on specific teams and working groups. The project is divided into a set of top-level teams, sub-teams, working-groups, and project-groups (see Table 1). There are seven top-level teams including language, compiler, infrastructure, dev-tools, library, moderation, and launching-pad team. Each of the top level teams has sub-teams attached. Sub-teams that do not have a clear top-level team are attached to the launching-pad team. Apart from teams, there are project-groups, and working-groups each of which is also attached to a top-level team. For example, the language team has seven sub-teams, six project-groups, and three working-groups attached to it. In addition, there is a leadership council team which focuses on governance aspects of Rust such as communicating between the Rust foundation and the Rust project and overseeing project management processes.

4 Methods

This exploratory qualitative study consisted of three phases: familiarization, immersion, and data collection and analysis. During the first phase of the study we spent one month familiarizing ourselves with the history, work, and structure of the Rust project. We referred to four primary data sources in this phase: the GitHub repositories of the project, the online blog posts released by the community for members inside and outside the project, the Rust RFC book that contained an archive of all the technical and governance-related discussions of the project, and meeting notes taken by the Rust project members in their publicly accessible meetings. We used this extensive documentation to evaluate how the project members interacted with each other and the community members, community's pressing concerns as evident from the RFC discussions, Rust's team structure, and the work of members serving in leadership roles. The authors met every week to discuss primary insights emerging from this familiarizing process. For example, among other insights, this phase made it evident that the community had been doing structural changes to the leadership processes in order to address certain transparency-related concerns. This phase also helped us identify the primary means of communication and interaction among project members and prepared us for the second phase of the study i.e., immersion.

During the second phase of the study, we immersed ourselves in the community by attending their annual conference and online meetings. This phase helped us get an in-depth understanding of the social context of the community as we met with community members at the conference and in online team meetings. Participants at the conference included project members, Rust users, and project and conference sponsors among others. Attending conference talks and socializing with community members in-person helped us sample potential interviewees stratified across their roles as users, sponsors, and maintainers. We spent one month immersing ourselves in the community context, met each other on a weekly basis, and discussed emergent insights with regard to ongoing technical and governance-related discussions in the community.

Table 2. Interviewees' roles.

Role	Role description	Count
Internal Project Member	Individuals who are currently or have in the past served on Rust project teams or other groups	14
External Community Member	Individuals who have interacted with the project in the capacity of a user, contributor, and/or as an employee of sponsoring organizations but have not served on a project team or group	7
Total		21

These two phases conducted over a period of two months collectively helped us build an understanding of Rust's team structure, provided leads to recruit interview participants, and informed our semi-structured interview protocol.

4.1 Interview Data Collection and Analysis

Following the previous two phases of context immersion, we spent one month conducting 21 semi-structured interviews with users, sponsors, and project members of the Rust community and also explicitly evaluating roadmap-related discussions on Rust's GitHub repository. We had met Rust community members at the conference and we used those contacts to recruit participants for interviews through snowball sampling. See Table 2 for more details on participant characteristics. While meeting participants at the conference we made sure to get in touch with members both internal and external to the project and serving in the role of users, contributors, and sponsors in order to allow for a sample stratified by their roles and involvement with the project. We used insights from phase 1 and phase 2 to inform the interview protocol (see supplementary material) which largely focused on interviewee's role and involvement in the Rust community and the project, their motivation to engage with the project, their involvement in engaging stakeholders via the roadmapping process, their understanding of the ambitions behind the roadmapping process, and the reasons leading to phasing out of the process.

Interviews with community members provided an in-depth understanding of the social and cultural context in which the stakeholders were situated at the time the roadmapping processes were conducted. We interleaved interviews with regular weekly discussions with each other. In these discussions we discussed emerging ideas and specifically focused on how they related with community's roadmapping decisions over the years and what that explained about accountability in this context. We took notes of our discussions and used those notes to iteratively inform the interview protocol and note-taking strategy after the interviews. For example, during one of these regular meetings, we discussed how the participants found the Mozilla layoffs and the pandemic as a source of cultural change in the community and a shift in Rust stakeholders especially during the last year of the roadmapping process. Hence, we decided to add specific questions about the roadmapping process during the last year in order to collect more data about the context in which roadmapping-related decisions were taken.

For the purposes of data and context immersion, we read through the interview transcripts, listened to the interviews, prepared interview memos, and referred to the ideas we came across during the conference and meetings. This process helped us identify repetitive, recurring, and forceful ideas [49] across the data. Some of these ideas included "lack of resource," "interpersonal conflict," "burn out," "governance-related changes," "unclear technical rationale to proposal rejection," "external factors," etc. We discussed these descriptive codes with each other on a weekly basis and

Table 3. Categorization of thematic codes.

Descriptive focused codes	Thematic category
Lack of resources and burn out Shifting interests of existing project members Technical feasibility and unpredictability Governance-related issues owing to increasing size	Unfinished work items on the yearly roadmaps were unavoidable and made the process feel disingenuous
Unhealthy culture Recurring interpersonal conflict Emotional burnout Veil of technicality or technical veneer around technical discussions Members exiting the project Lack of trust in governance processes	The community culture shifted and facilitated gradual decline of the roadmapping process
Cultural changes and lack of structure with layoffs Uncertainty introduced by the pandemic Governance related restructuring due to Mozilla layoffs Members exiting the project Need for a legal entity	External factors such as the pandemic and layoffs at Mozilla led to changes in the community priorities

extracted segments of the interview transcripts that aligned with these codes. Two team members analyzed transcripts and created codes separately and then compared them. Once we had a list of these descriptive codes, we grouped them based on themes [8, 10, 52]. Specifically, we focused on how closely these codes aligned with each other and with the research question in explaining the phasing out of the roadmapping process. For example, “governance-related changes,” “burn-out,” “lack of resources,” were combined under the category “Unfinished work items on yearly roadmaps were unavoidable and made the process feel disingenuous” because they explained the reasons why work was not getting done by the end of the year. Similarly, “interpersonal conflict,” “trust issues within teams,” “ambiguous technical rationale” were combined under the category “Community culture shifted and facilitated gradual decline of the roadmapping process” because they explained the cultural issues that were instrumental in project members losing faith in governance processes. These themes were reflective of work dynamics among members serving on teams internal to the Rust project. Thus, we used comments from internal project members as evidence to support these themes in the Results section. However, we also included some comments from external project members as a confirmation of internal team dynamics, a triangulation mechanism, and an indication that these dynamics were prevalent enough to have been observed and noted by members outside the project who were not necessarily serving on internal teams, and had fewer reasons to be directly involved in team discussions and to be directly impacted by team decisions. As a member checking mechanism we shared this paper with both internal and external project members to gather their feedback and comments. See Table 3 for more details on categorization of these codes into three primary themes. This thematic analysis led to three primary findings that we discuss below.

