

Supplementary Material: Beyond Adoption: Examining the Evolution and Impact of Code of Conduct on Open Source Communities

Anonymous Author(s)

1 (RQ1) CoC implementation and maintenance practices

1.1 (RQ1.1) Intentions of CoC Changes

Figure 1 illustrates the different CoC update types over time, highlighting their occurrences in commits over time across all repositories analyzed. The heatmap reveals that *Non-Content Changes* consistently remained the most frequent type of update, with notable peak occurrences. Additionally, updates related to *Availability*, *Enforcement*, and *Behavior Specification* have shown an increase in frequency.

The overall trend shows that CoC updates have been an ongoing process, with *non-content changes* consistently prevalent and a gradual rise in updates related to enforcement and behavioral expectations. This pattern reflects a growing tendency among OSS communities in implementing CoC to make it more actionable with more refined enforcement mechanism (clarity in expectation and how to handle violations) and approaches (contacts of community for reporting violations).

1.2 (RQ1.2) Patterns in CoC Update over time across Various Projects

1.2.1 Repository Types. To assist in interpreting the CoC update patterns among different repositories, it is valuable to understand the nature of the project (e.g., software development, resource collection, community management) and the repository’s stage of maturity. Therefore, we manually reviewed the repositories by examining their GitHub pages and applied inductive coding to characterize the nature of each repository. Two authors initially coded 40 repositories independently to develop the codebook, then compared and discussed our coding results to refine the codebook, ensuring consistency in the coding process. After reaching a final agreement with a Cohen’s Kappa score of 0.82, one author applied the finalized codebook to the remaining repositories in the sample. In total, we summarized repositories into five valid types and discarded the repositories that are invalid (e.g., a developer can upload a clone of popular projects with its full commit history). The final codebook and corresponding labeling results are shown in Tab. 1.

1.2.2 Steps for Clustering. For identifying patterns of CoC changes over time, 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. Figure 2 illustrates the three-step process for clustering repositories based on the three factors:

Step 1 (Vectorization of repositories) converts raw repository data into a high-dimensional feature representation using one-hot encoding. **Step 2 (Feature Dimensionality Reduction for K-Means Clustering)** applies feature dimensionality reduction, where segmented CoC changes undergo binarization (8 groups)

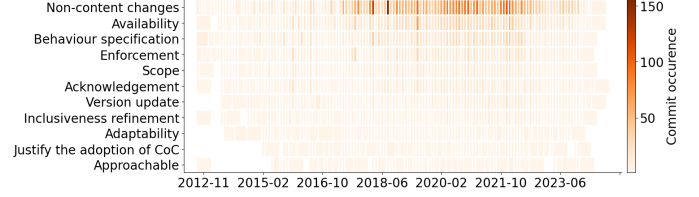


Figure 1: Illustration of CoC change types observed in commits over time. The number of commits associated with each CoC change type is shown, with the x-axis representing the timeline, the y-axis listing change types, and the color intensity indicating the number of commits associated with each type.

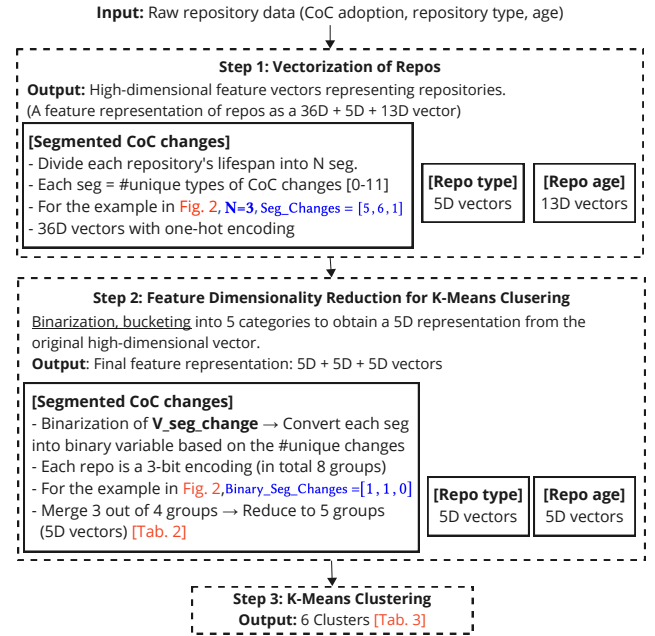


Figure 2: Overview of RQ1.2 method. Tables and figures highlighted in red correspond to their respective indexes in the main paper.

and group merging (5 groups), alongside repository type and age transformations, resulting in a 5D + 5D + 5D feature representation. Finally, **Step 3 (K-Means clustering)** groups repositories into 8 clusters based on their transformed features.

