Beyond Adoption: Examining the Evolution and Impact of Codes of Conduct on Open Source Communities

Jiayi Sun
University of Toronto

Hongbo Fang
University of Chicago

Junming Zhang
University of Toronto

Jiakai Shi
University of Toronto

Ruitao Lai
University of Toronto

Anita Ihuman
CHAOSS

Richard Littauer
Victoria University of Wellington

Shurui Zhou
University of Toronto

Abstract

While open source software (OSS) communities thrive on collaboration, conflicts such as toxic behavior and discrimination can surface, threatening project sustainability. To address these concerns, many communities have adopted a Code of Conduct (CoC) as a mechanism for governing and moderating members' behavior. Prior research has explored the motivations behind CoC adoption and their content updates, but it remains unclear how different communities are using and maintaining CoCs over time after initial adoption and how adoption impacts communities. To bridge these gaps, in our study, we compile a large-scale dataset of CoCs along with their change histories from GitHub to quantitatively (1) understand the evolution of CoC content and identify change patterns across different communities, and (2) investigate the potential impact of CoC adoption on community engagement. Our results show that OSS communities with a CoC attract more new contributors in both the short- and long-term. However, CoC adoption also leads to a short-term increase in disengagement among existing contributors, with very limited effects on long-term disengagement. The insights from this study can provide guidance for maintaining CoCs and offer statistically significant evidence of their impact.

CCS Concepts

• **Human-centered computing** → **Open source software**; • **General and reference** → **Empirical studies**.

ACM Reference Format:

Jiayi Sun, Hongbo Fang, Junming Zhang, Jiakai Shi, Ruitao Lai, Anita Ihuman, Richard Littauer, and Shurui Zhou. 2026. Beyond Adoption: Examining the Evolution and Impact of Codes of Conduct on Open Source Communities. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3744916.3773268

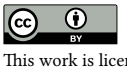
1 Introduction

With the growing popularity of social coding platforms like GitHub, more developers are now contributing to open source software (OSS) [14]. As communities grow, conflicts could emerge, such as toxic and discriminatory speech, entitlement, and aggression [51]. According to a survey conducted by GitHub in 2017, cumulatively around 17% of contributors have experienced or witnessed serious incidents, such as sexual harassment, stalking, or doxxing [29]. Conflicts can result in an unhealthy community culture, developers abandoning projects, and an increased workload for already overworked maintainers, leading to a lack of sustainability for OSS projects [43]. Given the critical role of OSS in modern digital infrastructure, fostering welcoming communities could have a substantial impact on software development and the overall OSS ecosystem, which is crucial to global digital infrastructure.

To address conflicts, many OSS communities have adopted a Code of Conduct (CoC), a form of project governance. It can "*define community standards, signal a welcoming and inclusive project, and outline procedures for handling abuse*" [30]. It typically specifies expected behaviors from community members and consequences for violations of the CoC. OSS communities either craft their own CoC, or adopt existing templates and modify them for their own use, such as the `Contributor Covenant` [23], a widely-used CoC template that outlines standards for respectful and inclusive behavior among contributors. Similar moderation practices, including the establishment of community guidelines and mechanisms for reporting violations, are found in other online communities like Reddit [5].

CoCs were originally adopted by projects to improve the psychological and social well-being of contributors, which is a critical part of intrinsic ethical values [55], and to improve the long-term sustainability of OSS communities [77]. Despite the effort of adopting CoCs to address conflict and foster healthy communities, there are still existing challenges with CoCs in OSS communities, such as a lack of enforcement, as most CoCs do not outline an effective way to ensure compliance of contributors [43]. Also, many contributors and maintainers are against adopting CoCs in their community [43], holding that the value of meritocracy should be prioritized over community culture. Many consider improving Equity, Diversity, and Inclusion (EDI) to be a liberal political ideology [15, 43, 67], which can be polarizing to some community members. At the same time, while some existing work shows no statistically significant change in community diversity with a CoC [62], some studies show otherwise [69].

Moreover, prior research has qualitatively explored the motivations behind CoC adoption [69], the elements shaping CoC content, and the challenges of enforcement [43, 77]. Researchers have also investigated the impact of CoC adoption, particularly on diversity, but findings remain inconclusive, with some reporting positive effects while others find no statistically significant changes [17, 62, 69]. Therefore, the effectiveness of CoCs remains an open question, and there is no large-scale statistical evidence on the impact of CoC



ICSE '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2025-3/2026/04
https://doi.org/10.1145/3744916.3773268

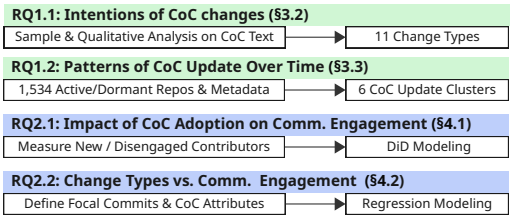


Figure 1: Overview of research method for each RQ.

adoption and updates on community engagement. Furthermore, how CoCs evolve alongside community growth and the nature of changes over time has yet to be systematically examined.

To address this research gap, we systematically investigate the impact of CoCs on OSS communities with large-scale CoC history data. Specifically, we seek to answer the following research questions (RQs). Fig. 1 provides an overview of the study design and analysis steps across RQs.

- **RQ1: How do CoCs evolve in open-source communities?** We address RQ1 by analyzing **why CoCs are updated (RQ1.1)** and **how they are maintained over time (RQ1.2)**. We compile a large-scale dataset of CoC adoption and update activity across OSS projects and use it to qualitatively analyze CoC changes, identifying 11 change intentions (RQ1.1). We then analyze update histories from 1,534 repositories to identify six clusters of update patterns (RQ1.2).
- **RQ2: What is the impact of CoC on community engagement?** We address RQ2 by examining **how CoC adoption affects contributor engagement (RQ2.1)** through a difference-in-differences (DiD) method [13, 36], and further investigate **how CoC updates relate to changes in engagement (RQ2.2)**. Our results show that communities with a CoC attract more new contributors in both the short- and long-term, while also demonstrating an increase in existing contributors disengaging in the short-term, with limited long-term effects (RQ2.1). Additionally, for RQ2.2, most CoC changes show minimal short-term correlation with newcomer growth, some positive correlations over the mid-term (6 months), and predominantly small negative correlations in the long-term (12 months).

In sum, our findings offer insights into the short- and long-term effects of CoC adoption on contributor engagement, laying the groundwork for future research on CoC effectiveness and its impact on diverse contributor demographics in OSS communities.

2 Related work

2.1 Opinions on CoCs from OSS Communities

Many sources highlight the positive impacts of CoCs in OSS communities [23, 38, 53]. These documents often emphasize the role of CoCs in diffusing tensions and formalizing a less restrictive, more guideline-focused approach to communication [70, 71]. Several articles and blog posts from OSS practitioners advocate for CoCs as tools for education on proper conduct in OSS communities and as a way to clarify acceptable behaviors, thereby reducing incidents of misconduct [40, 50]. They also underline the importance of inclusivity and the effectiveness of CoCs in handling workplace harassment [7, 80]. Conversely, online discussions [59, 60] have voiced opposition to CoCs, with some advocating for a "No CoC" stance, a pushback against CoC adoption in community moderation. There are also opinions critiquing CoCs, particularly focusing on their potential to oppress rather than protect community members [44, 54], arguing that CoCs can be misused by maintainers to restrict contributions or suppress dissent, cultivating an atmosphere of fear rather than open dialogue [41]. Furthermore, the effectiveness of widely adopted CoCs, such as the Contributor Covenant[23], has been questioned, with concerns that they lack tangible evidence of fostering inclusivity and are often used more as a marketing tool than to effectually improve community health [26].

2.2 Previous Research On CoCs

Qualitative studies on CoC adoption in OSS. Previously, Tourani et al. [77] conducted interviews with OSS project maintainers to identify common elements and the motivations and processes behind their implementation of CoCs. Li et al. [43] investigated why OSS communities adopt CoCs and what aspects they address, by analyzing GitHub issue discussions. They found that CoCs are primarily introduced to prevent harassment and foster a more inclusive environment, with common content areas including language, behavior, and participation guidelines. Frluckaj et al. [25] conducted interviews with contributors from minority groups and found that CoCs can serve as signals of safety, helping these contributors decide whether to join a project. Their study also discussed the paradoxes between the openness in OSS and having effective governance structures to manage community members' behavior. The lack of effective enforcement remains a challenge [43], raising concerns about the potential harm of CoC violations to contributor well-being and community sustainability, as CoCs may signal a "false sense of safety" [25]. When enforcing CoCs, resistance can also arise from community members, often rooted in the belief that meritocracy should take precedence over considerations of EDI. Some opponents perceive EDI efforts as political or unnecessary, particularly when they involve inclusivity toward marginalized groups [43]. Additionally, opinions on the overall effectiveness of CoCs in OSS communities remain divided [15, 43, 67].

Quantitative studies on CoC's impact. Researchers have also investigated the impact of CoCs. For instance, Singh et al. [69] studied OSS project websites and analyzed online discussions among women and found that CoCs can help to improve team diversity in OSS communities, while Robson [62] analyzed the 500 most popular GitHub projects and found that there was no statistically significant change in team diversity as a result of CoC adoption. Additionally, Enache [17] investigated the adoption of CoCs with women's participation in 149 OSS communities and found no statistically significant results to indicate there is an impact.

Tooling support for CoC enforcement. To address enforcement inefficiencies, Cobos and Izquierdo [8] proposed a bot-based approach that automates defining, monitoring, and enforcing CoCs in OSS projects, leveraging ethical guidelines derived from the Contributor Covenant to support community moderation.