5 Results

Data suggest that while roadmaps were intially introduced to rally collective effort, they, indeed, facilitated emergence of various forms of accountability in the community. However, over a period of five years (2017–2021), the use of yearly roadmaps in the Rust community lost its effectiveness for the project members for three primary reasons: unfinished work items on the yearly roadmaps were unavoidable and made the process feel disingenuous, the community culture shifted and facilitated a gradual decline of the roadmapping process, and external factors such as the pandemic and layoffs at Mozilla led to changes in the community priorities.

5.1 The Roadmap Process Facilitated Various Forms of Accountability

As we expected based on our review of the literature (Section 2 above), the roadmap process helped keep the internal project members accountable to the community of Rust users (many of whom contribute to the project in various forms themselves). Contributing proposals for the roadmap was one of many ways in which community members provided input, and there was an expectation that the internal project members would listen and act on these suggestions:

> "Something I liked about the [roadmap item] vision process is that we explicitly pulled in things from users, people that are using these features." (P13 – Internal Project Member)

More broadly, the roadmap process provided accountability through transparency. Internal project members acknowledged that being clear about what is being worked on and what isn't, which was the roadmap's intent, "is a big part of transparency" (P1 – Internal Project Member) and that one could take advantage of this to infer what the project was up to, and to determine its stakeholders:

> "You can look and say 'Hey, this is being worked on, this is being worked on, these are the people who are interested, and by extension these are the companies that are interested.' I think this is important to fewer people but I think it is actually more important for ensuring honesty and so forth." (P1 – Internal Project Member)

The roadmap was also seen as a way for internal project members to be accountable to the project's corporate sponsors:

> "Corporate sponsors who are paying for storage, paying for Rust development. Not only can you see, every release, what we're working on, [but also] here are the big features that are coming down the pipe. That will satisfy whatever corporate stakeholders are at this company or that company, or at the foundation, etc, who are holding the purse strings." (P15 – Internal Project Member)

Despite the emergence of accountability through the roadmap process, the tasks listed on the roadmaps often remained pending at the end of the year, to the point where the project eventually stopped following the roadmapping process:

> "We kind of stopped doing roadmaps a few years ago... We would produce this document that says, 'We're going to do X, Y, or Z,' and then a lot of those just didn't happen." (P4 – Internal Project Member)

In the following section, we share the reasons the roadmap process was discontinued.

5.2 Unfinished Work Items on the Yearly Roadmaps Were Unavoidable and Made the Process Feel Disingenuous

A common theme throughout the interviews was that the roadmaps fell out of favor because there was **no mechanism to enforce that roadmap items got implemented** and the community had to prioritize **governance-related issues over technical work**. Enforcing the roadmap was challenging because it wasn't always clear how to prioritize roadmap items:

> "It is important to have the perspective that input from the community is just input. It is not like requirements or whatever. And, we might put different weights on that." (P1 – Internal Project Member)

However, some inputs from members internal and external to the project were recognized as more important than others, in particular those coming from "big users":

> "I have always been in terms of getting input and feedback much more about prioritizing real industrial users rather than enthusiasts." (P1 – Internal Project Member)

The relative weight of members acting in the capacity of corporate users was recognized as a source of tension. This was partly given the community's perception of the project's accountability towards itself, which created a need to strike a balance:

> "What we have tried to do really hard is avoid taking that too far and set in the direction based on who is funding things." (P1 – Internal Project Member)

Second, interviewees mentioned that the roadmap's capacity to imply accountability towards internal and external members of the project who engaged in the capacity of users was limited. Roadmaps were commonly seen as "speculative and completely unenforceable" (P15 – Internal Project Member), because contributors can typically choose what they wanted to work on, as expected in an open source community. As one interviewee puts it:

> "We can't just say we're going to accomplish these five things and hold people specifically accountable to going and accomplishing those things because they can choose what they work on. They can choose to step away. They can choose to do whatever they want." (P19 – Internal Project Member)

In other words, "in open source, it's hard for there to be any real accountability because you can't really force anyone to do anything" (P17 – Internal Project Member).

More pragmatically (as compared to ideologically in the previous examples), roadmap items were not tied to effort allocations:

> "The old road mapping process didn't really have hard commitments from individuals in the teams to either ensure something happens or do the work themselves, or do the reviewing work." (P4 – Internal Project Member)

In practice, this had various consequences. For example, it meant that the project teams often didn't have the effort available to implement the roadmap items they were interested in and had to wait for external contributors. An interviewee indicated that "If nobody shows up to do that work, it's not getting done" (P17 – Internal Project Member). However, the hope that somebody else was going to show up was acknowledged as "a very optimistic perspective" (P4 – Internal Project Member), and there was a perception among some that corporate users weren't contributing their fair share of effort in helping develop the features they needed:

> "A lot of companies will be like, 'well, we want this thing.' 'Okay. So are you spending any of your engineering time on it?' 'Oh, no, no, we won't do that'... at least in this case it's like, 'all right, you have to provide some support now.' (P9 – Internal Project Member)

As typical in mature open-source projects, the contribution process demanded that new contributors met with project teams, developed enough context about the project's needs, and socialized with relevant members. However, this rarely happened and when external members' work deviated from project's expectations, it resulted in a perception of wasted effort:

> "You want people to come along and solve problems but what is really bad is... somebody comes, they find some problem… they work on that for a year and then you say 'No'. And they feel terrible.. And probably quit and walk away. And, you feel terrible because you have thrown away that work." (P1 – Internal Project Member)

In other cases, roadmap items turned out to be more ambitious, or less technically feasible, than planned: “It ended up being more complicated than we thought” (P4 – Internal Project Member). Project members acknowledged that they found themselves steering away from the roadmap:

> “Plans always change. So, if we announce a thing… six months in you might be like… actually the thing we thought was easy was hard. And the thing we thought was hard became very easy all of a sudden.” (P12 – Internal Project Member)

> “Especially in software, things always take longer than people think they will. ” (P4 – Internal Project Member)

Moreover, prioritizing technical roadmap items in any given year was not always possible. For example, as the size of the Rust project increased over time, it became necessary for the core team to focus on **governance-related issues**. That is, managing the governance of the project took priority over technical features:

> “A traditional core team in a small open source project is responsible for everything... But once you get big enough, the core team ends up having no part in the technical direction... only being about governance… One tension that is created is that people on the core team end up having no time to actually do technical work.” (P1 – Internal Project Member)

Given that the internal project members serving on the Rust core team had the technical expertise and necessary historical context of the project to drive roadmap items, they had to gradually transition into technical leadership and community management roles. It was evident from the ways in which roadmaps changed over the years: they became less technical and included more governance-related items. The last roadmap released by the community explicitly mentioned not including any technical work but only “project-health” related tasks. Another interviewee confirmed that the efforts of the core team, recently re-structured as the Leadership Council, was largely consumed by governance-related work:

> “The role of the Leadership Council is to… speak as the Rust project. So… the technical concerns of the language and all that tend to be best done by the teams… a lot of their [leadership’s] job is interfacing with the Rust Foundation, with the project directors, with the community. And then cross-cutting concerns within the Rust Project… making sure the project is well supported” (P12 – Internal Project Member)

