

Governance in Practice: How Open Source Projects Define and Document Roles

Pedro Oliveira
Pedro.Oliveira@nau.edu
Northern Arizona University
Flagstaff, AZ, USA

Marco Gerosa
marco.gerosa@nau.edu
Northern Arizona University
Flagstaff, AZ, USA

Tayana Conte
tayana@icomp.ufam.edu.br
Federal University of Amazonas
Manaus, Amazonas, Brazil

Igor Steinmacher
igor.steinmacher@nau.edu
Northern Arizona University
Flagstaff, AZ, USA

Abstract

Open source software (OSS) sustainability depends not only on code contributions but also on governance structures that define who decides, who acts, and how responsibility is distributed. We lack systematic empirical evidence of how projects formally codify roles and authority in written artifacts. This paper investigates how OSS projects define and structure governance through their GOVERNANCE.md files and related documents. We analyze governance as an institutional infrastructure, a set of explicit rules that shape participation, decision rights, and community memory. We used Institutional Grammar to extract and formalize role definitions from repositories hosted on GitHub. We decompose each role into scope, privileges, obligations, and life-cycle rules to compare role structures across communities. Our results show that although OSS projects use a stable set of titles, identical titles carry different responsibilities, and different labels describe similar functions, which we call role drift. Still, we observed that a few actors sometimes accumulate technical, managerial, and community duties. By understanding authority and responsibilities in OSS, our findings inform researchers and practitioners on the importance of designing clearer roles, distributing work, and reducing leadership overload to support healthier and more sustainable communities.

CCS Concepts

• **Software and its engineering** → **Open source model**; *Collaborative software development*.

Keywords

Open Source Software, Governance Model, Sustainability

ACM Reference Format:

Pedro Oliveira, Tayana Conte, Marco Gerosa, and Igor Steinmacher. 2026. Governance in Practice: How Open Source Projects Define and Document Roles. In *19th International Conference International Workshop on Cooperative and Human Aspects of Software (CHASE'26)*, April 12–18, 2026, Rio de Janeiro, Brazil. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3794860.3794911>



This work is licensed under a Creative Commons Attribution 4.0 International License. *CHASE'26, Rio de Janeiro, Brazil*

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2484-8/2026/04
<https://doi.org/10.1145/3794860.3794911>

1 Introduction

Open source software (OSS) plays a central role in the digital infrastructure that sustains modern computing. Over 90% of commercial software includes OSS components [23], and nearly all major technology platforms depend on them [8]. As the role of OSS expands, governance has emerged as a key factor related to project sustainability. Both industry and institutional sources emphasize that effective governance is central to sustaining healthy OSS communities [37, 41]. Governance defines how decisions are made, who has authority, and how responsibilities are distributed.

The governance structures influence contributor retention, diversity, and resilience [20, 50, 54]. Several OSS projects still rely on informal or opaque governance practices, often centered on a few long-standing maintainers [22, 29, 39]. When these governance mechanisms remain undefined, projects risk stagnation, contributor burnout, and a decline in trust. These issues have been recognized as systemic threats to the sustainability of the OSS ecosystem [51].

Despite its recognized importance, OSS governance remains empirically underexplored. Existing guidance often derives from gray literature (e.g., foundation templates, blog posts, and practitioner) [1, 14, 15, 32, 37, 52], rather than from a systematic analysis of real-world governance artifacts.

Empirical understanding of how governance is defined and varies across projects remains limited. Prior work focusing on individual projects, such as the case study of the data.table community [34], provides valuable qualitative insights into the consequences of restructuring projects through governance models, but lacks a cross-project perspective that captures the diversity of governance structures across the OSS landscape.

As a result, we still lack a clear understanding of how authority and responsibility are documented in OSS projects. Without a systematic analysis of these governance documents, the authority distribution and how contributors gain or lose influence within different organizational structures remain unclear.

In this paper, we characterize how OSS projects define, structure, and differentiate roles in their written governance models. Governance artifacts (such as GOVERNANCE.md files) offer an explicit record of how communities describe participation. We focus on these textual traces to understand how projects formalize collaboration, transforming tacit community norms into written structures that define how contributors coordinate their work.

To achieve this, we analyze a sample of GitHub repositories. From each repository, we identify governance files and extract all documented roles into a structured representation grounded in Institutional Grammar (IG) [9, 46, 47]. We then map these roles to a catalog of technical, managerial, and interpersonal skills and combine computational analysis with manual interpretive clustering to compare how communities differentiate their roles.

Our analyses reveal that OSS projects vary widely in how they label and define roles. The same title often carries distinct responsibilities across projects, while similar duties are often referred to under different names. These inconsistencies reveal a persistent misalignment of terminology, a lack of definitions, and overlapping responsibilities. By systematically analyzing these divergences, our study provides both an empirical foundation and practical guidance for maintainers, foundations, and researchers seeking to design models of participation in OSS governance, with the broader aim of fostering healthier OSS communities.

2 Related Work

OSS communities are living examples of large-scale collaboration. Since early studies, OSS has been conceptualized as a form of community of practice [26], where learning, legitimacy, and authority emerge through participation, rather than formal appointment. Early governance models, such as the benevolent dictator paradigm or consensus-based meritocracies, depended on informal social contracts, technical reputation, and sustained contribution instead of codified policy [4, 30, 42].

As OSS projects grow in scale and complexity, empirical studies indicate that informal governance arrangements often become insufficient for ensuring accountability, transparency, and continuity [24, 56]. This leads to the institutionalization of OSS governance, where communities formalize norms and practices into explicit structures like charters, and artifacts to manage coordination challenges and sustain development [7, 11, 15].

2.1 Roles, Pathways, and Belonging in OSS

Understanding governance requires understanding the social trajectories that sustain open source communities. Contributors rarely assume authority immediately. They progress from peripheral participation to more central forms of involvement through a combination of learning, recognition, and sustained interaction [48, 53, 58]. These trajectories are shaped not only by technical expertise, but also by social and communicative engagement, as contributors' progress depends on their ability to build relationships, exchange feedback, and gain recognition within the community [3, 49, 53, 58].

More recent work reframes these challenges as affective rather than procedural. Contributors' sense of belonging and identification with the community was evidenced as a stronger predictor of retention than technical proficiency alone [54]. Collectively, these studies highlight that governance defines the visible and invisible pathways through which contributors participate inside the project.

2.2 The (Lack of) Documented Governance

Despite the extensive body of research describing OSS roles, pathways, and collaboration practices, few studies have examined governance as it is documented in project repositories. Existing work

has mostly characterized how contributors gain authority or how leadership structures evolve, but has rarely investigated the governance artifacts that define these arrangements [24, 36]. Recent work has begun to explore the consequences of formalizing governance models, Oliveira et al. [34] showed how rewriting a project's governance file reshaped decision-making patterns and community health in the data. table ecosystem.

Although multiple studies discuss roles such as maintainers, committers, or core contributors [3, 53], little is known about how such roles (e.g., associated privileges, responsibilities, and decision rights) are formally defined in writing. We still lack an empirical account of what responsibilities and decision structures OSS projects codify, and how these vary across ecosystems.