While prior studies have identified the motivations for creating CoCs and the types of content updates made, the different maintenance practices of CoCs over time and their impact on communities remain unclear. Therefore, in this study, we aim to first understand

the maintenance of CoCs post adoption over time in OSS communities and then we would like to quantitatively investigate the implementation of CoCs and estimate their impact on communities.

3 (RQ1) CoC Adoption and Update Practices

To examine CoC adoption and its impact, we first compile a comprehensive dataset of OSS repositories that have adopted CoCs and their associated historical changes. We then analyze the evolution of CoCs over time and identify update practices within OSS communities.

3.1 Identifying Projects with CoCs Adopted

3.1.1 ***Data Collection.*** We use a four-step approach to collect repositories that have adopted a CoC.

Step 1: Identifying repositories with potential CoC adoption. We use the GitHub API [34] to search for issues, pull requests (PRs), and commit messages containing the case-insensitive phrase "Code of Conduct". While effective, this method may mistakenly include cases where a CoC is discussed but not adopted, as it only matches the string mentioned, such as in *acassen/keepalived*, PR#2045, where a proposed CoC adoption was rejected by the community. Also, repositories under the same owner account may not explicitly discuss CoC adoption in each repository. For example, while we identify *yahoo/elide* via PR#732 mentioning a CoC, *yahoo/YMTreeMap*, which also has a CoC, is overlooked due to the absence of such references. This suggests that many repositories under the same organization account can have CoCs, but cannot be detected through the approach of matching keywords in issue/PR discussions. To address this, we extend our dataset to include all non-fork repositories under accounts with at least one CoC-related match, yielding 3,844,209 repositories from 153,579 accounts (as of June 2024).

Step 2: Validating CoC presence. To ensure high precision, we examine **two** key locations in each repository:

- **Community profile.** GitHub provides a checklist to help maintainers include community health files such as CoCs, `LICENSE`, or `CONTRIBUTING.MD` [33]. We use the GitHub API to retrieve each repository's community profile and verify whether the `code_of_conduct_file` field is `non-null`, indicating a linked CoC file–either stored locally (e.g., *bert/libdxf* [31, 32]) or externally (e.g., *microsoft/vscode* [47–49]).
- **Root folder.** If absent from the community profile, we search the root folder for filenames containing case-insensitive variations of "code of conduct" (e.g., "coc").

Step 3: Filtering non-popular projects. We exclude repositories with zero forks or stars, as these signal public interest [3]. This ensures a meaningful analysis of CoC usage in OSS communities.

Step 4: Collecting metadata for each repository and its CoC file. We extract repository metadata (e.g., commits and age) to assess project activity and engagement. In addition, we analyze CoC files by retrieving their commit history (e.g., lines changed) to track changes over time. As outlined in **Step 2**, CoC files can be hosted internally or linked to a centralized CoC within an organization. To avoid redundancy, we identify unique CoC files based on full file paths, providing a scalable approach; although it may treat duplicated content in different directories as distinct, it avoids the complexity and computational overhead of large-scale textual comparison, which could miss subtle but meaningful CoC edits. This results in 93,501 unique files across 158,735 repositories linked to 126,203 commits. Using the GitHub API, we examine file attributes such as size and the presence of email or external links, which can be indicators of enforcement mechanisms [17, 43, 69].

3.1.2 ***Data Overview.*** Across the 158,735 repositories, each has an average of 1.42 CoC-related commits, with a maximum of 149, seen in *randsleadershipslack/documents-and-resources*. Over 75% of repositories have only one commit, suggesting most CoCs are rarely revised post-adoption.

Location and hosting of CoC files. 92,034 repositories (58%) host CoC files directly, while 66,701 link to CoCs in external repositories via their community profile. Interestingly, 1,607 CoC files are shared across multiple repositories, indicating some level of standardization or reuse within the community. For example, the account *sunpy* [74] links the same CoC file [73] in 32 repositories, and *alibaba* [1] hosts separate CoC files across 38 repositories. In *rust-embedded* [65], 18 repositories link to a centralized CoC in *rust-embedded/.github* [64], whereas 39 others host their own CoC files, reflecting diverse management practices.

Characteristics of CoC Files Content. We observed that 97.3% of the CoC files include at least one link to an external site that offers additional CoC-related information. while 72.1% of the CoC files (68.4% repositories) have email contact information. Among the CoCs without contact information, 94.1% of them contain links to external websites. For instance, *OctoPrint/OctoPrint* project includes only a single sentence in its CoC file, providing a link to the complete CoC on the official website [56].

3.2 (RQ1.1) Intentions of CoC Changes

3.2.1 ***Sampling a Subset of CoC Files for Qualitative Analysis.*** To investigate the motivations behind CoC modifications, we qualitatively analyze the change history of CoC files. Unlike prior work that relied on random issue samples [43] or practitioner interviews [77], our study systematically collects all CoC file changes over a project's lifespan. This approach facilitates a comparative analysis of trends across communities to uncover evolving norms and practices (RQ1.2). As manual review of all 93k files is infeasible, we apply the following inclusion criteria (IC):

- **IC1: One CoC per account.** To avoid redundancy, for each account, we select the repository with the highest number of CoC commits to represent it, forming *account/repo* pairs.
- **IC2: CoC files with substantial update activity**. To focus on sustained maintenance, we include CoC files with commit counts above the 90th percentile (>3 commits).
- **IC3: Content-modifying CoC commits.** We select `Modified` CoC commits, indicating content changes (added/deleted lines), rather than file-level renames or deletions.

Applying the three **ICs** sequentially, we identify 1,716 *account/repo* pairs, with corresponding CoC files and associated 10,628 CoC commits, where 7,986 of them are `Modified` commits.

3.2.2 ***Identifying Intentions of CoC Changes.*** The analysis process consists of two steps, as illustrated in Fig. 2.

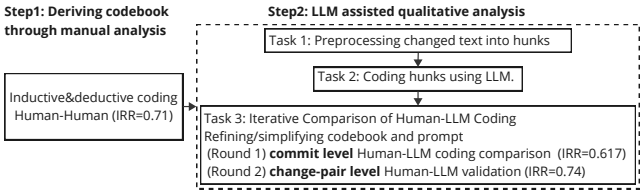


Figure 2: Workflow analyzing commit changes

Step 1: Deriving the codebook with inductive & deductive coding. We extend prior codes [43, 77] using both inductive and deductive coding [21]. First, two authors independently code 30 CoC commits from a selected repository *nf-core/website*, reviewing full CoC files and comparing line changes to group text chunks by intent and assign codes. For each commit, we review the entire CoC file. There is no fixed granularity for chunk size, so consecutive sentences may have different codes, while consecutive paragraphs may share the same code. Single commits may also contain multiple codes. Next, two authors randomly select three more repositories (i.e., *apache/superset*, *InsightSoftwareConsortium/ITK*, and *MDAnalysis/mdanalysis*), and code all 38 commits and compare their results to update the codebook, achieving a Cohen's Kappa of 0.71 [46], indicating substantial agreement.

Step 2: Leveraging LLMs for coding a large number of commits. To scale the analysis of 7,986 CoC commits, we leverage GPT-4 Turbo [57], drawing inspiration from prior work on large language models (LLMs) for qualitative analysis [22, 81]. This step replicates the manual process through three structured tasks (**T**):

T1: *Preprocessing changed text into hunks per commit.* As noted in Step 1, manual analysis involves grouping text by intent and assigning codes. In testing the LLM's coding performance, we observe that analyzing entire changed files often results in missed or inaccurate codes. To address this, we segment the changed files into `hunks`, a contiguous block of changes in a file's "`Git diff`" [35]), rather than relying on the LLM to determine the chunks.

T2*: Coding hunks using the LLM.* We designed a two-step prompt for the LLM to process each hunk. First, it identifies coding units–either added/deleted text or modified text pairs. Then, it assigns each unit a corresponding code from the provided codebook. The full prompt is available in the replication package [72].

T3*: Iterative comparison of human-LLM coding.* After T2, we compare LLM and manual labels, calculating IRR for **two rounds**.

- **Round one**: We evaluate agreement at the **commit level**, using the full set of CoC-related commits from **Step 1**, yielding a Cohen's Kappa of 0.617 (acceptable agreement [46]). We address discrepancies by refining the codebook and prompts, merging overlapping codes like *Respectful* and *Unacceptable behaviors* into *Behavior specification* (Tab. 1), and then repeat T1 and T2.
- **Round two**: We perform a finer-grained validation at the **change-pair level**, focusing on individual CoC-related changes rather than commits. To ensure coverage of all 11 change types (see Tab. 1), we randomly sample one commit per type (11 commits, resulting in 76 change pairs) and manually label them, allowing us to assess coding consistency across types. The coded sample yields a Cohen's Kappa of 0.74 (moderate agreement [46]). Most discrepancies are from large location shifts in `git diff` and missing context. With the finalized codebook, we then code the remaining 7.9k commits with the LLM.

3.2.3 ***Result.*** Tab. 1 summarizes all codes and their occurrence across the sampled repositories. We reused eight codes from prior studies [43, 69, 77], extended the "*Non-content changes*" code to include updates like translations or wording clarifications that do not alter the original intent, and introduced three new codes (i.e., *Acknowledgement, Adaptability,* and *Justify the adoption of CoC*) to capture previously unaddressed intentions. Among 1,716 sampled repositories, we found 21,745 CoC change type occurrences in `Modified` commits. On average, each repository exhibited 3.78 change types (min=1, max=11) and 2 unique CoC file authors.