Many interviewees also talked about how the roadmap process required a lot of project management work i.e., summarizing minutes, tracking issues, coordinating with the various stakeholders, etc.:

> “The old roadmap process, in practice, ended up really falling on one person very heavily. It was essentially like a full time job to do the roadmap work.” (P13 – Internal Project Member)

In addition, much of this “project management” work, while generally recognized as important, was seen as uninteresting by many:

> “Tracking and keeping up to date on the current status of things is a lot of work, and it’s also very boring work; most people don’t like doing it.” (P17 – Internal Project Member)

> “If I wasn’t paid to do this I would still be writing programs at my house, and putting them up online and all that. I don’t know that people really do project management as a hobby.” (P13 – Internal Project Member)

> “The programming part builds up energy for me. Doing the blog post and some other stuff is energy draining. There’s a limit to how much I am able to do those things.” (P19 – Internal Project Member)

Given these reasons, the roadmap process eventually became “a little useless because it didn’t really change anything”, was “extra work for no reason” , was “repetitive” (P17 – Internal Project Member), and seemed “disingenuous” (P4 – Internal Project Member), with internal and external project members becoming “a little jaded about the value of roadmaps” (P19 – Internal Project Member). In the words of a senior project member:

> “We stopped doing roadmaps a few years ago... there was just a lack of steam behind the process… it felt a little disingenuous to say that… this is a priority, or, this is what we plan to do.” (P4 – Internal Project Member)

5.3 The Community Culture Shifted and Facilitated Gradual Decline of the Roadmapping Process

At times, the project suffered from an unhealthy work culture due to **interpersonal conflict**, stalling of work owing to **the perception that personal subjective opinions were often masked as objective technical rationales**, and **diminished trust** in the governance processes. Internal and external project members found it difficult to work and engage with the project on multiple occasions. Some project members emotionally burned out “due to this kind of stuff” (P1 – Internal Project Member) and left the project. Those who stayed continued to deal with these issues on a daily basis. Given that the roadmapping process was driven by the leadership team and was reportedly a communication mechanism that brought the community together and motivated them to work towards a set of common goals, such cultural factors led to the roadmap gradually losing favor over the years.

A project member asserts that much of the persistent **interpersonal conflict** among project members and a continuing unhealthy work environment could be, in part, attributed to the organizational culture at Mozilla that was the key organization that incubated and sponsored Rust:

> “Rust inherited some of this culture from Mozilla… The people involved have had [the view], ‘Let’s pretend that everyone gets along and everything is always happy and.. No one ever has any problems and can always solve everything by consensus and there’s never any conflict’... Which works really well until it doesn’t. And, then, everything goes really badly wrong and I think that has been really difficult to change.” (P1 – Internal Project Member)

While it was not a surprise that interpersonal conflict existed, the concern, as highlighted in this comment, was that such a culture was rarely addressed in a way that would stop it from perpetuating. Another project member recalled that when interpersonal conflicts occurred, those conflicts were not handled properly and eventually led to people exiting the project:

> “In 2018… two [working groups] imploded over interpersonal arguments about how and what should be done, and in what way… It was handled poorly… it was just not handled at all. And then it eventually became very bad, and everyone quit." (P17 – Internal Project Member)

Per Rust’s 2018 roadmap, which was the second iteration of the roadmapping process, the community had planned to release Rust 2018 edition. This roadmapping year was marked not only with interpersonal conflict but also deadline pressures. It was a stressful situation for project members given that the editions were rolled out every three years. Project members raced to accomplish features that would otherwise have had to wait for the next cycle:

> “2018 was a very important release for us, and it changed the language for the better in a lot of ways. But also… it burned a lot of people out… it was very much like, ‘I am working around the clock for the last couple of weeks’ … that was far, far too much.” (P17 – Internal Project Member)

Blog posts received for planning of the subsequent 2019 roadmap asked the community to “go slow” and “to be empathetic and responsive”. The blog posts also acknowledged that “it’s much harder to pinpoint specific people who need to modify their behavior in specific ways.” The 2019 roadmap itself explicitly mentioned that “2019 will be a year of rejuvenation and maturation for the Rust project”. These ideas emphasized that in 2019, the community acknowledged the burn out issues not just owing to the workload but also the interpersonal conflicts in the project. However, these issues did not get resolved because of multiple reasons; one of them being that the community was missing the organizational support they needed (e.g., Human Resources) which was only available to them at the time the project was supported by Mozilla. An interviewee confirmed that interpersonal conflict continued to be an issue in their day-to-day work. Speaking to the support they had in navigating such conflicts, the interviewee mentioned "very little" and that it was not a "healthy work environment":

> "Today I was a little bit in a bad mood because I was again on the receiving end of some of that… if I were to quit the project, it would be for reasons like that. We can have technical disagreements all day long. But, if people… cross into… constant insults… This is not a constructive or healthy work environment" (P12 – Internal Project Member)

Other interviewees mentioned that they often perceived push back under a **"veil of technicality"** during technical discussions, which stalled progress on the technical designs they would propose; the push back seemed disingenuous to them:

> "[The proposals] will be softly denied… They just won't get put on meeting agendas, and they won't get a lot of feedback, or instead of engaging earnestly with it, a lot of people will just respond with, 'Do we really need this?' … [or], 'Oh, well, this is interesting. But what's your use case?' Even though the use cases [is] obvious… Whenever something comes along that isn't part of their plan... [they] stall or push back on doing anything like that... (P7 – Internal Project Member)

This interviewee was explaining stalling of or "slow rolling" of technical design proposals led by members from outside relevant groups or teams: "If you stall someone else's proposal long enough and keep making progress on your own...eventually your proposal will be the one that gets implemented." Another interviewee concurred that this was a long standing issue in the project:

> "This is one of our biggest flaws… A lot of issues will have a veil of technicality... but they're not actually about technical things… They're just 'someone believes' opinions, dressed up in fancy words... there's a lot of technical veneer. (P12 – Internal Project Member)

Arguments in certain technical discussions were not perceived as genuine. As a consequence, conflicts were not resolved but believed to have been won over through what was referred to as "soft power". This tendency fed into shaping the community over time:

> "A lot of the big name features… were always introduced as, 'Hey, here's this crazy idea. Can we just integrate it and see if it works?' And generally it was like, 'Yeah, sure we can. We can put this in and see if it works'. But nowadays, there's a lot more of 'we shouldn't even put this in in the first place, because we have alternatives'. There's a lot more of 'don't try at all' than there is 'try and see if it works or not.' (P7 – Internal Project Member)

Responding to such conflicts, a member posted on a publicly accessible chat that they found it "emotionally draining" that they were responsible for having to explain such conflicts to their employer and Rust users who they thought were "personally accountable" to in terms of delivering specific features. This conflict occurred in reference to an issue which was, at the time, a roadmap item. An external community member who was aware of the conflicts within teams mentioned feeling pessimistic about certain features they needed:

> "My impression is that it was basically a recurrence of the same fractures… So now I'm very pessimistic about it [the feature] again. (P11 – External Community Member).