Table 1: Codebook for Labelling Repository Types

Repo Type	Definition	Example
Software	Hosts source code for a sw. project.	pixijs/pixi-haxe
Resource Collection	Focus on collecting resources to support governance, communication, and information sharing within an organization or by an individual.	SurPathHub/starter-kit,
CoC Doc.	Host only CoC document for reference/adoption by other groups, communities, or organizations.	CDCgov/code-of-conduct
OSS Template	Provide templates for starting an open-source project, offering reusable structures and guidelines.	msc-acse/git-novice
CoC Template	Provide reusable CoC templates for customization by other groups, communities, or organizations.	hackference/code-of-conduct

1.2.3 Clustering Results. 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, *Femme-js/Hoaxify* added a CoC in 2020 and made only minor same-day updates, such as formatting and contact availability, while *DARIAEngineering/dcraf_case_management*, created in 2015, 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 suggest that some projects may view the period immediately following CoC adoption as the appropriate time to make necessary 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.

Projects following the **100-G2 (N=121)** pattern show more early initiative. For instance, *unoplatform/uno* introduced its CoC in 2018 and actively refined it over the next year before settling into stability, addressing behavior, scope, and formatting, followed by no further revisions. Similar observations are seen in *rack/rack-attack*, *pytorch/botorch*, and *athensresearch/athens*. These cases suggest that some projects intentionally front-load CoC updates in the early stages, aiming to produce a self-contained and complete document.

Meanwhile, the **001-G4 (N=77)** pattern reflects delayed but significant updates. Software projects like *qiskit-community/qiskit-aqua* (created in 2018 and dormant since 2021) made significant updates years after adoption, reflecting a shift toward centralized CoC management under the organization’s main repository. Similarly, *bootstrap-vue/bootstrap-vue*, an active software repository created in 2016, adopted Contributor Covenant v1.4 in 2017 and made only minor formatting changes until 2023, when it upgraded to version 2.1 of the template. These cases suggest that some projects revise

their CoC primarily in response to **external changes**, such as updated templates or shifts in organizational structure, rather than through ongoing maintenance.

Cluster 2 (Resource-collection dominated, N=423) is dominated by *resource collection* projects and mirrors Cluster 1 in update patterns, with **000-G5** again being most common. Notably, a portion of repositories (**100-G2**) revise CoCs early on, but many later transition to external frameworks (e.g., PSF CoC [2]). This transition highlights how resource-focused communities may experiment with internal policy-making early on but eventually pivot to **external governance** models as they mature.

Cluster 3 (010-G3 pattern dominated, N=150) consists entirely of repositories exhibiting this pattern, where CoC revisions are concentrated during the **middle phase** of the project’s lifespan. Although less prevalent, this mid-life update pattern stands out by falling between the early and late revision phases seen in other clusters. The cluster includes a mix of software projects (N=108), resource collections (N=40), and dedicated CoC repositories (N=2). One illustrative example is *apache/superset*, a still-active software project created in 2015. Initially, the project linked externally to the Apache Software Foundation (ASF)’s CoC. In 2019, the maintainers updated the repository to include the full ASF CoC text into the repository. This change was motivated by GitHub’s “community profile” prompts, which encourage repositories to make community guidelines more visible. Contributors discussion around this change reflected a tension between accessibility and maintenance burden. One user noted, “*Not really sure if this helps the cause or just creates another thing to maintain*,” while others encouraged contributions to keep the content aligned: “*If there’s a discrepancy between the two, let us know or submit a PR to fix it*.” Since then, the CoC has received only minor changes, such as link fixes, formatting changes, and updated contact information without major content revisions, suggesting that the mid-life update may sometimes be reactive, driven by visibility or compliance pressures rather than ongoing community feedback.

Cluster 4 (Long-lived, actively maintained projects dominated, N=101) is composed primarily of **consists mostly of mature, still-active software projects** with sustained maintenance. The dominant pattern is **000-G5 (N=44)**, where CoCs are adopted but rarely updated, suggesting that many projects treat them as a stable, one-time commitment.

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 [3] in 2014 and GitHub’s introduction of a one-click CoC template in 2016 [1] lowered the barrier to adoption and established it as a 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)*, 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)* pattern, continuously evolved their CoCs to reflect growing community scope and values—for example, changing “*project*” to “*ecosystem*” to signal inclusivity.