Among all update types, *Non-content changes* were most common, appearing in 4,469 commits across 1,551 repositories (>90%), typically improving clarity and formatting. *Availability* updates followed (1,934 commits, 1,086 repos), often updating contact information to enhance accessibility and responsiveness. *Behavior specification* updates appeared in 46.9% of repositories, clarifying norms and expectations. *Enforcement* (42.8%) and *Scope* updates (34.6%) were also frequent, while other types were less common.

Additionally, we analyzed the trend of changes over time and found that while *Non-content changes* remained the most frequent, *Availability*, *Enforcement*, and *Behavior specification* also increased, reflecting a shift toward clearer enforcement and reporting (See replication package [72] for details).

(RQ1.1) Insight. While non-content changes dominate, many projects revise CoCs to clarify norms and enforcement, suggesting a shift from symbolic adoption to intentional governance practices.

3.3 (RQ1.2) Patterns in CoC Update over Time

To examine how communities update their CoCs over time, we analyze the change history of CoC files alongside project metadata.

3.3.1 ***Selection of Repositories and Metadata.*** We use the dataset from Sec. 3.2.1, which includes 1,716 repositories with LLM-analyzed CoC commit histories. To interpret CoC update patterns, we manually categorized repositories by purpose (e.g., software development, resource collection) using inductive coding [66]. Two authors independently coded 40 repositories to develop an initial codebook, then refined it through discussion, achieving a Cohen's Kappa [10] of 0.82. One author then applied the finalized codebook to the full dataset, identifying five valid repository types while discarding invalid cases (e.g., *clone projects*), resulting in 1,651 repositories. Additionally, we examined the distribution of project age (calculated as $T_{LatestCommit} - T_{FirstCommit}$) and excluded repositories of less than 3 months (bottom 1%) to ensure sufficient project maturity for observable development activity. This filtering step yielded 1,534 repositories for analysis. Most are software development (992, 64.7%) and resource collection (496, 32.3%), while other types include *OSS template*, *CoC template*, and *CoC* repos (See codebook and replication package [72]).

To account for project activity status, we classify the 1,534 repositories as either ***dormant*** (no commits in the 12 months prior to June 2024, in total 568 repositories) or ***active*** (966 repositories). This helps contextualize update behaviors: dormant repositories reflect completed lifecycles, while active ones may exhibit evolving patterns, reducing potential misinterpretation from lifecycle heterogeneity within clusters.

Table 1: Codebook of 11 CoC Change Types. Reused codes from prior work are cited [42,72]; Extended codes are marked (+); #Cmt: the number of commits.

Code	Definition	#Cmts	#Repos
Non-content changes+ [43]	Add or update hyperlinks, reorganize text, translate content to another language, or format to improve readability, and eliminate vagueness through rephrasing and substituting terms to prevent misunderstandings. [E.g., link]	4469	1551 (90.4%)
Availability [43]	Update contact information (e.g., email addresses, names, and user accounts) for reporting offenses.[E.g., link]	1934	1086 (63.3%)
Behavior specification [77]	Update the definition of respectful or unacceptable behaviors, including general behavior expectations. [E.g., link]	1504	804 (46.9%)
Enforcement [77]	Define and refine the mechanisms for reporting and handling violations of the CoC, detailing the composition of the committee members and clarifying who is responsible for addressing CoC violations.[E.g., link]	1029	734 (42.8%)
Scope [77]	Define the online and offline spaces where the CoC applies, and identify the individuals who are subject to the provisions of the CoC. [E.g., link]	792	593 (34.6%)
Acknowledgement	Acknowledge the resource and the authorship of the CoC.[E.g., link]	726	588(34.3%)
Version update [43]	Many projects used popular, publicly available templates for their CoC. When these templates were routinely updated, contributors sometimes updated the version in their repository.[E.g., link]	586	387 (22.6%)
Inclusiveness refinement [43]	Define and refine the scope of inclusiveness.[E.g., link]	447	380 (22.1%)
Adaptability	Demonstrates a commitment to continuous improvement and responsiveness to the community's needs.[E.g., link]	222	176(10.3%)
Justify the adoption of CoC	Justify why the community adopted the CoC.[E.g., link]	135	121(7.1%)
Approachable [43]	Update to change words used carefully in the contributing guidelines document to make projects seem approachable and less intimidating to new contributors.[E.g., link]	73	71 (4.1%)

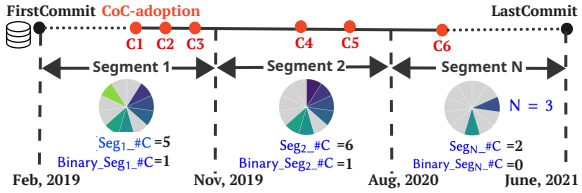


Figure 3: Example of a segmented repository lifespan, where CoC-related commits are marked as red dots. The lifespan is divided into three segments (N=3). Based on the coding results from RQ 2.1, the unique types of CoC updates are aggregated within each segment: Seg. 1 includes five types, Seg. 2 contains six, and Seg. 3 has two. Then $Seg\_Changes = [5, 6, 1]$ after Step 1 and $Binary\_Seg\_Changes = [1, 1, 0]$ after Step 2.

3.3.2 ***Identifying Patterns of CoC Changes over Time.*** To characterize how CoC files evolve, we iteratively developed a unified project representation based on three key factors: (1) the repository's CoC update trajectory, (2) project type, and (3) age. We design the three-step process (see RQ1.2 in replication package [72] for illustration of the process) for clustering repos as follows:

Step 1: Vectorization of repositories. To ensure a fair comparison across repositories with varying ages, we divide the lifespan of each repository evenly into a fixed number of N segments (denoted as $Seg_i, i = [1 : N]$). Further, for each segment Seg_i, we count the unique types of CoC changes (denoted as $Seg_i\_\#C \in [0, 11]$). Thus, the CoC change activities for each repository can be presented as: $Seg\_Changes = [Seg_1\_\#C, Seg_2\_\#C, \ldots, Seg_N\_\#C]$. We also use the project type (as described in 3.3.1) and project age as two other factors to represent each project. Therefore, for each project, we define the vector $V = [Seg\_Changes, Repo\_Type, Age]$ incorporating CoC type changes over time, repository type, and age.

Fig. 3 illustrates an example repository categorized as a *Software* project with a lifespan of 2 years and 4 months. When partitioning its lifespan into $N = 3$ segments, the resulting vector representing the CoC update pattern is $Seg\_Changes = [5, 6, 2]$. Consequently, the complete vector representation of this repository is given by $V = [5, 6, 2, \textit{Software}, \textit{2 years}]$.

Step 2: Feature dimensionality reduction for K-Means clustering. Since $Seg_i\_\#C$ ranges from 0-11 and $Repo\_Type$ ranges from 1-5, and, Age ranges from 0-12, features with a larger range may dominate the clustering process. To mitigate this imbalance and ensure equal weighting across factors, we normalize the range of each dimension. Specifically, we constrain $Seg_i\_\#C$ and Age to five discrete values, aligning them with the range of $Repo\_Type$. For Age, this transformation involves partitioning the original range (0–12) into five equally distributed intervals.

Regarding transferring $Seg_i\_\#C$ to a feature of five values, we considered running K-Means to obtain five clusters. However, we observed that the resulting clusters did not align with intuition, making it harder to explain the results. To prioritize interpretability, we applied a **threshold-based binarization approach** to encode $Seg_i\_\#C$ based on its empirical distribution. Since the distribution is long-tailed, we distinguish common versus high values of $Seg_i\_\#C$, preserving relevant information while reducing unnecessary granularity. A binary encoding simplifies the representation, making the clustering less sensitive to minor variations while still distinguishing meaningful trends. We selected a threshold at the **85th percentile** ($Seg_i\_\#C \geq 4$), encoding values as follows:

$$Binary\_Seg_i\_\#C = \begin{cases} 1, & \text{if } Seg_i\_\#C \geq 4 \text{ (major updates)} \\ 0, & \text{otherwise (minor updates)} \end{cases}$$

Binarizing at this threshold allows us to distinguish common versus high values of $Seg_i\_\#C$, preserving relevant information while reducing unnecessary granularity. Thus, the CoC change activities per repository can be further presented as: $Binary\_Seg\_Changes = [Binary\_Seg_1\_\#C, \ldots, Binary\_Seg_N\_\#C]$. In the example in Fig. 3, $Binary\_Seg\_Changes = [1, 1, 0]$ after Step 2.

We then decide the number of segments N for each project, forming 2^N repository categories. To balance detail and granularity, we set $N = 3$, segmenting each project's lifespan into early, mid, and late phases. This granularity allowed us to capture meaningful temporal variation in CoC update patterns without overfragmenting (e.g., $N = 4$ yields 16 patterns, many with very few instances, while $N = 2$ yields 4 patterns in total). We believe that $N = 3$ produced more interpretable and stable clustering results. Tab. 2 summarizes the threshold-based binarization results.

Our results show that among the 8 categories, only 94 (6.13%) repositories have active CoC updates in more than two segments (patterns 111, 011, 101, 110), while 535 (34.88%) have updates in a single segment (patterns 100, 010, 001). The majority, 905 (59%), show no updates (000). Thus, to simplify the analysis, we merge

Table 2: CoC Update Pattern of Repositories.

Group	Vec.	Category	Pattern	#Repos (%)
G1	111	**Consistent Major Updates**		13 (0.85)
	011	**Mid-Late Major Update**		34 (2.22)
	101	**Early-Late Major Update**		26 (1.69)
	110	**Early-Mid Major Updates**		21 (1.37)
G2	100	**Early Major Updates Only**		209 (13.62)
G3	010	**Mid-Stage Major Updates Only**		164 (10.69)
G4	001	**Late Major Updates Only**		162 (10.56)
G5	000	**Minimal or No Major Updates**		905 (59.0)

Note: Each color represents one type of CoC change during the time segment in repo.

the first four patterns into one group (G1) and retain the remaining four as G2-G4, yielding five discrete values (as shown in column **Group** in Tab. 2) for Seg_i_#C.