A senior project member shared that a part of this mistrust was owing to external project members not assuming good faith on part of the internal teams:

> "If you're a contributor coming to Rust, and... ask for a feature, the assumption of good faith, is not that your feature has been rejected on just weird grounds that don't make any sense… The good faith... is that it probably was rejected, but it was probably done for a very real reason... [In] open source, a lot of folks feel very entitled... to have the perspective of 'you gave me the software for free, and therefore I can ask anything of you', and you must give it to me for free… " (P18 – Internal Project Member)

This skepticism that others might not be operating in good faith encouraged **mistrust in governance processes** such as roadmapping. Some interviewees felt that the roadmap was and continued to be a top down process which was not "necessarily healthy." The top down nature of roadmaps was also quoted as a motivation for adopting new processes going forward. Furthermore, this mistrust and skepticism was also noted as contributing to a lack of transparency in governance. Expressing their skepticism and mistrust, an interviewee recalled personal experiences that made them feel disillusioned:

> "There are a lot of people who are in positions of power within the community, both social and true political power who, I don't think, understand the nature of power and they don't understand how their actions affect others. (P9 – Internal Project Member)

Speaking to the possible causes of such mistrust, a senior maintainer mentioned that a leaders' respect figured into whether or not a community could trust them. In the context of Rust, as the leaders moved away from technical work and transitioned into governance related work, they lost the trust and respect of the community:

> "Your respect as a leader starts with your technical work and when you stop doing your technical work and you have people who have less of that background then it is very hard to maintain that over time. (P1 – Internal Project Member)

Governance related restructuring in the project i.e., the resignation of the moderation team followed by gradual stepping down of core team members within a period of a few months, coincided with phasing out of the roadmapping process. While the last roadmap was discussed and finalized it was never announced to the community. Given that roadmapping was driven by the core team from early days, this mistrust and "fractures" (P11 – External Project Member) persisted through the years. The last roadmap specifically mentioned focusing on the project-health related items only. Cultural issues got in the way of the community's ability to align themselves with the roadmap work. Speaking to whether or not the community explicitly decided to stop using roadmaps, multiple interviewees confirmed that roadmaps gradually lost favor with the community and that project members "stopped caring" (P17 – Internal Project Member) about it over time:

> "I think a lot of people sort of slowly came to the realization that the roadmaps were not working as well as they could" (P18 – Internal Project Member)

> "Plans are good for coordinating between groups of people. And if people don't care about the plan, then, it can't serve its purpose sort of by definition. So, I don't think there's any time that it [the roadmap] really worked." (P17 - Internal Project Member)

The community was dealing with the pressure of having to deliver work while being short on resources and simultaneously dealing with cultural and governance-related issues within the project. While part of this pressure was related to the accountability that project members had towards their respective employers, for those who worked on volunteer basis, it was, as an interviewee put it, "self imposed":

> "For the people… for whom it wasn't their job to work on Rust, it was very much self imposed, 'I care about this very deeply. And so, you know, I want to succeed. And so I'm gonna do what I have to do to make sure that that happens."' (17 - Internal Project Member)

The comment indicates that despite the pressure and emotional burnout, some project members were able to stay involved because of their investment and belief in the future of the project. However, external factors such as the pandemic, and layoffs at Mozilla made it challenging for project members to continue to engage in similar capacities as before.

5.4 External Factors such as the Pandemic and Layoffs at Mozilla led to Changes in the Community Priorities

A series of external events beginning with the pandemic in February 2020 followed by layoffs at Mozilla were cited as potentially responsible for the ways in which project members engaged with the community. The two incidents led to an increased uncertainty about the future of "the world" and the project which, per the interviewees, **changed priorities** of the community members, **lack of in-person socializing** prevented existing members to rally for the roadmap and discuss cross-team dependencies, and **enabled departure of project members** in significant numbers. As individual priorities shifted in response to this uncertainty, it contributed to the phasing out of the roadmapping process:

> "There were also other things happening around 2020. So it's possible that we're blaming more on roadmaps… We were doing an all-hands pretty regularly until the one that didn't happen... People had different social skills before the pandemic than during the pandemic or after the pandemic." (P8 – Internal Project Member)

> "People's priorities shifted a bit… 2020 was a year of fairly low contributions. It felt like everyone was like, 'OK, let's make sure the project keeps going. But this is not necessarily our core priority right now.' I contributed very little in 2020… I quit my contracting business… took a paid job because I was like, 'I don't know what's going to happen. I don't know if there will be contracts. I'd rather be an employee right now." (P12 – Internal Project Member)

Besides the understandable shift in personal priorities in 2020, cancellation of conferences led to **zero socializing at conferences** and the inability of members to rally the community around the roadmapping process, which was a regular practice earlier. A senior project member acknowledged that presenting roadmaps at conferences was important as it built excitement within the project:

> "You know, [a project member] is at conferences, and every time giving a talk about.. 'Here's the Rust roadmap, and here's what we're working on… it's so exciting'… telling the community what we are trying to do… Talking about this roadmap is more about building excitement and energy within the project and getting people to say, 'Yeah, I believe in that vision. And I'm gonna devote my volunteer efforts to try to do that"'. (P13 – Internal Project Member)

> "The main effect that the pandemic had was… the lack of conferences… was making it harder for us to have some of the more casual conversations… high velocity conversations… particularly with the cross-team stuff… we all had our own video calls that were doing a decent job… within teams. But, any of the cross team conversations, we don't have a lot of coordination points across team except at conferences. (P19 – Internal Project Member)

Cross-team collaboration was important. The process of undergoing the roadmapping process would make it evident to the project members how their work was cross-cutting the work of other project teams. Some members even acknowledged that this process of discovering cross-team dependencies was more important than the work tasks on the roadmap itself. The pandemic limited opportunities for project members to get together and discuss such dependencies. However, at the time the pandemic hit, the community, for reasons mentioned earlier, was already winding down the roadmapping process. A project member highlighted that it was emotional distress rather than the lack of in-person meetings that exacerbated already existing interpersonal issues within the community:

> "The pandemic didn't really disrupt any of that kind of stuff. I do think that… raising the stress levels of everyone involved, because the world feels like it's going to end, does not help with interpersonal conflict that was already existing before." (P17 – Internal Project Member)

With no opportunities to see each other in person, the distress and uncertainty proved to be an additional burden. During the same year, Mozilla announced layoffs of some project members. Even though the community had been planning to move away from Mozilla's governance and towards establishing a foundation for a number of years, the layoffs at Mozilla marked a point in time when the community not only lost a significant number of people but also lost a formal structure to rely on. Speaking to the layoffs, a maintainer explained that the community witnessed a **departure of many project members** and experienced much uncertainty and frustration during that time:

> "Uncertainty… disappointment, frustration… there was also just a bleed out of people participating in the project, because people who were working full time…either their new job and their life circumstances made it so, they didn't have time anymore, or they burnt themselves out during their time working for Mozilla... we lost a good segment of people around that time." (P19 – Internal Project Member)