This study addresses that gap by analyzing governance documents across a diverse sample of OSS projects. Through a large-scale, cross-ecosystem comparison, we reveal how authority, responsibility, and participation are codified in writing, bridging sociotechnical theory, which views governance as both a social process and a technical artifact, with textual evidence to reveal how communities formalize governance in practice.

2.3 Institutional Grammar

Institutional Grammar (IG) provides a structured framework for analyzing how governance rules are expressed in written form. IG conceptualizes institutions as systems of statements that specify who may or must do *what*, *under which conditions*, and *with what consequences* [9]. Rather than treating governance as an abstract structure alone, IG focuses on the language through which authority, permissions, and responsibilities are formally articulated.

To enable systematic analysis, each institutional statement can be decomposed into syntactic components, commonly summarized as Attribute (actor), Deontic (permission, obligation, or prohibition), Aim/Object (action), Conditions (context), and Or-else (sanction). This decomposition makes heterogeneous governance texts comparable by transforming qualitative rules into structured, analyzable elements [47].

In this study, we adopt a lightweight, role-centric interpretation of IG tailored to software projects. Instead of decomposing every sentence at the clause level, we operationalize the grammar into four practical dimensions: scope, privileges, obligations, and promotion/demotion criteria. These dimensions correspond to the Attribute-Deontic-Aim-Condition structure while remaining tractable for large-scale comparative analysis. This adaptation enables consistent extraction and comparison of how projects formalize authority in governance documents.

3 Research Method

We conducted a qualitative analysis of how OSS projects define, structure, and differentiate roles through written governance artifacts. Governance files represent one of the publicly accessible traces of how communities formalize who participates, the responsibilities they have, and under what conditions. We systematically collected governance files from multiple OSS projects on GitHub, coded their contents to identify explicit role definitions, and compared how authority, responsibility, and participation are described across projects.

Following the qualitative approaches discussed by Easterbrook et al. [13], our inductive analysis centers on three interconnected lenses: how governance artifacts **codify roles** and delineate participation; how they **distribute responsibilities and decision rights** across these roles; and how they **differentiate** or blur boundaries between technical, managerial, and symbolic functions.

The research process followed an iterative approach combining exploration, structured extraction, and interpretive analysis. We began by familiarizing ourselves with governance files to identify recurring structures and language. Guided by these observations, we applied systematic data extraction, skill mapping, and successive clustering, refining each step through discussion and validation.

3.1 Researcher Positionality and Reflectivity

This study was conducted by researchers with long-standing involvement in OSS communities and experience studying socio-technical collaboration. All the authors have experience in conducting empirical studies on OSS community governance, onboarding, and sustainability. This insider familiarity offered an interesting understanding of governance language and practices, but also carried the risk of interpretive bias.

To mitigate the potential bias, we adopted a reflexive stance throughout the analysis. Coding and interpretation were discussed collaboratively, with deliberate attention to negative cases and staying close to the textual evidence rather than presumed norms of "good governance." Iterative peer debriefing among the authors helped ensure that decisions about role categorization and skill mapping reflected the actual data. Our goal was not to judge whether projects governed "well", but to understand how authority and responsibility were codified in practice.

3.2 Finding Candidate Projects

Our population consists of OSS projects hosted on GitHub, which represents one of the largest ecosystems for software development and collaboration [16, 17]. To identify suitable candidates for our analysis, we focused on repositories that (i) are publicly available, (ii) explicitly define an open-source license, and (iii) have community recognition as reflected by stargazer count.

Based on these criteria, we collected a license-stratified set of projects on GitHub. Our selection is grounded in the recent license popularity rankings reported by GitHub's Innovation Graph [19]. We included the following top licenses in our sampling: MIT, GPL-2.0, GPL-3.0, Apache-2.0, BSD-2-Clause, BSD-3-Clause, MPL-2.0, LGPL, EPL-2.0, CC0-1.0, and AGPL-3.0. We then retrieved 1,000 repositories licensed under each category, ordered them by stargazer count in descending order, as showed in Table 1.

After the project retrieval, we checked the presence of governance files. Following GitHub's official documentation [18], which recognizes `GOVERNANCE.md` as a file that describes project governance, we restricted our search to files whose names contained the term "governance". The search was executed recursively, at any folder depth within the repository tree, in any file format. We dismissed the projects in which we did not detect a governance file.

Table 1: Projects with Governance Files.

License Type	Projects Collected	Files Detected	Governance Files w/ Content
Apache-2.0	1,000	40 (4.0%)	30 (3.0%)
MIT	1,000	10 (1.0%)	7 (0.7%)
AGPL-3.0	1,000	7 (0.7%)	5 (0.5%)
BSD-3-Clause	1,000	6 (0.6%)	3 (0.3%)
BSD-2-Clause	1,000	3 (0.3%)	3 (0.3%)
MPL-2.0	1,000	3 (0.3%)	3 (0.3%)
LGPL-2.1	1,000	2 (0.2%)	1 (0.1%)
CC0-1.0	1,000	1 (0.1%)	1 (0.1%)
Total	8,000	72 (0.90%)	54 (0.67%)

3.3 Extracting Governance Topics

Once candidate projects were identified, we analyzed the themes in the governance files using a two-stage qualitative coding process following Saldana's guidelines for iterative coding [45]. In the first cycle, we conducted an open reading of each governance file and recorded all distinct topics explicitly mentioned in the text. Examples included references to merging rights, release authority, review obligations, quorum requirements, and funding arrangements. Each element was coded at the level of granularity in which it appeared, preserving the original terminology and scope.

In the second cycle, we compared and clustered related subtopics into broader categories that reflected recurring governance mechanisms. For instance, provisions about merging rights, release procedures, and security disclosures were grouped under a category representing core technical processes. Finally, we consolidated these categories into a smaller set of topics that summarize the main dimensions addressed by governance files across projects. This iterative process yielded six high-level topics that structure our subsequent analysis:

- **Organizational Structure** - how roles are defined, responsibilities distributed, and authority transferred.
- **Decision-Making** - the mechanisms for collective choices, including voting and elections.
- **Core Processes** - technical workflows such as release management, contribution pipelines, and security disclosures.
- **Community and Communication** - norms for interaction, including codes of conduct and official channels.
- **Project Context** - institutional affiliations, licensing, and maturity signals.
- **Compensation Schemes** - financial arrangements, funding sources, and disbursement rules.

By deriving topics through this bottom-up grouping, we ensured that our coding was grounded in the content of governance files rather than in a predefined structure. This method allowed us to compare heterogeneous documents while being sensitive to project-specific terminology. The frequencies for each topic are reported in Table 2. It is important to note that frequencies were calculated at the document level: a topic was considered present in a file if at least one of its associated subtopics appeared in the text.

Table 2: Topics Approached by Governance Documents.