Step 3: K-Means clustering. We applied one-hot encoding to *Update pattern* (**G1-G5**), Repo_Type, and Age to obtain a 15-dimensional vector per repository for clustering.

3.3.3 ***Results: Clustering of the Repositories.*** We applied the silhouette score [63] to identify the optimal number of clusters. For all valid repositories sampled, $K = 6$ was the most suitable choice for the clustered data. In Tab. 3, we summarize the results, showing the predominant repository type and associated update patterns. Due to space limits, we highlight key characteristics of **Clusters 1, 4, and 5**. Full cluster details are in the replication package [72]. We emphasize that the examples provided are illustrative rather than exhaustive; a comprehensive analysis of the full evolution of CoC updates is beyond the scope of this work. Our focus is specifically on update patterns in relation to the repositories' lifespans. Future research could examine the content and contextual factors of CoC changes in greater depth.

Cluster 1 (Software-dominated, N=752) is the largest cluster, consisting predominantly of software projects, with five CoC-related repositories. Despite their diverse development trajectories, most projects in this cluster exhibit minimal engagement with their CoC after initial adoption. The **000-G5** (N=549) pattern dominates, indicating that for many software projects, the CoC is treated as a one-time requirement. For example, *DARIAEngineering/dcaf_case_management* (created in 2015) [11] introduced its CoC in 2017, and made all updates within a week, addressing enforcement, behavior specification, and reporting mechanisms. Notably, this pattern of limited post-adoption activity appears across both **active** and **dormant** projects, suggesting that CoC maintenance is not necessarily tied to overall project activity. These cases indicate that some projects may view the period immediately following CoC adoption as the appropriate time to make refinements, potentially aiming to establish a complete and self-sufficient document early on. The lack of subsequent updates could imply that, in the absence of external pressure or internal feedback, further revisions are not seen as necessary.

121 projects following the **100-G2** pattern show more early initiative. For instance, *unoplatform/uno* [79] introduced its CoC in 2018 and refined it over a year before stabilizing, suggesting an intentional effort to front-load updates and create a complete, self-contained document early on. Meanwhile, the **001-G4** pattern

Table 3: RQ1.2 Results: Six Repository Clusters Defined by Project Age, Project Type, and CoC Update Patterns.A = Active, D = Dormant

Cluster (#)	Repo type	#Repos	Group	#Repos	#A vs #D
1 (752)	CoC	2	G5	1	0 vs 1
			G4	1	0 vs 1
	CoC template	3	G5	3	1 vs 2
	Software	747	G5	549	374 vs 175
			G4	77	50 vs 27
			G2	121	76 vs 25
2 (423)	CoC	9	G4	2	0 vs 2
			G2	7	0 vs 7
	Resource collection	414	G5	292	136 vs 156
			G4	45	25 vs 20
			G2	77	37 vs 40
3 (150)	CoC	2	G3	2	0 vs 2
	Resource collection	40	G3	40	24 vs 16
	Software	108	G3	108	77 vs 31
4 (101)	Resource collection	5	G5	2	2 vs 0
			G4	2	2 vs 0
			G2	1	1 vs 0
	Software	96	G5	42	37 vs 5
			G4	31	30 vs 1
			G3	13	12 vs 1
			G2	1	1 vs 0
			G1	9	9 vs 0
5 (85)	CoC	5	G1	5	2 vs 3
	CoC template	2	G1	2	1 vs 1
	Resource collection	37	G1	37	27 vs 10
	Software	41	G1	41	31 vs 10
6 (23)	OSS template	23	G5	16	6 vs 10
			G4	4	2 vs 2
			G3	1	1 vs 0
			G2	2	2 vs 0

reflects delayed but significant updates. For example, *bootstrap-vue/bootstrap-vue* [2], an active software repository created in 2016, adopted Contributor Covenant v1.4 in 2017 and made only minor edits until upgrading to v2.1 in 2023, suggesting some projects revise CoCs in response to **external changes**, not ongoing maintenance.

Cluster 4 (Long-lived, actively maintained projects dominated, N=101) consists mostly of **mature, still-active software projects** with sustained maintenance. The dominant pattern is **000-G5** (N=42), where CoCs are adopted but rarely updated, suggesting that many projects treat them as a stable, one-time commitment. CoC adoption in this cluster predominantly occurred between 2015 and 2018, with a sharp rise in 2017–2018 (20 projects per year). This trend aligns with broader developments in the OSS ecosystem: the publication of the Contributor Covenant in 2014 [23] and GitHub's introduction of a one-click CoC template in 2016 [30] lowered the barrier to adoption and established it as an ecosystem community norm. Most projects in this cluster appear to have treated their CoC as a **stable, one-time commitment**, requiring little follow-up. However, not all projects follow this static trajectory. For example, *CocoaPods/CocoaPods* **(100-G2)** [9], was created in 2011 and adopted a CoC in 2015, introducing several updates that year (e.g., specifying behaviors, adding acknowledgments), followed by only a minor typo fix in 2019. In contrast, *elixir-lang/elixir* **(G1)** [16], continuously evolved their CoCs to reflect growing community scope and values–for example, changing *"project" to "ecosystem"* to signal inclusivity.

These contrasting examples suggest that even with long-lived active projects, CoC maintenance varies. Some projects adopt a "*set-it-and-forget-it*" mindset, while others treat the CoC as a living document that evolves alongside the project's mission and community norms. The latter approach may indicate a **proactive governance** style, emphasizing responsiveness and transparency in the face of growth and change.

Cluster 5 (Frequent CoC updates across types, N=85) includes a mix of repositories that **consistently engage in sustained CoC maintenance**, distinguishing it from clusters dominated by static or one-time update patterns. All follow the **Others-G1** pattern, with updates often driven by evolving governance needs, incidents, community feedback, or alignment with external standards. Notably, both active and dormant repositories appear in this group, with a significant share remain active (over 70%) across all types. Examples like *hackference/code-of-conduct* [39] and *EthicalSource/contributor_covenant* [18] demonstrate how CoCs evolve with community input, reflecting a collaboratively governed and responsive approach. Such practices are not limited to template repositories, but are also in software and resource collections repositories.

Overall, this cluster reflects a governance model that treats the CoC as a living document, regularly revised to align with shifting values, norms, and expectations.

(RQ1.2) Insight. CoC maintenance practices vary widely across communities. While most projects adopt a CoC with minimal follow-up, a subset engage in sustained or lifecycle-specific updates, reflecting different governance needs and responsiveness to community or external pressures. These patterns highlight that CoC evolution is shaped more by project dynamics than by repository type.

4 (RQ2) Impact of CoCs

In this section, we present a quantitative assessment of (1) the impact of *CoC adoption* on *community engagement* in OSS projects, as well as (2) the relationship between subsequent *CoC updates* and *changes in community engagement* following the initial adoption.

4.1 (RQ2.1) Impact of CoC Adoption on Community Engagement

4.1.1 ***Methodology: Causal Inference Using Difference-in-Differences (DiD).*** To examine the causal relationship between CoC adoption and community engagement, a simple comparison of engagement levels before and after CoC adoption may not account for other external factors that could influence engagement. Therefore, we employ **DiD**, a statistical method commonly used in econometrics and social sciences to estimate causal effects of interventions [13, 36]. This approach has been applied in software engineering to study the impact of external factors on OSS [20, 45].

In our study, the **treatment group** consists of projects that have adopted a CoC, while the **control group** comprises projects with comparable attributes that did not adopt a CoC (detailed in Sec. 4.1.3). If both groups display similar trends in community engagement before CoC adoption, we can reasonably infer that any **control variables** influencing engagement affect both groups equally. Thus, post-adoption divergence in engagement between the groups can be attributed to the CoC's influence, as DiD ensures alternative factors would affect both groups similarly. In Fig. 4, we present the timeline of a repository that adopted a CoC at a given time as *treatment time* (T). We define ΔT as the *length period of interest*. $T - \Delta T$ to T is the *pre-treatment period* (P_{pre}), and T to $T + \Delta T$ is the *post-treatment period* (P_{post}).

4.1.2 ***Measurement of the Outcome Variable for Community Engagement.*** In OSS where participation is voluntary, turnover can indicate ongoing community engagement. In this study, we operationalize turnover as (1) incoming new contributors and (2) disengaged existing contributors.

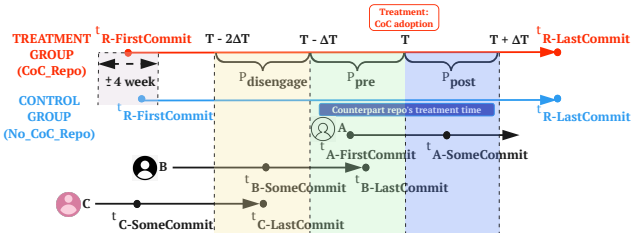


Figure 4: Illustration of treatment & control groups, community engagement.

We define **new contributors** as those who had no recorded commits prior to the start of the specified period (P_{pre} or P_{post}) and made at least one commit during the respective period. For example, in Fig. 4, contributor A would be counted as a new contributor during P_{pre}. We define **disengaged contributors** as individuals who actively contributed by making at least one commit within the period $T - 2\Delta T$ to $T - \Delta T$, denoted as the period $P_{disengage}$, but did not make any commits during the subsequent P_{pre} period, thereby indicating disengagement. For instance, in Fig. 4, contributor B made one commit during $P_{disengage}$ and their last commit during P_{pre}. Therefore, B is classified as a disengaged contributor during the P_{post}. Conversely, contributor C, whose last commit occurred within $P_{disengage}$, is considered disengaged during the P_{pre}.