Additionally, the transition from Mozilla left the community with no support of a formal organizational hierarchy. Even though the project had its own governance mechanisms at the time, it was still a challenge for the community to manage the transition from Mozilla:

> "There was a really big cultural shift within the Rust project after the Mozilla layoffs. Because we went from a structure that, not completely mind you, but at least partially, represented an org chart in a formal sense as well within the company." (P12 – Internal Project Member)

While the community was already dealing with resource crunch well before the pandemic, the layoffs contributed to a slowing down of work that was necessary to creating and maintaining roadmaps. Furthermore, without external organizational support, the community was pressed to establish a legal entity that would be responsible for funding the Rust project. This work took more time and attention of the project members:

> "We are extremely open source but Mozilla kept the trademarks. And it was felt that trademark protection was important… Mozilla didn't want to have that role but we needed a legal entity to have that and that was the original motivation for the foundation… we needed an entity that can own the trademark and have a bank account." (P1 – Internal Project Member)

These transitions were external shocks to the community that diverted its attention away from roadmap work; both technical and social. It was not a surprise that while the project members were mostly attempting to manage these transitions (2020–2021), no roadmap was created for the subsequent years (2022–2023).

6 Discussion

Findings indicate that changes to the roadmapping process reflected shifting needs of project members over the years. In that sense, the roadmapping process was consequential for and reflective of how accountability operated in the community. For example, when members engaging in the roadmap management suffered burn out, the community stopped doing individual tracking of roadmapping items which was "a full time job". When interpersonal conflicts within teams led to mistrust in governance processes, technical work on the roadmap yielded to "project-health related work". When external factors such as the pandemic and layoffs presented existential threat to the community, items on the roadmap became broad enough to allow project members and teams to work as they saw fit through the year. We argue that these changes to the roadmapping process and its gradual phasing out offers two key take-aways. First, accountability was demonstrated through participatory work processes rather than through predicted and/or accomplished work outcomes. Second, moral accountability, as opposed to transactional accountability, was crucial for cultivating participatory work processes.

6.1 Accountability is Demonstrated Through Participatory Work Processes Rather Than Work Outcomes

As discussed earlier in the paper, literature suggests that stakeholders in different organizations invest various kinds of resources and derive benefits from the work produced with the help of those investments [32] . Given this, the work outcomes of organizations tend to be held accountable to the needs of such stakeholders. However, in this open source community, accountability manifested not through work outcomes of the community but through the processes the community followed to produce that work. Data indicates that roadmap items were not as important as the roadmapping process itself. Regardless of whether or not items listed on the roadmap resulted in the intended outcome, the roadmap enabled processes that were helpful in keeping the community afloat and in providing entry points for outside community members to engage with the project.

Many interviewees confirmed that the process of creating and maintaining roadmaps allowed members to (a) direct external contributors' concerns about community work towards the roadmap; (b) understand cross-team dependencies revealed while creating the roadmaps; (c) show intent for transparency to all stakeholders; and (d) communicate stability and progress to external members. Prior work on this community aligns with these findings in that the work done by the community in 2018 drifted away from the roadmap items that year [30]. And yet, the community continued to engage in the process despite its heavy demands on time and effort and low returns as far as work outcomes were concerned. It was through changes to the roadmapping process (and not through the work items listed on the roadmap) that project members signaled and perceived their accountability towards other members and towards the community, more broadly.

Interpersonal conflicts in the community were also related to project members' expectations of how work ought to be done in the community, in addition to, what work was done or which features were implemented. For example, interpersonal conflict was often related to the perception that external contributors were not able to try out their own design proposals or that the internal project members did not engage in technical discussions about those proposals in good faith. Those concerns were primarily process-related (e.g., how RFCs were reviewed and approved) rather than concerns about what outcomes were accomplished. The RFC discussions of rejected proposals demonstrated that community members inside and outside the project were invested in conflicting processes; members serving as maintainers were invested in managerial processes and had reasons to contain RFC discussions while outside members were invested in getting community feedback and had reasons to encourage RFC discussions. One of the most prevalent complaints of burn out was also a process issue because it questioned how workload and interpersonal conflict in the community was handled over time

Moreover, findings demonstrate that work outcomes in the community were instrumental in deflecting accountability rather than shouldering it. This is contrary to the arguments extended by extant literature on stakeholder engagement which suggests that work outcomes help identify relevant stakeholders [32], and that work outcomes, and value creation are critical to defining and demonstrating stakeholder welfare [17, 23]. Put simply, prior literature postulates that work outcomes of a community hold participants accountable to the stakeholders of those communities. However, roadmap items helped Rust project members bypass their employers' expectations of them in subtle ways. Broader work items (as against specific technical work items) on the roadmap made it easier for them to make the case that they accomplished the work they were accountable for as per the roadmap. Kraut [31] has argued that collaboration among members of online communities were less successful in getting work done when work tasks were broadly defined. Roadmap items also enabled project members to avoid having to explain to external contributors why some of their contributions were rejected; misalignment with the roadmap was a convenient explanation. Both of these cases could be understood as ways in which project members utilized roadmapping items to absolve themselves of the expectations (legitimate or otherwise) of their employers and external contributors.

Thus, we argue that accountability in open source is better understood through participatory work processes of the community rather than its work outcomes. This process perspective makes it apparent that accountability in open source is emergent and vulnerable to frequent shifts. As a community reacts to and accommodates internal and external demands, it undergoes process changes, which are indicative of changes in its priorities. This is contrary to extant literature's explanation on how accountability operates in for-profit organizations where processes are established as an explicit acknowledgement of their stakeholders' interests and needs[43] and are less vulnerable to scrutiny from the participants of such organizations (e.g., shareholders, employees, clients, users etc.). However, in open-source communities, processes are subject to heightened scrutiny from different stakeholders because stakes are rarely openly acknowledged. It is through asking questions and demanding changes to how work happens that stakeholders assert their needs, presence, and participation. It was evident in how Mozilla encouraged adoption of the roadmapping process in the community and how the community's reaction to project members' way of working led to changes in the way the leadership organized work. This is to say that when processes shift, they suggest that a community is reacting to internal and external pressures indicative of its shifting priorities and its emergent accountability to different stakeholders.

6.2 Moral Accountability is Crucial for Cultivating Participatory Work Processes

Our case findings indicate that community members expected each other to be held accountable for ensuring a healthy work environment. When project members were perceived as exercising power and not understanding "how their actions affect others" (P9 – Internal Project Member), it demonstrated members' lack of accountability in demonstrating empathy and transparency. Having more power and status, and therefore increased work-related responsibilities, did not imply concessions towards the demands of community-engagement. On the other hand, when internal and external project members were perceived as asserting entitlement to their needs because of the open-source nature of the community, it demonstrated their lack of accountability in understanding the "costs of maintainership" (P18 – Internal Project Member). Having less decision making power did not absolve them of this community responsibility. Literature has shown that this dynamic is an outcome of accountability being understood as transactional [3].