Section	Frequency
Organizational Structure	54 (100%)
Decision Making	53 (98%)
Core Processes	49 (90%)
Community and Communication	43 (80%)
Project Context	32 (59%)
Compensation Schemes	7 (13%)

3.4 Extracting Role Information

Among the six topics, organizational structure emerged as the one documented in all projects and the most directly related to our research questions. Governance files varied in length and emphasis, but all specify, implicitly or explicitly, a set of roles and the distribution of authority among them. We conducted a detailed extraction focused specifically on this dimension.

In this step, we aimed to move from unstructured textual descriptions to structured and comparable records of governance roles. To achieve this, we designed an extraction template grounded in Institutional Grammar [9, 46, 47], a framework for decomposing institutional statements into distinct analytical components. IG models governance as a composition of Attribute (the actor or role), Deontic (the normative modality indicating permissions or obligations), Aim/Object (the activity or domain of action), Conditions (the circumstances or qualifications under which a rule applies), and Or-else (sanctions).

Following this model, we coded each identified role according to four aspects that operationalize these elements in the context of OSS governance.

- **Scope of Responsibility** - The activities, domains, or areas the role is accountable for (e.g., "responsible for releases", "maintains project infrastructure").
- **Privileges** - The explicit powers granted to the role (e.g., merge or commit rights, voting eligibility, authority to approve pull requests, or to represent the project externally).
- **Responsibilities** - The duties or expectations associated with the role (e.g., reviewing contributions, mentoring newcomers, participating in decision-making processes).
- **Promotion and Demotion Criteria** - The rules that determine how individuals acquire or lose a role (e.g., nomination and election, activity thresholds, appointment by a committee, or voluntary resignation).

Each of these aspects captures a different dimension of authority: scope defines what is governed, privileges indicate the levers of power, obligations reveal the expectations attached to authority, and entry/exit criteria regulate access to these positions. Together, they enable us to reconstruct governance as a set of titles and as a structured system of rules for participation and control.

We annotated governance files using this template and coded role definitions only when explicit textual evidence was present. If the role's privilege, obligation, or promotion/demotion criteria were not stated, the corresponding field was marked as "absent" rather than inferred. The output of this step, available in the replication package [35], was a role-centric dataset, where each entry represents a single role instance described in a governance document.

3.5 Mapping Titles to Talents

As our analysis of governance documents progressed, we noted that the role names drifted across projects. The same label could signal very different responsibilities, while different labels could denote nearly identical positions. Terms such as "Owner", "Project Lead", or "Core Developers" were occasionally used to describe almost the same responsibilities. This drift made it problematic to rely on titles alone for comparison. Treating the title as equivalent risked conflating distinct responsibilities, while treating them as different risked ignoring similarities. To resolve this, we decided to move beyond focus on the skills encoded in governance provisions.

For this step, we adapted the open-source skills catalog proposed by Liang et al. [28] to fit our analytical goals. Table 3 presents our adapted version, with the respective IDs for each skill. Each extracted role from the governance files was mapped to this catalog using the evidence we collected, specifically, the scope, privileges, obligations, and promotion/demotion criteria.

Table 3: Skills Catalog Description [28].

ID	Name	ID	Name
Technical Skills (TS)			
T1	Programming	T4	DevOps
T2	Software Engineering	T5	Domain
T3	Technologies	T6	Version Control Systems
Working Styles (W)			
W1	Excellence	W3	Organized
W2	Available		
Problem Solving (P)			
P1	Creative	P3	Analytical
P2	Initiative		
Contribution Type Skills (CT)			
CT1	Bug triaging	CT4	Documentation
CT2	Bug reporting	CT5	Visual Design
CT3	Code Review	CT6	Translation
Project-Specific Skills (PSS)			
PSS1	Purpose	PSS3	Processes
PSS2	Organization		
Interpersonal Skills (IS)			
IS1	Kind	IS4	Giving help
IS2	Communication	IS5	Conflict Resolution
IS3	Asking for help	IS6	Collaboration
External Relations (ER)			
ER1	Stakeholders	ER3	Licenses
ER2	Marketing	ER4	Funding
Management (M)			
M1	Community	M4	Delegating
M2	Project	M5	Time
M3	Planning		
Characteristics (C)			
C1	Adventurous	C6	Reliable
C2	Open-minded	C7	Persevering
C3	Patient	C8	Diligent
C4	Adaptable	C9	Self-aware
C5	Curious		

This reframing allowed us to analyze the distance between roles that shared the same name. For example, two distinct "Maintainer" roles could now be compared in terms of their skill profiles. If their bundles overlapped only partially, the distance between them was large; if they converged on similar competencies, the distance was small. To formalize this comparison, we applied a series of unsupervised clustering experiments using the 45-dimensional binary skill matrix extracted from governance files. Each role instance was represented as a vector of skill presences (Table 3), standardized with scikit-learn's StandardScaler function [27].

We tested multiple algorithms (K-Means, Agglomerative Clustering, and DBSCAN) to identify potential groupings. Role-to-role similarity was computed using Euclidean distance, and cluster coherence was evaluated with Rank-Biased Overlap (RBO) between ranked skill frequency profiles at different thresholds (Figure 1). However, the results showed high sensitivity to parameter choices and low stability across thresholds. At smaller distance cutoffs, agglomerative clustering produced many micro-clusters, while higher thresholds merged roles into broad, uninformative categories.

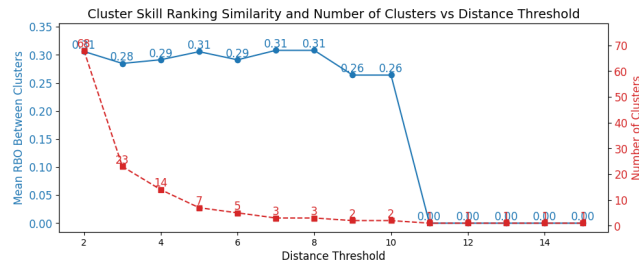


Figure 1: Role Automatic Clustering.

A closer investigation revealed that the reason was that several governance roles were not atomic but composite, aggregating multiple smaller responsibilities that, in other projects, were distributed across distinct roles. For example, one project’s “Maintainer” could simultaneously act as reviewer, release manager, community leader, and triager.

These hybrid roles inflated their skill bundles, causing them to appear artificially similar to multiple categories at once and undermining the convergence of the clustering. Recognizing this limitation, we decided not to overinterpret the unstable clusters and instead advance toward another analytical approach capable of capturing these composite role structures.

3.6 Manual Role Clustering Interpretation

The next step was a manual interpretive clustering process, recognizing that governance roles are complex social artifacts. They are embedded in community norms, historical trajectories, and project-specific practices that resist purely computational classification. Our interpretive clustering unfolded in three steps.