4.1.3 ***Identifying the Control Group.*** DiD specification requires that repositories in the treatment and control groups have similar characteristics during the *pre-treatment period* (P_{pre}) so that the alternative factors that might have an impact on the outcome variables of interest will apply similarly to both the treatment and control group (*parallel trend assumption [4]*). However, in the context of OSS communities, we only have observational data, making it challenging to find a control group where the only difference between repositories is the presence of a CoC.

To address this challenge, we aim to identify a control group of repositories that have a comparable likelihood of adopting a CoC to our treatment group, yet have not adopted one. We select factors considered influential for the likelihood of CoC adoption and ensure that repositories in the control group match closely with the treatment group based on these factors.

Specifically, we define matching criteria (**MC**) to pair each repository in the treatment group (denoted as `CoC_Repo`), which has adopted a CoC, with one from the control group (`NO_CoC_Repo`), representing repositories with similar characteristics around the CoC adoption time in `CoC_Repo` , but did not adopt a CoC.

- **(MC1) Similar repository creation time**: Projects at different development stages may vary in their needs for adopting a CoC. Therefore, we require each `NO_CoC_Repo` to have been created within a 4-week window (either before or after) of the creation date of the corresponding `CoC_Repo`. This helps match repositories at similar development stages while keeping enough valid pairs.
- **(MC2) Primary programming language (PL)**: Different PL communities can have distinct cultural norms, ecosystems, and collaboration styles that may influence CoC adoption. To mitigate

these confounding factors, each control repository (NO_CoC_Repo) must share the same primary PL as its corresponding treatment repository (CoC_Repo).
- **(MC3) Comparable repository activity metrics**: Project visibility and engagement can impact the need for a CoC. Highly active repositories with larger contributor bases typically experience more complex community interactions, potentially prompting CoC adoption. Thus, we ensure that the cumulative number of stars, commits, and unique contributors in each NO_CoC_Repo at the CoC adoption time (T) of the matched CoC_Repo are within a 20% deviation of the corresponding values in CoC_Repo.
- **(MC4) Similar recent contributor growth trend**: To satisfy the parallel trend assumption between treatment and control groups required for DiD analysis, the trend in new contributors growth for each control repository (NO_CoC_Repo) in the weeks prior to CoC adoption (T) should deviate by no more than 10% from its matched CoC_Repo's trend. This trend is assessed by computing the ratios of new contributors across consecutive weeks prior to CoC adoption (T). Specifically, we calculate two ratios for the three weeks leading up to CoC adoption as follows: $Ratio_{3to2w} = \frac{\text{Week 2 count}}{\text{Week 3 count}}$ and $Ratio_{2to1w} = \frac{\text{Week 1 count}}{\text{Week 2 count}}$. Finally, we locate projects with similar trends.

For the choices of thresholds of these criteria, we experimented with stricter thresholds (e.g., 2- or 3-week time windows of repository creation time for **MC1** and tighter activity metric deviations, such as 5% to 15% for **MC2-4**), but these significantly reduced the number of matched control repositories. Looser thresholds, like a 5-week window or larger deviations, offered only marginal gains while increasing the risk of dissimilarity between treatment and control groups. We chose these thresholds to strike a balance between comparability and sufficient control group repository size.

Given practical constraints related to time and API limitations, we utilize GHTorrent [28, 37] rather than the GitHub API to perform control group matching. We query GHTorrent to identify NO_CoC_Repo for a given CoC_Repo that matches all specified criteria and has no CoC adoption record. Given that GHTorrent's latest complete data collection ended around 2020, from the dataset described in Sec. 3.1.1, we excluded the repositories that are not in GHTorrent or have CoCs created after January 1, 2020 to ensure data consistency. In total, we identified 4684 matched pairs of CoC_Repo and NO_CoC_Repo to form the treatment and control groups for different lengths of ΔT.

4.1.4 ***Control Variables.*** To account for confounding factors beyond CoC adoption, we include **seven** control variables (collected via the GitHub API), covering (1) **project-level metadata**, such as project scale, popularity, and maturity, and (2) **signals of openness and community engagement**. Specifically, for (1) **project-level metadata**, we include **four** metadata variables per matched repository: the number of **commits**, **stars**, **repository age**, and **unique contributors**. For (2) **signals of openness and community engagement**, we include **three** binary indicators: the presence of a README file, a CONTRIBUTING file in the community profile, and whether it uses issue labels to support newcomers. Prior work has highlighted the importance of such practices in fostering accessible communitiy culture, including maintaining a comprehensive README file, offering contributing guidelines to establish community norms and expectations [19, 27, 61], and labeling issues to help newcomers onboard more easily [12, 75, 76]. To detect newcomer-supporting issue labels, we follow the defined method in the work by Tan et al. [76] to identify if the repository has adopted "Good First Issues" related labels.

4.1.5 ***Model Specification.*** We specify the model as follows:

$$Y_{it} = \beta_1 h_i + \beta_2 p_{it} + \beta_3 (h_i \times p_{it}) + \sum_{j=1}^{k} \gamma_j x_{j,it} + \epsilon_{it} + \alpha_{it} \quad (1)$$

where Y_{it} is the outcome variable (i.e., number of new contributors and number of disengaged contributors) for repository i at time t. The key components of the model are as follows:
- β_1: the effect of having a CoC (h_i, a binary indicator: 1 if the repository has a CoC, 0 otherwise).
- β_2: the time period effect (p_{it}, a binary indicator: 1 for post-treatment, 0 for pre-treatment).
- $\beta_3(h_i \times p_{it})$: the interaction term ***Has CoC : Post-treatment***, which is essential for DiD estimation, measuring the additional effect of having a CoC in the *post-treatment period (P_{post})*.
- $\sum_{j=1}^{k} \gamma_j x_{j,it}$: the control variables, where $x_{j,it}$ denotes the j-th control variable for repository i at time t, and γ_j is its associated coefficient. We examine ΔT values of 1, 3, 6, and 12 months.

Estimating the models. We preprocess the data by removing outliers (values outside the 1st-99th percentile range) in control and dependent variables for each group and time period. Count-based variables undergo logarithmic transformations to normalize the distribution. We then fit the model specified in equation (1) with two outcome variables (i.e., number of new contributors and disengaged contributors) as Model 1 and Model 2.

4.1.6 ***Result: Impact of CoC Presence on Community Engagement.*** Tab. 4 demonstrates the results for Model 1 (new contributors) and Model 2 (disengaged contributors) across four different ΔT. Overall, the interaction term ***Has CoC : Post-treatment*** ($\beta_3(h_i \times p_{it})$) reveals that CoC adoption positively impacts the number of new contributors across various ΔT (1, 3, 6, and 12 months). Interestingly, this adoption also increases the number of disengaged contributors, but primarily over shorter ΔT=1 and 3 months).

In **Model 1**, the interaction term ***Has CoC : Post-treatment*** is positive and significant across all intervals, indicating that CoC adoption is associated with an increase in new contributors during the post-treatment period. Although the ***Has CoC*** term is also positive and significant, it does not capture temporal changes. The ***Post-treatment*** variable is significantly negative at the 6- and 12-month intervals, suggesting an overall decrease in new contributors over time, independent of CoC adoption, while in **Model 2**, the interaction term is positive and significant for 1- and 3-months, indicating a small increase in disengaged contributors shortly after CoC adoption. However, this effect is not significant at 6- and 12-months, suggesting limited long-term impact on contributor disengagement. For the R-squared explanation power of the models, Model 1 has lower values than Model 2, which may indicate that newcomer engagement is influenced by external factors not included in the model, such as exposure through social media or other outreach efforts. This does not compromise the validity of the DiD estimates, provided that such factors affect treatment and

Table 4: Regression Results for DiD Models (*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$)

	Model 1: Newly Contributors				Model 2: Disengaged Contributors			
Time Interval	1m	3m	6m	12m	1m	3m	6m	12m
Has CoC	0.063***	0.051***	0.054***	0.035	0.002	-0.016	-0.013	-0.026
Post-Treatment	-0.014*	-0.029**	-0.050***	-0.106***	-0.005	-0.018	-0.021	-0.064**
Has CoC : Post-Treatment	0.022*	0.078***	0.086***	0.102**	0.022**	0.042**	0.016	0.061
Commits Count (log)	-0.010***	0.007*	0.018***	0.020*	0.005**	0.005	0.005	0.009
Contributors Count (log)	0.073***	0.179***	0.252***	0.373***	0.112***	0.282***	0.443***	0.644***
Stars Count (log)	0.005**	0.009**	0.022***	0.024***	0.001	0.010***	0.012**	0.019**
Repo Age (log)	-0.067***	-0.112***	-0.148***	-0.170***	-0.041***	-0.137***	-0.196***	-0.241***
Has Contributing	0.043***	0.079***	0.067***	0.106***	0.003	-0.013	0.012	0.031
Has Readme	-0.005	0.005	-0.009	-0.034	0.000	0.005	-0.018	-0.005
Has GFI	-0.009	0.014	0.048***	0.104***	0.003	0.000	0.014	-0.040
R-squared (No.Obs)	0.08 (18724)	0.11 (13720)	0.13 (10192)	0.18 (5876)	0.09 (16832)	0.21 (11452)	0.32 (7396)	0.46 (3400)

control groups similarly. Although the increase in disengagement at the 12-month interval is not statistically significant, it can be due to the possibility of longer-term factors unrelated to CoC adoption. The DiD framework accounts for time-invariant confounders and assumes parallel trends, but cannot control for evolving project conditions such as reaching maturity or leadership changes, which may influence contributor behavior over time. Future research could explore these dynamics through qualitative methods such as interviews to better understand contributor motivations and influences on participation.