That is, responsibilities in a system based on transactional accountability are determined based on "conditions" that "ensure that each actor is held to account, and that sanctions can be applied when actors do not fulfill their obligations" (pp-797). In the Rust community, when project members serving as maintainers made decisions in a non-transparent manner, they were chastised publicly, and when project members violated code of conduct, they were expelled from the community. These repercussions were evident when interviewees acknowledged (a) being quite wary of online public controversies that explicitly called them out; or (b) feeling disillusioned and disappointed when senior members got expelled for code of conduct violations. Research highlights that this dynamic signifies presence of multiple conflicting accountabilities where members are influenced by "power asymmetries" and "prioritize certain accountability relationships over others" despite being aware of potential sanctions in response of their actions [3] (pp-797). This form of transactional accountability is fear-based rather than empathy-based and is problematic especially when community members are "not well-known" to each other which is often the case in large open source software communities [3]. Such dynamic also leaves much scope for misunderstanding and interpersonal conflict as was demonstrated by the findings of this study.

To differentiate transactional accountability from the accountability stemming from participatory processes and community engagement, we posit the notion of "moral accountability" in open source communities. Per this notion of moral accountability, the burden of ensuring healthy community processes must be shared across all participating members alike, irrespective of power and status. As the Rust community shifted away from roadmapping, we noticed a growing need for participatory processes that would uphold members' moral accountability towards healthy community engagement. For example, when interpersonal conflict occurred, some project members felt reluctant in intervening and asserting their decision rights as legitimate decision makers for the fear of being "written up in a headline on the register.com, or something, for… yet another Rust drama" (P6 – Internal Project Member). While this is a trust issue, it also signifies that some community members perceived others as not adhering to a structure that would honor and yield to others' decision rights. This is not surprising given that formal organizational structures run the risk of framing accountability as an obligation and transactional [3] rather than as a personal, social, and moral responsibility [7]. It explains why the use of a new code of conduct in the community led to further mistrust in the community rather than providing a structure that would encourage moral behavior. Bovens [7] highlights the need for establishing structures in organizations that would encourage members' civic accountability towards each other.

Moral accountability on the part of project members serving in a variety of (and often conflicting) roles may also serve to mitigate free-riding, a long-standing issue in open source [59]. While free-riding is often understood in terms of lack of contributions on the part of users, we argue that when community members, both with formal power and no formal power, do not hold themselves accountable to "good-faith" empathy-based community engagement, they exhibit, both internal and external project members alike, a free-rider tendency for taking the community, their power, and their participation access for granted. In an ecosystem where most members were burned out navigating their work-related obligations and interpersonal conflict, these kinds of additional challenges could dis-incentivize individuals from feeling empathetic towards others and to assume their role as a responsible community member. We argue for moral accountability to, therefore, be framed in ways that help establish a culture of mutual trust, empathy, and empowerment.

The new initiative adopted by the community, referred to as "project goals initiative" seems to be accounting for individual responsibility towards work outcomes. Per this plan, each item on the community's project-level to-do list has a specific owner who is "inside the project [and] who has stepped up and said 'I really care about this, I am willing to be that person' " (P1 – Internal Project Member). Every item also has a mentor assigned who has more knowledge of the project (but does not necessarily have the time to work on it) and therefore will "iterate with [an owner who is] somebody less involved with the project" (P1 – Internal Project Member). These steps ensure a point of contact for following up on work and distributing the burden of tracking progress across those serving as owners. However, this mechanism does not seem sufficiently different from the roadmapping process in a way that could improve the cultural issues that continue to (a) impact members' ability to get work done; and (b) lead to their burn out. Per an interviewee, "Some of the things [on the project goals].. have been greatly delayed because of this [interpersonal conflict]" (P12 – Internal Project Member).

While it is difficult to ascertain how the new initiative might be more successful than the previous roadmapping iterations, we suspect that the mentorship model of this initiative could potentially begin to address some of the issues related to lack of empathy on external members' behalf. The mentorship model legitimizes an ongoing dialog between external members who are willing to do the work and internal members who have the knowledge and context to guide them. Apart from getting work done, it could address external members' concerns around transparency and visibility by affording them an opportunity to be mentored and socialized [1, 15, 20] into internal processes of the project by engaging with specific members of the project.

Overall, these findings offer a perspective on micro-level processes that contribute to the development of common ground across conflicting sets of values of private firms and volunteer contributors in open source communities relying on hybrid models. For example, Bonaccorsi and Rossi [6] compare the motivations of volunteer contributors with those of private firms and argue that firms care less about learning opportunities, reputation, code reciprocation, and the idea of free software. West and Gallagher [62] point out that the incentives for businesses to participate in open-source projects include sharing the cost of innovation and creating demand for their complementary products or services. These arguments about conflicting values center the importance of the open source product for firms and volunteers. We argue that shifting this focus from product to process not only highlights that firms and volunteers have a common need, i.e., to sustain the community, but also suggests that cultivating healthy participatory processes serves these common needs. Prior literature on toxicity in open source demonstrates that lack of healthy participatory processes negatively impact community's sustainability [42]. Henderson [25] argues that prioritizing prosocial goals can facilitate monetary incentives and "long term viability" of firms. This paper serves as one of the entry points to this discussion on the importance of participatory work processes and moral accountability in open source software communities.

7 Conclusion

The title of our paper, "Drinking from a Firehose while Herding Cats" was obviously intended to conjure up an amusing image. But we also think it succinctly and fairly precisely summarizes the two biggest themes of the paper that make accountability extraordinarily difficult in an open source context. Internal project members are indeed drinking from a firehose, as they are on the receiving end of a flood of requests for fixes and features which far exceeds what the community of limited resources can actually do. And there is little in the way of effective mechanisms for reducing the flow to manageable proportions. The project is also engaged in what is commonly referred to as cat herding, as members serving on the core team and as team leads try to rally an essentially independent-minded group toward common goals. Software developers, operating in a culture of open source, often tend not to respond well to herding. The new goal and ownership process is intended to address these problems by allowing teams to accept or reject goals, thereby managing their resources and taming the fire hose, and assigning ownership and thereby evoking commitment to them. We are hopeful for the success of the new approach, and curious to see the result.

Acknowledgments

This work is funded by the National Science Foundation award #2317168 and #2317169.

References

[1] Sogol Balali, Igor Steinmacher, Umayal Annamalai, Anita Sarma, and Marco Aurelio Gerosa. 2018. Newcomers' barriers... is that all? an analysis of mentors' and newcomers' barriers in OSS projects. *Computer Supported Cooperative Work (CSCW)* 27 (2018), 679–714.

[2] Carliss Y Baldwin and Kim B Clark. 2006. The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Management Science* 52, 7 (2006), 1116–1127.

[3] Caitlin Bentley. 2017. An analysis of accountability concepts for open development. In *Information and Communication Technologies for Development.* Springer International Publishing, Cham, 793–802.