- (1) We revisited the role definitions extracted from governance files together with their mapped skills bundles. Rather than treating skills as independent features to be automatically grouped, we examined how they were expressed in context: what combination of privileges, obligations, and responsibilities they represented within each project.
- (2) We compared roles across projects, looking for overlaps and connections. When two roles showed similarities, we grouped them into the same cluster. We prioritized classification into the most prominent or dominant cluster, then recorded cross-links to secondary or tertiary clusters where overlaps existed. This enabled us to acknowledge ambiguity without fragmenting the dataset into an explosion of micro-categories. For example, a “Maintainer” who also performed triage and release tasks was primarily assigned to the Core Maintainer cluster, but linked to Triage and Release Manager as secondary roles

- (3) We consolidated the assignments into a set of recognizable clusters. These included Contributor, Core Maintainer, Steering/Leadership, Reviewer, Triage, User, and Project-Specific Roles, along with less frequent clusters such as Advocacy or Emeritus positions. The frequency of these categories across projects is reported in Table 4, which reveals both dominant role types (e.g., Contributor and Maintainer) and symbolic or transitional ones (e.g., Emeritus Maintainer).

Table 4: Manual Clustering Result.

CLUSTER	SIZE	CLUSTER	SIZE
Core Maintainer	50	Advocacy	4
Contributor	29	Emeritus Maintainer	4
Steering	20	Triage	3
User	12	Reviewer	2
Project Specific	9	Committer	1
Owner	8		
TOTAL			133

The interpretive clustering was conducted in one intensive session involving three of the authors. Two of them have over a decade of experience studying OSS sustainability, contributing extensive domain knowledge to the process. Each role identified per file was discussed collaboratively, one by one, examining its textual definition, skill composition, and possible alignment within the emerging clusters. Decisions were made only through unanimous agreement among all three researchers, ensuring that each categorization was grounded in shared understanding and justified reasoning.

To support our reasoning, we used a shared whiteboard to visualize the relationships among roles and their functional domains. This visualization served as a living artifact. As discussions unfolded, we mapped hierarchical dependencies and overlapping responsibilities. The evolving diagram enabled us to trace structural connections between roles and reveal latent governance patterns (e.g., similar titles with different responsibilities across projects).

This interpretive approach was more effective, although more time-consuming, than automatic clustering because it could accommodate the composite nature of governance roles. By privileging a primary cluster while preserving cross-links, we balanced comparability with contextual sensitivity. Figure 2 illustrates the resulting organization: roles are arranged into organizational and operational layers, with skill distributions showing both recurrent patterns and overlaps. All role definitions and cluster mappings are provided in our replication package [35] to ensure full traceability between source data and outcomes.

4 Results

We begin by presenting how OSS projects distribute responsibilities and authority across governance roles (Figure 2). Then we detail each governance dimension, unpacking how responsibilities, privileges, and skills are articulated in practice.

4.1 Overlap and Differentiation of Responsibilities

Figure 2 provides a visual map of how governance documents describe the distribution of responsibilities across projects. Each

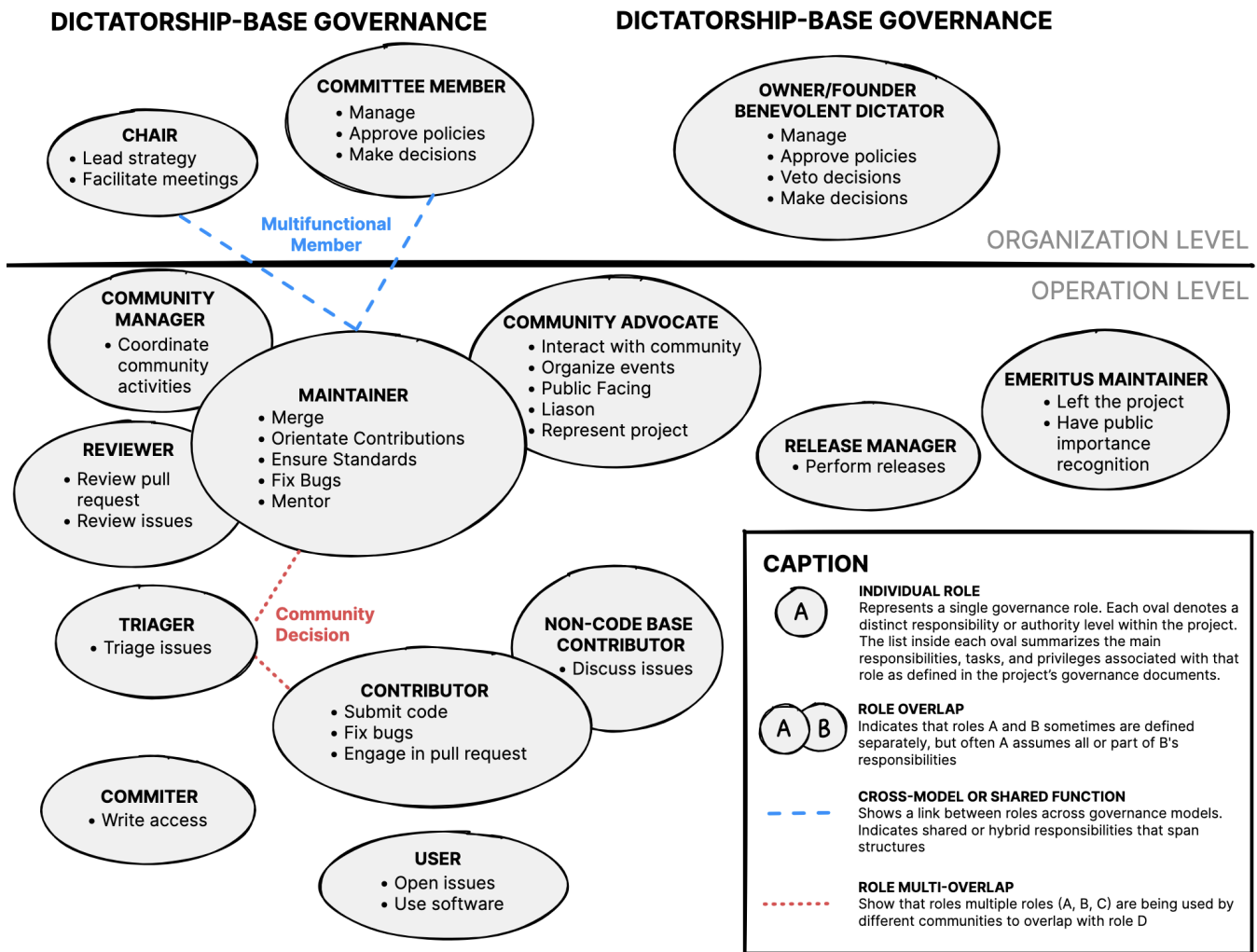


Figure 2: Roles Organization and Skill Distribution.

bubble corresponds to a category derived from our interpretive analysis of governance files. The items inside the bubble summarize the main responsibilities explicitly associated with that role. The overlap/dashed lines between bubbles capture observed overlaps or dependencies across roles, respectively.

To ensure consistency, we classified roles into organizational or operational layers based on their formal decision authority rather than the technicality of their tasks. Roles were labeled organizational when governance documents granted them strategic or institutional powers (e.g., defining policies, voting on decisions, or setting project direction). Roles were labeled operational when they primarily executed, supported, or coordinated work without formal authority over project direction.