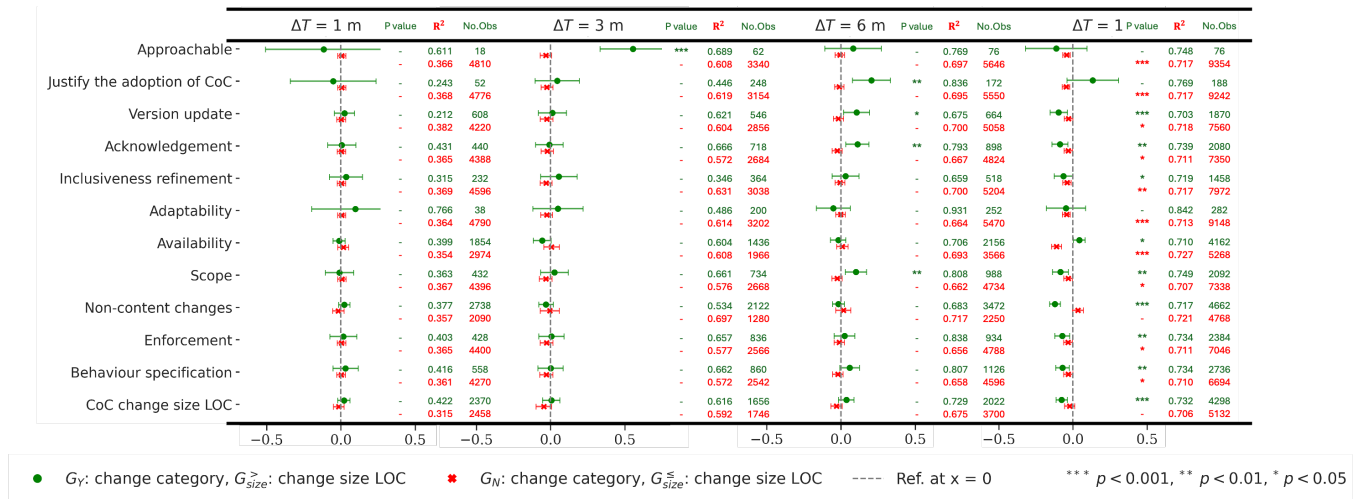


Figure 3: This plot shows the estimated coefficients and their 95% confidence intervals ($\alpha = 0.05$) for different groups (y-axis) across four different lengths of ΔT . Points represent the estimated coefficients, with error bars showing the confidence intervals.

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 project types, N=85) includes a mix of repositories that **consistently engage in sustained and repeated CoC maintenance**, distinguishing it from clusters dominated by static or one-time update patterns. All follow the **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 of activity still ongoing, with over 60% of the projects remaining active across all types.

Illustrative examples like *hackference/code-of-conduct* and *EthicalSource/contributor_covenant* demonstrate how CoCs evolve with community input, reflecting a collaboratively governed and responsive approach. Notably, such practices are not limited to template repositories—software and resource collections also exhibit sustained CoC engagement.

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. It illustrates how CoC stewardship can support long-term inclusivity and adaptability.

Cluster 6 (OSS templates dominated, N=23) is a small but distinct cluster composed entirely of open-source template repositories that are scaffolding projects designed to be reused or forked by others. Most follow the **000-G5** pattern, where CoCs are adopted early but remain unchanged, serving as **standardized boilerplate** rather than active governance tools. For example, *mpi-astronomy/mpi-python-template* includes a CoC without further updates, reflecting its role as part of a ready-made scaffold. These CoCs are typically preemptive, intended to signal expectations to downstream

adopters rather than manage an active community. This highlights a distinct use of CoCs: as propagated norms embedded in templates to promote best practices across derivative projects, even without ongoing community engagement.

Insight: CoC maintenance practices vary widely across OSS communities. While most projects adopt a CoC with minimal follow-up, a subset engage in sustained or lifecycle-specific updates, reflecting differing 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.

2 (RQ2) Impact of CoC

2.1 (RQ2.2) Relationship between CoC Update characteristics and community engagement

Figure 3 presents results of the estimated relationship between CoC update factors (i.e., change category and change size measured in LOC (lines of code)) and changes in community engagement, measured by the number of new contributors joining before and after the CoC update over a certain time period ΔT .

Overall, CoC updates show minimal short-term effects, except for *approachable* changes, which positively correlate with new contributor influx. At $\Delta T = 6$ months, changes related to *scope*, *justification*, *version updates*, and *acknowledgment* boost engagement, while larger updates show a weak positive trend. When $\Delta T = 12$ months, we observed significant negative correlations for most CoC change types regardless of the presence in the update, but the effect size is small and not robust when experimenting with slightly different choices of ΔT (e.g., 10 months). Therefore, the long-term effect of CoC updates remains an open question for future research.

References

- [1] 2023. GitHub: 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
[2] 2025. chicago python user group. <https://www.chipy.org/pages/conduct/>

[3] Organization for Ethical Source. 2014. <https://www.contributor-covenant.org/>