[4] Tanmay Bhowmik, Nan Niu, Prachi Singhania, and Wentao Wang. 2015. On the Role of Structural Holes in Requirements Identification: An Exploratory Study on Open-Source Software Development. *ACM Trans. Manage. Inf. Syst.* 6, 3 (Sept. 2015), 1–30.

[5] Knut Blind, Mirko Böhm, Paula Grzegorzewska, Andrew Katz, Sachiko Muto, Sivan Pätsch, and Torben Schubert. 2021. The impact of Open Source Software and Hardware on technological independence, competitiveness and innovation in the EU economy. *Final Study Report. European Commission, Brussels* 10 (2021), 430161.

[6] Andrea Bonaccorsi and Cristina Rossi. 2006. Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business. *Knowl. Technol. Policy* 18, 4 (Dec. 2006), 40–64.

[7] Marcus Alphons Petrus Bovens. 1998. *The quest for responsibility: Accountability and citizenship in complex organisations.* Cambridge university press.

[8] Virginia Braun and Victoria Clarke. 2012. Thematic analysis. In *APA handbook of research methods in psychology, Vol 2: Research designs: Quantitative, qualitative, neuropsychological, and biological.* Vol. 2. American Psychological Association, Washington, 57–71.

[9] George Caffentzis and Silvia Federici. 2014. Commons against and beyond capitalism. *Community Development Journal* 49, suppl_1 (2014), i92–i105.
[10] Victoria Clarke and Virginia Braun. 2017. Thematic analysis. *J. Posit. Psychol.* 12, 3 (May 2017), 297–298.
[11] 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 2012 conference on Computer Supported Cooperative Work (CSCW '12)*. Association for Computing Machinery, New York, NY, USA, 1277–1286.
[12] Linus Dahlander and Mats G Magnusson. 2005. Relationships between open source software companies and communities: Observations from Nordic firms. *Research Policy* 34, 4 (2005), 481–493.
[13] Shay David. 2005. Opening the Sources of Accountability. *First Monday* (Nov. 2005).
[14] Dany Di Tullio and D Sandy Staples. 2013. The governance and control of open source software projects. *Journal of Management Information Systems* 30, 3 (2013), 49–80.
[15] Nicolas Ducheneaut. 2005. Socialization in an open source software community: A Socio-technical analysis. *Comput. Support. Coop. Work* 14, 4 (Aug. 2005), 323–368.
[16] Berrin Erdogan, Raymond T Sparrowe, Robert C Liden, and Kenneth J Dunegan. 2004. Implications of organizational exchanges for accountability theory. *Human Resource Management Review* 14, 1 (March 2004), 19–45.
[17] Pernille Eskerod. 2020. A stakeholder perspective: Origins and core concepts. In *Oxford Research Encyclopedia of Business and Management*.
[18] Dwight D Frink, Angela T Hall, Alexa A Perryman, Annette L Ranft, Wayne A Hochwarter, Gerald R Ferris, and M Todd Royle. 2008. Meso-level theory of accountability in organizations. In *Research in Personnel and Human Resources Management*. Emerald, 177–245.
[19] D Gopal and K Lyytinen. 2017. Effects of Social Structures in Requirements Quality of Open Source Software Project Development. (2017).
[20] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie Van Deursen. 2015. Work practices and challenges in pull-based development: The integrator's perspective. In *International Conference on Software Engineering (ICSE)*, Vol. 1. IEEE, 358–368.
[21] Shane Greenstein and Frank Nagle. 2014. Digital dark matter and the economic contribution of Apache. *Research Policy* 43, 4 (2014), 623–631.
[22] F S Grodzinsky, K Miller, and M J Wolf. 2003. Ethical issues in open source software. *Journal of Information, Communication and Ethics in Society* 1, 4 (Jan. 2003), 193–205.
[23] Jeffrey S Harrison, Jay B Barney, R Edward Freeman, and Robert A Phillips. 2019. *The Cambridge handbook of stakeholder theory*. Cambridge University Press.
[24] J Piet Hausberg and Sebastian Spaeth. 2020. Why makers make what they make: motivations to contribute to open source hardware development. *R D Manag.* 50, 1 (Jan. 2020), 75–95.
[25] Rebecca Henderson. 2023. Moral firms? *Daedalus* 152, 1 (Feb. 2023), 198–211.
[26] Eric von Hippel and Georg von Krogh. 2003. Open source software and the "private-collective" innovation model: Issues for organization science. *Organization Science* 14, 2 (2003), 209–223.
[27] Manuel Hoffman, Frank Nagle, and Yanuo Zhou. [n. d.]. The Value of Open Source Software. ([n. d.]).
[28] Hafiza Maria Kiran and Zulfiqar Ali. 2018. Requirement elicitation techniques for open source systems: a review. *International Journal of Advanced Computer Science and Applications* 9, 1 (2018).
[29] Laurie S Kirsch. 1997. Portfolios of control modes and IS project management. *Information Systems Research* 8, 3 (1997), 215–239.
[30] Daniel Klug, Christopher Bogart, and James D Herbsleb. 2021. "They Can Only Ever Guide" How an Open Source Software Community Uses Roadmaps to Coordinate Effort. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–28.
[31] RE Kraut. 2012. *Building Successful Online Communities: Evidence-based Social Design*. MIT Press.
[32] Johanna Kujala, Sybille Sachs, Heta Leinonen, Anna Heikkinen, and Daniel Laude. 2022. Stakeholder engagement: Past, present, and future. *Bus. Soc.* 61, 5 (May 2022), 1136–1196.
[33] Jaison Kuriakose and Jeffrey Parsons. 2015. How do open source software (OSS) developers practice and perceive requirements engineering? An empirical study. In *International Workshop on Empirical Requirements Engineering (EmpiRE)*. IEEE, 49–56.
[34] Paula Laurent and Jane Cleland-Huang. 2009. Lessons learned from open source projects for facilitating online requirements processes. In *International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*. Springer, 240–255.
[35] J Lerner and J Tirole. 2002. Some simple economics of open source. *J. Ind. Econ.* (2002).
[36] Josh Lerner and Jean Tirole. 2003. Some simple economics of open source. *J. Ind. Econ.* 50, 2 (March 2003), 197–234.
[37] Chan Li, Gary F Peters, Vernon J Richardson, and Marcia Weidenmier Watson. 2012. The Consequences of Information Technology Control Weaknesses on Management Information Systems: The Case of Sarbanes-oxley Internal Control

Reports. *Miss. Q.* 36, 1 (2012), 179–203.

[38] Juho Lindman, Matti Rossi, and Anna Paajanen. 2011. Matching Open Source Software Licenses with Corresponding Business Models. *IEEE Softw.* 28, 4 (July 2011), 31–35.

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

[40] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. 2013. Impression formation in online peer production: activity traces and personal profiles in github. In *Proceedings of the 2013 conference on Computer supported cooperative work (CSCW '13)*. Association for Computing Machinery, New York, NY, USA, 117–128.