What emerges from this visualization is the diversity in how projects separate/combine responsibilities. In some projects, bubbles are distinct and specialized. Reviewers handle only code assessment, triagers manage bug queues, and release managers focus

on distributing versions. Other projects, however, adopt multifunctional roles where bubbles merge.

The maintainer bubble, centrally positioned, often aggregates responsibilities that, in other projects, are distributed across several roles. A maintainer might simultaneously merge contributions, enforce coding standards, triage issues, mentor newcomers, and even act as a community representative.

4.2 What's in a Role? Skills Tell the Story

Figure 3 reveals the distribution of skill categories across governance positions. The heat map shows how often each skill category appears in the definition of different governance roles. Darker shades indicate higher frequencies. The vertical axis lists the skill categories, while the horizontal axis represents governance role types. Each cell reflects the number of occurrences of a skill type for a given role, illustrating which skills are most emphasized across the governance documents.

SKILL FREQUENCY DISTRIBUTION ACROSS OSS GOVERNANCE ROLES

	Advocacy	Committer	Contributor	Core Maintainer	Emeritus Maintainer	Owner	Reviewer	Specific	Steering	Triage	User
Technical Skills		1	21	35				2	1		1
Working Styles	1		3	20		1		2	3		
Problem Solving				3		2					
Contribution Type		1	23	38		2	2	1	1	2	3
Project-Specific Skills		1	19	40		4	1	3	12	3	2
Interpersonal Skills	4	1	21	48		4	2	7	19	2	8
External Relations	2		1	3		3		1	7		4
Management	3		3	29		8		7	20		
Characteristics			1	2		1		1	1		

GOVERNANCE ROLES

Figure 3: Skill Frequency Distribution Across Governance Roles.

4.2.1 Contributors: The Technical Backbone. The contributor role is the broadest and most balanced across technical and interpersonal skills. As shown in the graphs, contributors are commonly associated with programming (T1), bug reporting (CT2), code review (CT3), and documentation (CT4). This mix underscores that contributors may produce artifacts or engage socially through communication (IS2) and collaboration (IS6), often involving one or more of these activities rather than all of them.

For example, the *vitessio/vitess* governance file [55] captures this duality: contributors write code and documentation, are tasked with supporting new users and spreading the project through talks or advocacy. In other words, becoming a contributor is the entry point for participation, carrying productive or social obligations that may be fulfilled through one or more activities stated in the governance document.

Org: vitessio | Repo: vitess

Contributor: Contributors are community members who contribute in concrete ways to the project. [...] In addition to their actions as users, contributors may also find themselves doing one or more of the following: supporting new users (existing users are often the best people to support new users); reporting bugs; identifying requirements; providing graphics and web design; [...]

Similar formulations appear in *bpfftrace/bpfftrace* [6], *apolloconfig/apollo* [2], and *Rdatatable/data.table* [40], where contributors are described as open, entry-level participants engaging through both technical and community-facing activities.

4.2.2 Core Maintainers: The Hybrid Role. No role is more central, or more composite, than core maintainer. Maintainers' profiles span

across programming (T1), software engineering (T2), version control systems (T6), and documentation (CT4), while also emphasizing on management skills (M1, M2, M3) and interpersonal abilities like communication (IS2) and collaboration (IS6).

This breadth reflects their multi-dimensional mandate, ensuring technical quality and leading the community. Maintainers are not just coders. They are guardians of standards, planners of direction and vision, leaders, and often mentors. In the *marimo-team/marimo* project [31], maintainers are (as they describe) the "ultimate authority" over project direction, but their day-to-day may include triaging issues, reviewing pull requests, and maintaining CI pipelines, functions that in other projects are distributed across multiple roles.

Org: marimo-team | Repo: marimo

Core Maintainer: Project Maintainers lead the technical development of the marimo project, and they are the ultimate authority on the direction of the marimo project. [...] This includes triaging issues, proposing and reviewing pull requests, and updating continuous integration as needed. [...]

In *vitessio/vitess* [55], maintainers are entrusted with ongoing development and strategic alignment while remaining accountable through peer review and voting procedures. In *rook/rook* [44], maintainers combine hands-on development with community representation, issue triage, and participation in steering meetings. Likewise, *grafana/tempo* [21] defines maintainers as both technical leads and mediators of consensus.

4.2.3 Committers and Reviewers: The Gatekeepers. Committers and reviewers show a narrower but essential set of skills. On the one side, committers are tightly focused on programming (T1), but also commonly involve management and interpersonal skills, such as coordinating merges, communicating changes, and aligning contributions with the project's goals. Their function is crucial, as they have direct access to the project's resources and can make changes without submitting patches. The *kopia/kopia* project [25] describes this commit-then-review process as efficient for trusted contributors, while their work continues to be reviewed by the community before acceptance.

Org: kopia | Repo: kopia

Committer: Committers are community members who have shown that they are committed to the project's continued development through ongoing engagement with the community. Committership allows contributors to more easily carry on with their project related activities by giving them direct access to the project's resources. That is, they can make changes directly to project outputs, without having to submit changes via patches [...]

Reviewers concentrate almost exclusively on code review (CT3) and analytical skills (P3). Their narrow scope underscores a quality-assurance orientation. The distribution/distribution [12] reviewers' votes are required to merge changes, making them guardians of project quality.

Org: distribution | Repo: distribution

Reviewer: A reviewer is a core role within the project. They share in reviewing issues and pull requests and their LGTM counts towards the required LGTM count to merge a code change into the project. [...]

4.2.4 Steering and Owners: The Strategists. In roles with greater strategic focus, steering and owner roles, technical skills are largely absent. Instead, they concentrate the management (M1, M2, M3), communication (IS2), and external relations (ER1, ER4). These roles embody governance as strategy rather than operation: they set direction, define policies, and coordinate with stakeholders. For example, the crossplane/crossplane [10] Steering Committee role is tasked not with coding, but with "owning the overall charter and direction" of the project, a responsibility reflected in the heavy management skills.

Org: crossplane | Repo: crossplane

Steering Committee: The Crossplane Steering Committee oversees the overall health of the project. Its made up of members that demonstrate a strong commitment to the project with views in the interest of the broader Crossplane community. Their responsibilities include: (i) own the overall charter, vision, and direction of the Crossplane project (ii) define and evolve project governance structures and policies, including project roles and how collaborators become maintainers. [...]

Owners and founders, often associated with the benevolent dictator for life (BDFL) model [33], exhibit skills distributions similar to steering roles, strong in management, communication, and external relations. Unlike steering committees, whose power is collective, founders centralize decision-making. For instance, in the zellij-org/zellij project [57], the BDFL is responsible for strategic decisions on finance and collaborations, retaining the veto power as a safeguard for coordinating community decisions.

Org: zellij-org | Repo: zellij

Benevolent Dictator for Life: [...] is in charge of steering the project and making large decisions regarding finances and collaboration. He will bring such decisions to the group when possible in order to hear all dissenting opinions, but the ultimate decision is his. He also has veto power over decisions made by the group. Since we strive to make decisions by consensus, this power is to be used only as a last resort.