Control variables like ***Commits***, ***Contributors***, and ***Stars*** positively correlate with both new and disengaged contributors. In contrast, ***Repo Age*** shows a negative correlation, suggesting older repositories have lower contributor turnover. This may be due to increased activity and visibility attracting both new and departing contributors, while older repos stabilize with a more consistent contributor base and fewer engagement fluctuations. Regarding the factors related to community onboarding practices, ***Has Contributing*** is positively associated with new contributor activity across all time intervals, suggesting that, after accounting for other variables in the model, projects with a contributing guide tend to see higher rates of new contributor engagement. For disengaged contributors, the coefficients are smaller and not statistically significant, indicating no clear association. For ***Has Readme***, the coefficients for both new and disengaged contributors are close to zero and not statistically significant in any time interval. This suggests that the presence of a README file does not show a meaningful relationship with contributor joining or disengaging. For ***Has GFI***, the coefficients are not statistically significant for disengaged models, while demonstrating a positive correlation for new contributors for the 6- and 12-month intervals.

This suggests that CoC adoption has a significant and *positive effect* on increasing new contributor engagement across time intervals. There is no clear evidence that CoC adoption reduces disengagement among existing contributors: as effects are also positive but limited to the short term, if present at all.

(RQ2.1) Insight. CoC adoption likely serves as a visible signal of inclusivity that attracts new contributors, rather than to reshape long-term engagement. While effective for onboarding, its limited effect on reducing disengagement suggests that sustained retention depends on broader community practices beyond policy.

4.2 (RQ2.2) CoC Update vs Engagement

Results from RQ2 reveal differences in CoC attributes across repositories, including the size of updates and types of changes made after adoption. These differences may reflect how much importance each community places on its CoC and whether it is actively maintained to meet evolving community needs. Following the DiD analysis of CoC adoption's impact, we next examine how specific CoC change types relate to changes in community engagement.

4.2.1 ***Methodology: Defining Focal Commits to Model Community Engagement Changes.*** To assess whether CoC updates influence community engagement, we compare engagement changes before and after an update. Analyzing every update is impractical due to varying update frequencies, as some occur within days, which limits the observation windows, while high-commit repositories may skew results. To address this, we select a single CoC update per project for analysis. Future work could refine this approach by exploring short-term engagement metrics or examining long-term trends within individual repositories.

In particular, as illustrated in Fig. 5, we identify a ***focal commit*** ($C4$), which corresponds to a CoC update created at the time of T_{focal} ($= T_{C4}$) that has a significant distance from its previous CoC commit ($= T_{C3}$), exceeding a predefined threshold (ΔT), which presents the observation window for analyzing changes in community engagement. This can ensure a meaningful inactivity period. While some focal commits are soon followed by another CoC update, our analysis focuses on engagement changes associated with breaking inactivity. The presence of a subsequent CoC commit may strengthen the observed effect but does not compromise its validity. Including these cases boosts sample size and statistical power. Therefore, effect observed should still be valid, though the size of the effect may change subject to future explorations. Through comparing the metrics of the community engagement during the *pre-update period* $P_{Pre-update}$ ($= T_{focal} - \Delta T$) and *post-update period* $P_{Post-update}$ ($= T_{focal} + \Delta T$), we can explore the relationship between the community engagement dynamics and the CoC update attributes at T_{focal}. We investigated multiple observation windows, setting ΔT to 1, 3, 6, and 12 months.

Selecting CoC update attributes for modeling. Using the dataset in Sec. 3.1.2, we start with 35,841 repositories that have multiple CoC commits. We examine two factors for each *focal commit*: (1) change size (LOC) and (2) CoC change type (11 types, Tab. 1). CoC change types are identified using the method in Sec. 3.2. We also include community metadata as *control variables* (e.g., commit count, authors, and repo age in pre- and post-update periods).

Modeling the relationship between CoC update attributes and community engagement. To analyze this relationship, we categorize repos based on factors described above. Rather than using one single regression model, where interpreting interactions

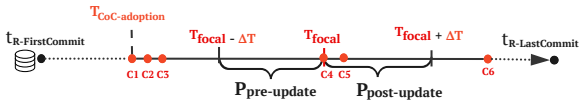


Figure 5: Illustration of focal commits analyzed in RQ 3.2. Red dots indicate CoC-related commits; $C1$ is initial adoption, and $C4$ denotes the focal commit examined. For each $C4$, we compare engagement before and after the update, referred to as periods $P_{pre-update}$ and $P_{post-update}$, respectively.

between multiple change types and change sizes becomes complex, we adopt a more granular approach by examining each factor separately: we split repos into two groups for each of the 12 factors:

- **[Type-based factor]:** For each of the 11 **change types**, repositories are classified into G_Y (where the change *occurred* in the *focal commit*) and G_N (where it did not).
- **[Size-based factor]:** Repositories are categorized into $G^{>}_{size}$ (where $G^{>}_{size}$ means the CoC changes size of the *focal commit is greater than* the median and $G^{\leq}_{size}$ (where it is $\leq$ to the median).

While this approach does not capture factor interactions, it enhances interpretability by isolating effects. Future research could investigate how combinations of factors influence community engagement. For each factor, we estimate a model to understand its correlation with community engagement. Here, we focus on new contributors to measure engagement. While disengaged contributors could also be considered, identifying them requires a longer observation period ($2\Delta T$), which significantly reduces the available data. Given the uncertainty in pinpointing CoC focal commits, we prioritize data availability and restrict our analysis accordingly. The model is as follows:

$$Y_{it} = \beta_1 p_{it} + \sum_{j=1}^{k} \gamma_j x_{j,it} + \epsilon + \alpha \tag{2}$$

where Y_{it} is the number of new contributors in period t for repository i; p_{it} is a binary indicator–1 for $P_{Post-update}$ period, 0 for $P_{Pre-update}$ period, for repository i; $\sum_{j=1}^{k} \gamma_j x_{j,it}$ is the control variables (repo age, commits count, number of unique commit authors) in period t for repository i.

By estimating p_{it}, we assess whether the factor of interest in the current group G is associated with variations in newcomers count Y_{it} during the $P_{Pre-update}$ and $P_{Post-update}$ period. Comparing the β_1 values across groups for the same CoC update-related variable will reveal the magnitude and direction of the variable's relationship with changes in the number of newcomers.

4.2.2 ***Result.*** We estimate 48 models, each corresponding to one of 12 factors across four time intervals (1, 3, 6, and 12 months), using Equation (2) to analyze the correlation between these factors and changes in newcomer counts. Full results illustrations are available in the replication package [72]; here, we highlight the key observations.

Across shorter intervals (ΔT = 1, 3 months), most CoC-related factors show no significant effect, except for *approachable*, which correlates with a rise in new contributors. This indicates that CoC updates have limited immediate correlation to attracting newcomers over ΔT time. **While at ΔT = 6 months**, four change types (*Scope*, *Justify the adoption of CoC*, *Version update*, and *Acknowledgment*) in G_Y demonstrate a statistically significant positive correlation with new contributor growth. This suggests that these specific CoC modifications are associated with an increase in new contributors over time. Although the correlation is not statistically significant, the group with larger CoC change sizes $G^{>}_{size}$ shows a tendency toward a positive relationship with new contributor growth, contrary to the group with smaller changes $G^{\leq}_{size}$. **At ΔT = 12 months**, multiple attributes in both groups exhibit statistically significant *negative* correlations with new contributor influx, suggesting that time, rather than CoC change attributes, may be the dominant factor. However, the effect sizes are generally small, indicating that the decline in engagement is subtle despite statistical significance. Similarly, we experimented with larger ΔT values ranging from 6 to 12 months (e.g., $\Delta T = 10$) but found few significant correlations. Therefore, the long-term effect of these CoC changes remains unclear. Future research could explore through longitudinal studies and examine how a specific type of CoC change contributes to sustained engagement over time.

(RQ2.2) Insight. CoC updates may function less as direct levers for immediate engagement and more as subtle signals of a project's evolving governance maturity. Contributors appear to respond more to the presence of deliberate, meaningful changes than to the mere occurrence of updates. Over time, however, structural or environmental factors likely outweigh individual policy revisions, highlighting the importance of coupling CoC changes with broader, visible community efforts to support sustained growth.

5 Discussion

5.1 Paradox with Contributor Attrition

The dual impact of CoC adoption. Our quantitative analysis reveals a **nuanced relationship** between CoC adoption, updates, and contributor engagement (RQ2). While adopting a CoC initially attracts new contributors, likely signaling a commitment to inclusivity, this effect diminishes over time, suggesting that a CoC alone is insufficient for sustained engagement. Moreover, CoC adoption correlates with a short-term increase in disengaged contributors, possibly due to initial resistance or misalignment with evolving community expectations. Meanwhile, **turnover is an inherent characteristic of OSS communities, bringing both risks and opportunities.** While disengagement is often seen as negative, a dynamic cycle of contributor departure and newcomer influx can introduce fresh ideas, enhance adaptability, and prevent stagnation. Prior research highlights the value of balancing experienced contributors, who preserve institutional knowledge, with new contributors that can drive innovation. However, excessive turnover, especially the loss of long-term maintainers, can disrupt development and sustainability and cause discontinuity of knowledge [42, 52]. Therefore, balancing the turnover of contributors can be critical to the sustainability of OSS projects.