[41] Vishal Midha and Sandra Slaughter. 2011. Mitigating the effects of structural complexity on open source software maintenance through accountability. In *ICIS 2011 Proceedings*. AIS eLibrary.

[42] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian KaUstner. 2022. Did you miss my comment or what?. In *Proceedings of the 44th International Conference on Software Engineering*. ACM, New York, NY, USA, 710–722.

[43] R K Mitchell, B R Agle, and D J Wood. 1997. Toward Theory Stakeholder Identification Salience: Defining Principles Who What Really Counts. *Academy Management Review* 22 (1997), 853–886.

[44] 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.

[45] Bianca M Napoleao, Fabio Petrillo, and Sylvain Halle. 2020. Open source software development process: A systematic review. In *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE, 135–144.

[46] John Noll. 2008. Requirements acquisition in open source development: Firefox 2.0. In *Open Source Development, Communities and Quality: IFIP 20 th World Computer Congress, Working Group 2.3 on Open Source Software, September 7-10, 2008, Milano, Italy 4*. Springer, 69–79.

[47] Siobhán O'Mahony. 2003. Guarding the commons: how community managed software projects protect their work. *Res. Policy* 32, 7 (July 2003), 1179–1198.

[48] Siobhán O'mahony and Fabrizio Ferraro. 2007. The emergence of governance in an open source community. *Academy of Management Journal* 50, 5 (2007), 1079–1106.

[49] William Foster Owen. 1984. Interpretive themes in relational communication. *Quarterly Journal of Speech* 70, 3 (1984), 274–287.

[50] Bidhan L Parmar, R Edward Freeman, Jeffrey S Harrison, Andrew C Wicks, Lauren Purnell, and Simone De Colle. 2010. Stakeholder theory: The state of the art. *Academy of Management Annals* 4, 1 (2010), 403–445.

[51] Jon Perr, Melissa M Appleyard, and Patrick Patrick. 2010. Open for business: emerging business models in open source software. *Int. J. Technol. Manag.* 52, 3/4 (Jan. 2010), 432–456.

[52] Stephanie Riger and Rannveig Sigurvinsdottir. 2016. Thematic analysis. *Handbook of methodological approaches to community-based research: Qualitative, quantitative, and mixed methods* (2016), 33–41.

[53] John Roberts. 1991. The possibilities of accountability. *Accounting, Organizations and Society* 16, 4 (1991), 355–368.

[54] John Roberts and Robert Scapens. 1985. Accounting systems and systems of accountability — understanding accounting practices in their organisational contexts. *Account. Organ. Soc.* 10, 4 (Jan. 1985), 443–456.

[55] Walt Scacchi. 2009. Understanding requirements for open source software. In *Design Requirements Engineering: A Ten-Year Perspective*. Springer, 467–494.

[56] Sonali K Shah. 2006. Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science* 52, 7 (2006), 1000–1014.

[57] Maliha Tasnim, Maruf Rayhan, Zheying Zhang, and Timo Poranen. 2023. A Systematic Literature Review on Requirements Engineering Practices and Challenges in Open-Source Projects. In *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 278–285.

[58] Radu E Vlas and William N Robinson. 2012. Two Rule-Based Natural Language Strategies for Requirements Discovery and Classification in Open Source Software Development Projects. *Journal of Management Information Systems* 28, 4 (April 2012), 11–38.

[59] Eric von Hippel. 2005. Open Source Software Projects as "User Innovation Networks". In *Perspectives on Free and Open Source Software*. The MIT Press, 267–278.

[60] Barry Warsaw, Łukasz Langa, Antoine Pitrou, Doug Hellmann, and Carol Willing. [n. d.]. PEP 8002 – open source governance survey. https://peps.python.org/pep-8002/. accessed Oct 2024.

[61] Richard T Watson, Marie-Claude Boudreau, Paul T York, Martina E Greiner, and Donald Wynn. 2008. The business of open source. *Commun. ACM* 51, 4 (April 2008), 41–46.

[62] Joel West and Scott Gallagher. 2006. Challenges of open innovation: the paradox of firm investment in open-source software. *R&D Management* 36, 3 (2006), 319–331.

[63] C G Wu, J H Gerlach, and C E Young. 2007. An empirical analysis of open source software developers' motivations and continuance intentions. *Information & Management* (2007).

[64] Yunwen Ye and Kouichi Kishida. 2003. Toward an understanding of the motivation of open source software developers. In *International Conference on Software Engineering (ICSE)*. IEEE, 419–429.
[65] C Zhang, J Hahn, and P De. 2013. Research note—Continued participation in online innovation communities: does community response matter equally for everyone? *Information Systems Research* (2013).

A Rust Roadmaps

Table 4. Primary differences across the five Rust roadmaps.

Roadmap	Summary	Roadmap items
#1	• 10 goals on the roadmap • No call for blog posts • Tracking of roadmap items • Included team-level roadmap items were a part of the project-level roadmap • Roadmap announced	• Rust should have a lower learning curve. • Rust should have a pleasant edit-compile-debug cycle. • Rust should provide a solid, but basic IDE experience. • Rust should provide easy access to high quality crates. • Rust should be well-equipped for writing robust, high-scale servers. • Rust should have 1.0-level crates for essential tasks. • Rust should integrate easily into large build systems. • Rust's community should provide mentoring at all levels. Other goals • Integration with other languages, running the gamut from C to JavaScript. • Usage in resource-constrained environments.
#2	• 4 goals and 4 target domains were listed on the roadmap. Each of the four target domains led to creation of 4 working groups. • 100 blog posts • No tracking of roadmap items was not done • Included team-level roadmap items were a part of the project-level roadmap • Roadmap announced	• Ship an edition release: Rust 2018. • Build resources for intermediate Rustaceans. • Connect and empower Rust's global community. • Grow Rust's teams and new leaders within them. Target domains to achieve these goals: • Network services • WebAssembly • CLI apps • Embedded devices
#3	• 3 goals on the roadmap that were broader and high-level • 73 blog posts • No tracking of roadmap items • Included team-level roadmap items were included in the project-level roadmap • Roadmap announced	• Governance: improving how the project is run • Finish long-standing requests: closing out work we've started but never finished • Polish: improving the overall quality of the language and tooling
#4	• 3 high-level goals on the roadmap • 74 blog posts • No tracking of roadmap items • Team-level roadmap not included in the project-level roadmap. • Roadmap not announced but a roadmap RFC was created and merged	• Prepare for a possible Rust 2021 Edition • Follow-through on in-progress designs and efforts • Improve project functioning and governance:
# 5	• 3 high-level goals • 10 blog posts • No tracking of roadmap items • Team-level roadmap not included in the project-level roadmap. • No technical items on the roadmap • Roadmap not announced but a roadmap RFC was created and merged	• Establishing charters for teams in the Rust project. • Provide for unified process and vocabulary across the project. • Creating a single place for tracking a list of ongoing projects.

Received October 2024; revised April 2025; accepted August 2025