4.2.5 Triage: The Coordination and Production. The triage role shows a narrow skill footprint. The graphs indicate loading on bug triaging (CT1) and communication (IS2). This matches the function of triagers as translators: they take the noisy inflow of bug reports and transform them into structured, actionable work items. The Prometheus-operator/prometheus-operator [38] triage team captures this role precisely. Triagers are given GitHub permissions to adjust issues, allowing developers to focus on actual fixes.

Org: prometheus-operator | Repo: prometheus-operator

Triage Team: Contributors who have the Member role and Triage GitHub permission on the prometheus-operator organization and all projects within it, allowing them to modify GitHub issues and PRs statuses and labels. [...]

4.2.6 Specialized and Symbolic Roles: Community Advocate, Emeritus. Beyond the central clusters, several roles carry symbolic weights. These positions show how projects carve out niches, delegate community-facing duties, or institutionalize memory.

Community Advocate: The public voice. The community advocate role is one of the clearest cases of a non-technical skill profile. As shown in Figure 3, advocate skills are almost exclusively related to communication (IS2), external relations (ER2), and community management (M1). This suggests that advocates are expected to convey the project's identity through various channels, including blogs, social media, events, and partnerships. The Rdatatable/data.table [40] governance exemplifies this orientation. The advocate position is tasked with maintaining a blog, coordinating events, and ensuring visibility.

Org: Rdatatable | Repo: data.table

Community Advocate: Community Engagement Coordinator. An individual who is involved in the project but does not also occupy the Committer or CRAN Maintainer role. In charge of maintaining The Raft blog, preparing Seal of Approval Applications, addressing Code of Conduct violations, and planning social or community events. [...]

Emeritus Maintainers: Memory and Symbolism. Unlike other roles, the Emeritus Maintainer category shows no measurable technical, managerial, or interpersonal skills. Emeritus roles institutionalize recognition, preserving the legacy of past contributors and ensuring continuity of identity even after active involvement has ceased. The grafana/tempo [21] project illustrates this. Here, emeritus status carries symbolic weight, signaling gratitude and respect for past leaders.

Org: grafana | Repo: tempo

Emeritus Maintainer: Emeritus maintainers are former maintainers who no longer work directly on Tempo on a regular basis. We respect their former contributions by giving them the Emeritus Maintainer title. This is honorary

only and confers no responsibilities or rights regarding the Tempo project. [...]

4.2.7 Users: The Silent Majority. Finally, the "User" role appears almost deceptively simple in the graph, with light references to bug reporting (CT2) and communication (IS2). The `bpfftrace/bpfftrace` [6] project captures this ethos: users require no qualifications, but their presence justifies the entire endeavor. In governance terms, users are symbolic placeholders for the external audience that validates the project's value.

Org: Bpfftrace | Repo: bpfftrace

User: Users are community members who have a need for the project. They are the most important members of the community, and without them the project would have no purpose. Anyone can be a user; there are no special requirements. [...]

5 Discussion

The preceding analysis detailed how OSS projects formalize participation through governance artifacts that define roles, responsibilities, and decision rights. In this section, we move from describing these patterns to interpreting what they reveal about organization and authority in open collaboration.

5.1 Governance as Textually Observable Infrastructure

Our study demonstrates that governance in OSS projects is not only enacted through participation but also documented through written artifacts that make authority visible and comparable. The governance files analyzed here define who decides, who acts, and under what conditions, transforming what might otherwise remain tacit community norms into explicit, inspectable rules.

By treating these artifacts as primary data, it is possible to analyze them through their textual form, much like source code or documentation. This perspective makes governance empirically observable and enables cross-project comparison. The method does not claim that written documents capture the entirety of practice, but rather that they provide a stable layer where organizational logic becomes legible.

5.2 The Drift Between Codification and Practice

While governance artifacts aim to clarify how participation is structured, our analysis reveals significant variation and ambiguity in how roles are described across projects. The same role title can denote very different responsibilities, while distinct titles often encode nearly identical duties. Some governance files omit key elements, such as promotion criteria or voting rules, leaving authority boundaries partially defined.

5.3 Governance as Layered System

Our results show that governance in OSS projects tends to organize responsibilities into two broad layers. The organizational layer

includes roles such as Owners, Chairs, and Steering Committees, which concentrate on strategic coordination, decision-making, and alignment with external stakeholders. The operational layer includes Contributors, Reviewers, and Triagers, who carry out the daily technical work.

This division appears directly in governance documents, where responsibilities are grouped by scope rather than by seniority. However, the same documents also show an overlap and interdependence between layers. Maintainers, in particular, frequently act as the connective tissue between strategic and operational domains, embodying both coordination and production.

5.4 The Maintainer Paradox

Across projects, governance documents consistently frame maintainers as technical stewards while also assigning community-facing duties. This hybridization of authority produces what we call the *Maintainer Paradox*: governance artifacts centralize power in those meant to distribute it. Our skill-mapping data reveals maintainers as the most multidimensional role, spanning technical, managerial, and interpersonal domains.

This observation reinforces concerns about bottlenecks and burnout in OSS leadership [22, 39]. By coupling organizational and operational authority within a small group, projects may embed dependency, burnout and turnover into their governance.

5.5 Implications for Governance Design

Our analysis reveals how governance documents encode expectations, authority, and participation within OSS projects. From these observations, several implications emerge for those designing or revising governance models. First, projects should **make responsibilities explicit**. Governance files frequently describe similar functions under different titles, which can obscure accountability and confuse contributors. Clarifying what each role entails improves shared understanding within the community.

Second, projects need to be **aware of composite responsibilities**. Maintainers often accumulate multiple responsibilities, where the concentration of duties may hide dependencies and potential burnout. Documenting these overlaps makes workload more visible to community. Third, the **inclusion of symbolic and communicative roles** underscores that governance extends beyond code production. These positions reflect contributions that sustain community identity, cohesion, and continuity. Acknowledging these roles within governance structures reinforces the social dimension of sustainability.

Third, this work suggests that governance artifacts themselves constitute a **rich and underutilized source of empirical evidence**. By treating governance as textually observable infrastructure, researchers can systematically compare how authority and responsibility are formalized across projects. This perspective complements interview- or activity-based studies by providing a stable, inspectable layer where institutional logic is explicitly recorded.

Beyond practice, our findings contribute methodologically and theoretically to the study of governance in open collaboration. By introducing a role-centric analytical lens grounded in Institutional

Grammar, we demonstrate how governance rules can be systematically decomposed into scopes, privileges, obligations, and transition criteria. This structured representation enables consistent comparison across heterogeneous projects, moving analysis beyond informal titles or anecdotal accounts. By applying this approach across repositories, licenses, and ecosystems, our study provides comparative evidence that supports theory-building about how OSS communities formalize authority, distribute responsibilities, and evolve their institutional structures over time.

6 Study Design Trade-Offs and Limitations

Following Robillard et al. [43], we report the key design decisions that shaped this study, the alternatives we considered, and their implications for interpretation. This approach highlights deliberate methodological reasoning and transparency instead of treating limitations as methodological flaws.