The dilemma of CoC enforcement on community engagement. One explanation for CoC adoption's impact on contributor engagement is its role as a governance mechanism. While CoCs aim to foster inclusivity and address harmful behavior, their enforcement can spark debate over moderation, freedom of speech, and meritocracy. Some contributors see strict enforcement as restrictive, while others view it as necessary for maintaining a welcoming environment. This tension aligns with prior observations on openness and governance in OSS [25], and also reflects the *paradox of tolerance* [58], a concept that argues that unlimited tolerance can allow intolerant behaviors to thrive, ultimately undermining inclusivity.

In OSS, communities that fail to address harmful behaviors may drive away marginalized contributors, while overly rigid enforcement risks alienating those who feel constrained by governance measures. Thus, OSS communities could face a dual challenge when it comes to CoC adoption and implementation: (1) If they tolerate harmful behavior too much, inclusivity suffers as contributors feel unsafe. (2) If enforcement is perceived as rigid or exclusionary, contributors who value autonomy may disengage. Considering both sides of the challenges when it comes to the implementation of CoCs, it is necessary to have an open discussion among community members rather than framing CoC enforcement as a binary issue. Future research could examine how different governance mechanisms can help to balance CoC enforcement and also address the potential maintenance burden while ensuring long-term sustainability.

5.2 Implications for OSS Communities

Our analysis offers practical takeaways for OSS communities navigating governance, engagement, and community trust.

Adopt CoCs as a meaningful signal, then treat them as living documents. We find that many projects treat CoC adoption as a one-off act. RQ2 shows that most projects rarely change CoCs post-adoption, even as their communities evolve, while long-lived projects practice keeping CoCs up-to-date. To sustain trust and relevance, CoCs should be treated as dynamic, not static.

Leverage diverse update strategies aligned with community needs. Different update patterns reflect different governance priorities. Projects aiming for agility and responsiveness might benefit from incremental updates. In contrast, more structured communities may prefer periodic revisions, aligned with organizational or platform-wide policy changes. Choosing update strategies that match a project's size, maturity, and culture can help reduce friction.

Platform support matters. Platform features (e.g., GitHub's "community profile" checklist) can encourage timely CoC updates. This highlights the role that platforms can play in nudging projects toward healthy governance practices.

Promote transparency and inclusiveness in CoC governance. Our observations on CoC updates from different communities (RQ1.2) suggest that transparent CoC changes are more likely to be seen as legitimate. Projects can engage contributors through public discussions, changelogs for governance files, or advisory groups. This fosters legitimacy and aligns with practices in mature projects like *elixir-lang/elixir* (as shown in Cluster 4 results).

Track and assess CoC influence systematically. Assessing CoC effectiveness is challenging due to the lack of systematic evaluation. Unlike other metrics, CoC impacts are often indirect and difficult to quantify [43]. Similar to corporate EDI training, which frequently fails to drive lasting change [68], our findings suggest CoCs may attract new contributors while disengaging existing ones, raising questions about their effectiveness (RQ2). OSS projects could issue transparency reports [24] detailing CoC-related incidents, enforcement actions, and contributor feedback. Such documentation not only helps internal reflection but also signals accountability externally. Metrics from initiatives like CHAOSS [6] (e.g., contributor growth and retention) can also help track whether governance practices align with community goals.

5.3 Implications for Future Research

Our study offers researchers a comprehensive dataset on **CoC adoption in OSS** and quantitative evidence regarding its impact. Future research could incorporate longitudinal data on contributor activity, clarifying the **long-term influence** of CoCs on *community engagement*. Qualitative studies could demonstrate contributor perceptions and response to these policies in practice. Examining **enforcement practices and their effects on contributors with different levels of community involvement** (e.g., core vs.peripheral contributors) would also be valuable. Such work could explore whether specific enforcement strategies disproportionately affect certain groups, shape community cohesion, or lead to measurable improvements in engagement.

Finally, future research could investigate **proactive governance approaches**, including automated tools such as violation-detection bots [8] or systems that help maintain CoCs by integrating community feedback and reducing maintainer workload. Studies on structured governance frameworks: for example, automated policy-auditing tools, version-controlled governance models, or machine-assisted moderation systems, which could further support communities in managing CoCs more efficiently.

6 Threats to Validity

Despite our best efforts, our study faces several validity threats. First, our analysis relies on community-linked CoC files, which may overlook alternative adoption methods, and our use of file paths to approximate unique CoCs may overcount duplicated content, although it offers a practical way to capture structural distinctions at scale. We included non-software repositories in our analysis, allowing us to capture a broader view of CoC adoption in OSS. However, these repositories may display varying update patterns and levels of engagement. We also focused exclusively on GitHub, while prior research shows OSS is not confined to this specific platform [78]. Future work could replicate our study with targeted subsets and other platforms. LLM-assisted labeling may miss nuanced intentions in CoC changes, and variations in formats can also complicate reliable interpretation. In the DiD analysis, although we controlled for observable project characteristics and community norms, unobserved factors such as social dynamics and leadership may still bias results. Our approach may also favor standardized CoC practices, potentially limiting generalizability. Finally, our causal analysis focuses on CoC adoption rather than subsequent updates. While we observe a causal relationship between adoption and contributor engagement, we do not establish causality for CoC updates.

7 Conclusion

In this study, we investigated CoC adoption and maintenance in OSS and its short-term effects on contributor engagement. Through analyzing commit histories and adoption using a DiD approach, we found CoC adoption may boost newcomer participation while also increase short-term disengagement. These findings lay the groundwork for future research on the evolving role of CoCs and their long-term, demographic-specific impacts.

Acknowledgments

This work was supported in part by a gift from NumFOCUS.

References

[1] Alibaba. 2025. Alibaba GitHub account. https://github.com/alibaba/

[2] Bootstrap-vue. 2025. Bootstrap-vue GitHub repository. https://github.com/bootstrap-vue/bootstrap-vue

[3] Hudson Borges, Andre Hora, and Marco Tulio Valente. 2016. Understanding the Factors That Impact the Popularity of GitHub Repositories. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 334–344.

[4] Brantly Callaway and Pedro H.C. Sant'Anna. 2021. Difference-in-Differences with multiple time periods. *Journal of Econometrics* 225, 2 (2021), 200–230.

[5] Eshwar Chandrasekharan, Mattia Samory, Shagun Jhaver, Hunter Charvat, Amy Bruckman, Cliff Lampe, Jacob Eisenstein, and Eric Gilbert. 2018. The Internet's Hidden Rules: An Empirical Study of Reddit Norm Violations at Micro, Meso, and Macro Scales. In *Proc. Conf. Computer Supported Cooperative Work (CSCW)*, Vol. 2. ACM Press, Article 32.

[6] CHAOSS. 2024. Metrics and metrics models. https://chaoss.community/kb-metrics-and-metrics-models/

[7] Chx. 2018. I was an open source lead developer and got banned due to Code Of Conduct violations. https://medium.com/@chx/a-note-from-an-open-source-lead-developer-who-got-banned-from-his-community-due-to-code-of-conduct-22d8f066ab9e

[8] Sergio Cobos and Javier Luis Cánovas Izquierdo. 2025. A Bot-Based Approach to Manage Codes of Conduct in Open-Source Projects. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS '25)*. IEEE Press, 59–69.

[9] CocoaPods. 2025. CocoaPods GitHub repository. https://github.com/CocoaPods/CocoaPods

[10] Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46.

[11] DARIAEngineering. 2025. Dcaf case management GitHub repository. https://github.com/DARIAEngineering/dcaf_case_management

[12] Jan Willem David Alderliesten and Andy Zaidman. 2021. An Initial Exploration of the "Good First Issue" Label for Newcomer Developers. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. 117–118.

[13] Stephen G. Donald and Kevin Lang. 2007. Inference with Difference-in-Differences and Other Panel Data. *The Review of Economics and Statistics* 89, 2 (2007), 221–233.

[14] Mike Droesch, Amit Karp, Ariel Sterman, and Ethan Kurzweil. 2020. Measuring the engagement of an open source software community. https://www.bvp.com/atlas/measuring-the-engagement-of-an-open-source-software-community

[15] Christina Dunbar-Hester and Jenna P. Carpenter. 2021. Hacking Diversity: The Politics of Inclusion in Open Technology Cultures. *IEEE Technology and Society Magazine* 40, 1 (2021), 16–18.

[16] Elixir-lang. 2025. Elixir GitHub repository. https://github.com/elixir-lang/elixir

[17] Bogdan Enache. 2021. *Effect of the introduction of code of conduct in collaborative development.* Master's thesis. Eindhoven, Netherlands.

[18] EthicalSource. 2025. Contributor Covenant GitHub repository. https://github.com/EthicalSource/contributor_covenant

[19] Hongbo Fang, James Herbsleb, and Bogdan Vasilescu. 2023. Matching Skills, Past Collaboration, and Limited Competition: Modeling When Open-Source Projects Attract Contributors. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM Press, 42–54.

[20] Hongbo Fang, Hemank Lamba, James Herbsleb, and Bogdan Vasilescu. 2022. "This is damn slick!": estimating the impact of tweets on open source project popularity and new contributors. In *Proc. Int'l Conf. Software Engineering (ICSE)*. ACM Press, 2116–2129.

[21] Jennifer Fereday and Eimear Muir-Cochrane. 2006. Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development. *International Journal of Qualitative Methods* 5, 1 (2006), 80–92.

[22] Jessica L. Feuston and Jed R. Brubaker. 2021. Putting Tools in Their Place: The Role of Time and Perspective in Human-AI Collaboration for Qualitative Analysis. In *Proc. Conf. Computer Supported Cooperative Work (CSCW)*, Vol. 5. ACM Press, Article 469.