Our first major decision concerned the sampling frame and repository ranking. We stratified repositories by license type and ranked them by the number of GitHub stars within each category. Other sampling strategies (e.g., random or activity-based selection, contributor thresholds, or curated foundation lists) were considered but ultimately set aside. License stratification ensured broad ecosystem coverage, while star counts provided a reproducible, API-accessible indicator of visibility.

However, as Borges and Valente [5] note, stars often reflect social signaling or appreciation rather than governance maturity. This means our dataset may over-represent socially prominent projects, and our findings should be interpreted as characterizing highly-visible communities that codify their governance structures, rather than all OSS projects.

A second decision concerned how we detected governance documents. We chose a narrow, reproducible heuristic, automatically identifying files whose names contained the term "governance" at any depth in the repository tree. Broader heuristics, or manual exploration of documentation sites and wikis, could have increased recall but by prioritizing a single, transparent rule, we ensured that detection remained verifiable and aligned with GitHub's "community health file" convention. The trade-off is that we likely excluded projects that govern informally or host their rules outside the repository.

We also made a key decision about the scope of textual evidence to analyze. We restricted our extraction to explicit statements found in governance files, marking elements such as privileges or responsibilities as absent when not directly expressed. Broader approaches (e.g., triangulating with issue discussions or interviews) could have captured tacit norms, but would have required substantial additional effort and time for data collection and interpretation. Our results, therefore, characterize governance as it is written, not necessarily as it is enacted.

In representing governance roles, we adopted a schema inspired by Institutional Grammar [9, 46, 47], decomposing each role into its scope, privileges, obligations, and promotion/demotion criteria. We considered looser thematic approaches and matrix-style responsibility models, but opted for the Institutional Grammar [9, 46, 47] structure because it enables comparability across projects and aligns with rule-based representations of authority. While this schema

standardizes analysis, it may fragment holistic responsibilities or understate the nuances of hybrid roles.

Finally, we initially applied unsupervised clustering to group roles, but found that composite or overlapping roles produced unstable results. We therefore pivoted to manual interpretive clustering, conducted collaboratively among authors and refined through consensus discussions. This increased interpretive judgment but allowed a richer understanding of how projects blend technical and organizational expertise. Our analysis emphasizes descriptive and comparative interpretation; we deliberately avoid linking governance forms to project outcomes such as retention or success, as they would require different data and design strategies.

7 Conclusion

This study examined how OSS projects define, structure, and differentiate roles through their written governance artifacts. By treating governance not as a static organizational chart but as a living textual system, we revealed how communities define authority, responsibility, and participation in writing. Governance files such as `GOVERNANCE.md` serve as socio-technical blueprints that document institutional arrangements that make participation pathways visible.

Through a multi-project analysis, we identified recurrent patterns in how roles are articulated and responsibilities distributed. Projects consistently distinguish between organizational and operational tiers of work, but the distinction often overlaps in practice, especially through composite roles, such as the Maintainer. These hybrid roles embody what we termed the Maintainer Paradox. Other findings exposed symbolic and communicative functions that extend governance beyond code to encompass identity, legitimacy, and continuity.

Beyond the descriptive patterns, our results show that governance artifacts materialize the community's evolving balance between autonomy and structure. Written rules shape participation by defining what is legitimate to do and who is authorized to decide. In this sense, governance operates as a textual infrastructure that organizes collaboration without necessarily centralizing it.

8 Replication Package

To foster transparency, we provide a replication package with all the materials used in this study [35]. The package includes the complete dataset of governance files, role extractions, coding templates, and analytical scripts we used to generate the results presented in this paper. Instructions for reproducing the study are included in a `README.md` file. The dataset and code are available to encourage reuse and extension by the community.

Acknowledgments

This work was supported by the National Science Foundation grant #2303612. Also, CNPq grants #314797/2023-8 and #443934/2023-1

References

- [1] Apache Software Foundation. 2023. ASF Governance Model. <https://www.apache.org/foundation/governance/>. Accessed 2024-04-15.
- [2] Apolloconfig/apollo GitHub Project. 2025. Apollo. <https://github.com/apolloconfig/apollo>. Accessed: 22 October 2025.