[23] Organization for Ethical Source. 2014. Contributor Covenant: A Code of Conduct for Open Source and Other Digital Commons Communities. https://www.contributor-covenant.org/

[24] Linux Foundation. 2022. Linux Foundation Code of Conduct transparency report. https://www.linuxfoundation.org/blog/linux-foundation-events-code-of-conduct-transparency-report-2022-event-summary

[25] Hana Frluckaj, Nikki Stevens, James Howison, and Laura Dabbish. 2024. Paradoxes of openness: Trans experiences in open source software. In *Proc. Conf. Computer Supported Cooperative Work (CSCW)*, Vol. 8. ACM Press, 1–24.

[26] GadflyKin. 2016. Codes of Conduct are all the rage. https://medium.com/@GadflyKin/code-of-conducts-are-all-the-rage-597fe033cd7f

[27] Matthew Gaughan, Kaylea Champion, Sohyeon Hwang, and Aaron Shaw. 2025. The Introduction of README and CONTRIBUTING Files in Open Source Software Development. In *2025 IEEE/ACM 18th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*. 191–202.

[28] Ghtorrent. 2019. Ghtorrent. https://ghtorrent.github.io/tutorial/

[29] GitHub. 2017. Open Source Survey. https://opensourcesurvey.org/2017/

[30] GitHub. 2025. Add a Code of Conduct to your project. https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/adding-a-code-of-conduct-to-your-project

[31] GitHub. 2025. API Example: Community profile of libdxf. https://api.github.com/repos/bert/libdxf/community/profile

[32] GitHub. 2025. API Example: libdxf CoC file. https://api.github.com/repos/bert/libdxf/contents/CODE_OF_CONDUCT

[33] GitHub. 2025. Community profile for public repositories. https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/about-community-profiles-for-public-repositories

[34] GitHub. 2025. GitHub API search engine. https://docs.github.com/en/rest/search/search?apiVersion=2022-11-28

[35] GNU. 2025. Hunks (comparing and merging files). https://www.gnu.org/software/diffutils/manual/html_node/Hunks.html

[36] Andrew Goodman-Bacon. 2021. Difference-in-differences with variation in treatment timing. *Journal of Econometrics* 225, 2 (2021), 254–277. Themed Issue: Treatment Effect 1.

[37] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: GitHub's data from a firehose. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 12–21.

[38] Open Source Guides. 2024. Open Source Guides. https://opensource.guide

[39] Hackference. 2025. Hackference CoC: an open source code of conduct template for user groups and small conferences. https://github.com/hackference/code-of-conduct

[40] Chris Holdgraf. 2019. CoC: Adding a list of violating actions. https://work.chron.com/code-conduct-violation-5380.html

[41] Jeremy Howard. 2018. I violated a code of conduct. https://www.fast.ai/posts/2020-10-28-code-of-conduct.html

[42] D. Izquierdo-Cortazar, G. Robles, F. Ortega, and J.M. Gonzalez-Barahona. 2009. Using Software Archaeology to Measure Knowledge Loss in Software Projects Due to Developer Turnover. In *2009 42nd Hawaii International Conference on System Sciences*. 1–10.

[43] Renee Li, Pavitthra Pandurangan, Hana Frluckaj, and Laura Dabbish. 2021. Code of Conduct Conversations in Open Source Software Projects on Github. In *Proc. Conf. Computer Supported Cooperative Work (CSCW)*, Vol. 5. ACM Press, Article 19.

[44] Jim Lynch. 2016. Are Codes of Conduct dangerous to open source software development? https://www.infoworld.com/article/2244934/are-codes-of-conduct-dangerous-to-open-source-software-development.html

[45] Danaja Maldeniya, Ceren Budak, Lionel P. Robert Jr., and Daniel M. Romero. 2020. Herding a Deluge of Good Samaritans: How GitHub Projects Respond to Increased Attention. In *Proceedings of The Web Conference 2020 (WWW '20)*. ACM Press, 2055–2065.

[46] Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica* 22, 3 (2012), 276–282.

[47] Microsoft. 2025. Microsoft CoC file. https://api.github.com/repos/microsoft/.github/contents/CODE_OF_CONDUCT.md

[48] Microsoft. 2025. Vscode community profile. https://api.github.com/repos/microsoft/vscode/community/profile

[49] Microsoft. 2025. Vscode GitHub repository. https://github.com/microsoft/vscode

[50] Steve Milano. 2021. What Is a Code of Conduct Violation? https://work.chron.com/code-conduct-violation-5380.html

[51] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian KaUstner. 2022. "Did you miss my comment or what?": understanding toxicity in open source discussions. In *Proc. Int'l Conf. Software Engineering (ICSE)*. ACM Press, 710–722.

[52] Courtney Miller, David Gray Widder, Christian Kästner, and Bogdan Vasilescu. 2019. Why do people give up flossing? a study of contributor disengagement in open source. In *Open Source Systems: 15th IFIP WG 2.13 International Conference, OSS 2019*. Springer, 116–129.

[53] Mozilla and contributors. 2025. Open Leadership Training Series: Write or Choose a Code of Conduct. https://mozilla.github.io/open-leadership-training-series/articles/building-communities-of-contributors/write-a-code-of-conduct/

[54] Dominic Muller. 2018. No Code of Conduct: A Code of Conduct for Adults in Open Source Software. https://github.com/domgetter/NCoC

[55] Engineering National Academies of Sciences and Medicine. 2022. *Fostering Responsible Computing Research: Foundations and Practices*. The National Academies Press.

[56] OctoPrint. 2025. OctoPrint CoC file. https://github.com/OctoPrint/OctoPrint/blob/master/CODE_OF_CONDUCT.md

[57] OpenAI. 2025. GPT-4 Turbo in the OpenAI API. https://help.openai.com/en/articles/8555510-gpt-4-turbo-in-the-openai-api

[58] Karl Popper, Ernst Hans Gombrich, and Vaclav Havel. 2012. *The open society and its enemies*. Routledge.

[59] Reddit post. 2025. Github's new Code of Conduct says. https://www.reddit.com/r/technology/comments/3fpnuw/githubs_new_code_of_conduct_says_our_open_source/

[60] Reddit post. 2025. No Code of Conduct: A Code of Conduct for Adults in Open Source Software. https://www.reddit.com/r/linux/comments/9gv6we/no_code_of_conduct_a_code_of_conduct_for_adults/

[61] Huilian Sophie Qiu, Yucen Lily Li, Susmita Padala, Anita Sarma, and Bogdan Vasilescu. 2019. The Signals that Potential Contributors Look for When Choosing Open-source Projects. In *Proc. Conf. Computer Supported Cooperative Work (CSCW)*, Vol. 3. ACM Press, Article 122.

[62] Neill Robson. 2018. Diversity and decorum in open source communities. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM Press, 986–987.

[63] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[64] Rust-embedded. 2025. Rust-embedded CoC file. https://github.com/rust-embedded/.github/blob/master/CODE_OF_CONDUCT.md

[65] Rust-embedded. 2025. Rust-embedded GitHub account. https://github.com/rust-embedded/

[66] Johnny Saldaña. 2021. The coding manual for qualitative researchers. *The coding manual for qualitative researchers* (2021), 1–440.

[67] Katharine Schwab. 2018. Codes of ethics probably don't work. https://www.fastcompany.com/90250846/codes-of-ethics-probably-dont-work

[68] Jesse Singal. 2023. Opinion | Diversity Trainings Don't Work. Here's What Could. https://www.nytimes.com/2023/01/17/opinion/dei-trainings-effective.html

[69] Vandana Singh, Brice Bongiovanni, and William Brandon. 2021. Codes of conduct in Open Source Software—for warm and fuzzy feelings or equality in community? *Software Quality Journal* (2021), 1–40.

[70] Richard Stallman. 2018. Announcing the GNU Kind Communication Guidelines. https://lists.gnu.org/archive/html/info-gnu/2018-10/msg00001.html

[71] Richard Stallman. 2022. GNU Kind Communications Guidelines. https://www.gnu.org/philosophy/kind-communication.en.html

[72] Jiayi Sun, Hongbo Fang, Junming Zhang, Jiakai Shi, Ruitao Lai, Anita Ihuman, Richard Littauer, and Shurui Zhou. 2025. *Replication package*. https://zenodo.org/records/15031104

[73] Sunpy. 2025. Sunpy CoC file. https://github.com/sunpy/.github/blob/main/CODE_OF_CONDUCT.md

[74] Sunpy. 2025. Sunpy GitHub account. https://github.com/sunpy/

[75] Xin Tan, Yiran Chen, Haohua Wu, Minghui Zhou, and Li Zhang. 2023. Is it Enough to Recommend Tasks to Newcomers? Understanding Mentoring on Good First Issues. In *Proc. Int'l Conf. Software Engineering (ICSE)*. IEEE Press, 653–664.

[76] Xin Tan, Minghui Zhou, and Zeyu Sun. 2020. A first look at good first issues on GitHub. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM Press, 398–409.

[77] Parastou Tourani, Bram Adams, and Alexander Serebrenik. 2017. Code of conduct in open source projects. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 24–33.

[78] Milo Z Trujillo, Laurent Hébert-Dufresne, and James Bagrow. 2022. The penumbra of open source: projects outside of centralized platforms are longer maintained, more academic and more collaborative. *EPJ Data Science* 11, 1 (2022), 31.

[79] Unoplatform. 2025. Uno GitHub repository. https://github.com/unoplatform/uno

[80] Helen Williams. 2001. Maintaining a harassment-free workplace. https://web.archive.org/web/20120328034350/http://apsc.gov.au/publications01/harassment.htm

[81] Ziang Xiao, Xingdi Yuan, Q. Vera Liao, Rania Abdelghani, and Pierre-Yves Oudeyer. 2023. Supporting Qualitative Analysis with Large Language Models: Combining Codebook with GPT-3 for Deductive Coding *(IUI '23 Companion)*. ACM Press, 75–78.