- [3] Ann Barcomb, Andreas Kaufmann, Dirk Riehle, Klaas-Jan Stol, and Brian Fitzgerald. 2018. Uncovering the periphery: A qualitative survey of episodic volunteering in free/libre and open source software communities. *IEEE Transactions on Software Engineering* 46, 9 (2018), 962–980.
- [4] Mirko Boehm. 2019. The emergence of governance norms in volunteer-driven open source communities. *IFOSS L. Rev.* 11 (2019), 3.
- [5] Hudson Borges and Marco Tulio Valente. 2018. What's in a GitHub star? understanding repository starring practices in a social coding platform. *Journal of Systems and Software* 146 (2018), 112–129.
- [6] Bpffrace/bpffrace GitHub Project. 2025. Bpffrace. <https://github.com/bpffrace/bpffrace>. Accessed: 22 October 2025.
- [7] Mahasweta Chakraborti, Curtis Atkisson, Ștefan Stănculescu, Vladimir Filkov, and Seth Frey. 2024. Do We Run How We Say We Run? Formalization and Practice of Governance in OSS Communities. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 923, 26 pages. doi:10.1145/3613904.3641980
- [8] Annamaria Conti and Laura Huang. 2024. Open-Source Software Creators: It's Not Just About the Money. NBER Digest.
- [9] Sue ES Crawford and Elinor Ostrom. 1995. A grammar of institutions. *American political science review* 89, 3 (1995), 582–600.
- [10] crossplane/crossplane GitHub Project. 2025. Crossplane. <https://github.com/crossplane/crossplane>. Accessed: 22 October 2025.
- [11] Benoit Demil and Xavier Lecocq. 2006. Neither market nor hierarchy nor network: The emergence of bazaar governance. *Organization studies* 27, 10 (2006), 1447–1466.
- [12] distribution/distribution GitHub Project. 2025. Distribution. <https://github.com/distribution/distribution>. Accessed: 22 October 2025.
- [13] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering*. Springer London, London, 285–311.
- [14] Eclipse Foundation. 2023. Eclipse Projects Handbook: Governance. <https://www.eclipse.org/projects/handbook/>. Accessed 2024-04-15.
- [15] Nadia Eghbal. 2016. Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure. <https://www.fordfoundation.org/learning/library/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/>.
- [16] GitHub. 2023. 100 Million Developers and Counting. <https://github.blog/news-insights/company-news/100-million-developers-and-counting/>. Accessed: October 21, 2025.
- [17] GitHub. 2025. About GitHub – Where people build software. <https://github.com/about>. Accessed: October 21, 2025.
- [18] GitHub. 2025. Creating a default community health file. <https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/creating-a-default-community-health-file>. Accessed: October 21, 2025.
- [19] GitHub. 2025. License Rankings Globally. <https://innovationgraph.github.com/global-metrics/licenses> Top Used OSS Licenses.
- [20] Sean P Goggins, Matt Germonprez, and Kevin Lumbard. 2021. Making open source project health transparent. *Computer* 54, 8 (2021), 104–111.
- [21] Grafana/tempo GitHub Project. 2025. Tempo. <https://github.com/grafana/tempo>. Accessed: 22 October 2025.
- [22] Mariam Guizani, Amreeta Chatterjee, Bianca Trinkenreich, Mary Evelyn May, Geraldine J Noa-Guevara, Liam James Russell, Griselda G Cuevas Zambrano, Daniel Izquierdo-Cortazar, Igor Steinmacher, Marco A Gerosa, et al. 2021. The long road ahead: Ongoing challenges in contributing to large OSS organizations and what to do. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–30.
- [23] Manuel Hoffmann, Frank Nagle, and Yanuo Zhou. 2024. The Value of Open Source Software.
- [24] Chris Jensen and Walt Scacchi. 2010. Governance in open source software development projects: A comparative multi-level analysis. In *IFIP International Conference on Open Source Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 130–142.
- [25] kopia/kopia GitHub Project. 2025. Kopia. <https://github.com/kopia/kopia>. Accessed: 22 October 2025.
- [26] Jean Lave and E Wenger. 1991. Learning in doing: Social, cognitive, and computational perspectives. *Situated learning: Legitimate peripheral participation* 10 (1991), 109–155.
- [27] Scikit learn Developers. 2025. sklearn.preprocessing.StandardScaler — scikit-learn documentation. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>. Accessed: 2026-01-25.
- [28] Jenny T Liang, Thomas Zimmermann, and Dena Ford. 2022. Understanding skills for OSS communities on GitHub. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Association for Computing Machinery, New York, NY, USA, 170–182.
- [29] Johan Linäker, Georg Link, and Kevin Lumbard. 2024. Sustaining maintenance labor for healthy open source software projects through human infrastructure: A maintainer perspective. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. Association for Computing Machinery, New York, NY, USA, 37–48.
- [30] Johan Linäker, Björn Regnell, and Daniela Damian. 2019. A Community Strategy Framework—How to obtain influence on requirements in meritocratic open source software communities? *Information and Software Technology* 112 (2019), 102–114.
- [31] marimo-team/marimo GitHub Project. 2025. Marimo. <https://github.com/marimo-team/marimo>. Accessed: 22 October 2025.
- [32] Dave Neary, Josh Berkus, Katrina Novakovic, and Bryan Behrenshausen. 2020. Understanding Open Source Governance Models. <https://www.redhat.com/en/blog/understanding-open-source-governance-models>. Red Hat Blog.
- [33] Dave Neary, Josh Berkus, Katrina Novakovic, and Bryan Behrenshausen. 2020. Understanding Open Source Governance Models. <https://www.redhat.com/en/blog/understanding-open-source-governance-models>. Red Hat Blog. Accessed: 2026-01-25.
- [34] Pedro Oliveira, Doris Amoakohene, Toby Hocking, Marco Gerosa, and Igor Steinmacher. 2025. Governance Matters: Lessons from Restructuring the data.table OSS Project. In *2025 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, Piscataway, NJ, USA, 1–12.
- [35] Pedro Oliveira, Tayana Conte, Marco Gerosa, and Igor Steinmacher. 2025. Governance in Practice: How Open Source Projects Define and Document Roles. Zenodo. doi:10.5281/zenodo.17430401 Dataset, mining and analysis scripts.
- [36] Siobhán O'mahony and Fabrizio Ferraro. 2007. The emergence of governance in an open source community. *Academy of Management Journal* 50, 5 (2007), 1079–1106.
- [37] OSPO Alliance. 2023. Good Governance Initiative Handbook v1.2. https://ospo-alliance.org/docs/ggi_handbook_v1.2.pdf. Accessed: October 21, 2025.
- [38] prometheus-operator/prometheus-operator GitHub Project. 2025. Prometheus-operator. <https://github.com/prometheus-operator/prometheus-operator>. Accessed: 22 October 2025.
- [39] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. 2020. Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. Association for Computing Machinery, New York, NY, USA, 57–60.
- [40] Rdataatable/data.table GitHub Project. 2025. Data.table. <https://github.com/Rdataatable/data.table>. Accessed: 22 October 2025.
- [41] Red Hat. 2022. The State of Enterprise Open Source: A Red Hat Report. <https://www.redhat.com/rhdc/managed-files/rh-enterprise-open-source-report-f31123-202202.pdf>. Accessed: October 21, 2025.
- [42] Dalia Ritvo, Kira Hessekiel, and Christopher Bavitz. 2017. Challenges & opportunities concerning corporate formation, nonprofit status, & governance for open source projects. *Berkman Klein Center Research Publication* 3 (2017), 17–31.
- [43] Martin P Robillard, Deeksha M Arya, Neil A Ernst, Jin LC Guo, Maxime Lamothe, Mathieu Nassif, Nicole Novielli, Alexander Serebrenik, Igor Steinmacher, and Klaas-Jan Stol. 2024. Communicating study design trade-offs in software engineering. *ACM Transactions on Software Engineering and Methodology* 33, 5 (2024), 1–10.
- [44] Rook/rook GitHub Project. 2025. Rook. <https://github.com/rook/rook>. Accessed: 22 October 2025.
- [45] Johnny Saldaña. 2021. Coding techniques for quantitative and mixed data. In *The Routledge reviewer's guide to mixed methods analysis*. Routledge, New York, 151–160.
- [46] Anamika Sen, Curtis Atkisson, and Charlie Schweik. 2022. Cui bono? Do open source software incubator policies and procedures benefit the projects or the incubator? *International Journal of the Commons* 16, 1 (2022), 64–77.
- [47] Saba Siddiki, Christopher M Weible, Xavier Basurto, and John Calanni. 2011. Dissecting policy designs: An application of the institutional grammar tool. *Policy Studies Journal* 39, 1 (2011), 79–103.
- [48] Yunlong Song. 2021. *Role Classification and Transition of OSS Developers*. Master's thesis. University of California, Irvine.
- [49] Igor Steinmacher, Marco Aurélio Gerosa, and David Redmiles. 2014. Attracting, onboarding, and retaining newcomer developers in open source software projects. In *Workshop on Global Software Development in a CSCW Perspective*, Vol. 16. Association for Computing Machinery, New York, NY, USA, 20.
- [50] Igor Steinmacher, Christoph Treude, and Marco Aurélio Gerosa. 2018. Let me in: Guidelines for the successful onboarding of newcomers to open source projects. *IEEE Software* 36, 4 (2018), 41–49.
- [51] The Linux Foundation. 2023. Rising Tides of Open Source: Linux Foundation Annual Report 2023. https://www.linuxfoundation.org/hubs/Reports/2023_lf_annual_report_122123a.pdf. Accessed: October 21, 2025.
- [52] TODO Group. 2023. Open Source Governance Models. <https://github.com/todogroup/governance>. Accessed 2024-04-15.
- [53] Bianca Trinkenreich, Mariam Guizani, Igor Wiese, Anita Sarma, and Igor Steinmacher. 2020. Hidden figures: Roles and pathways of successful OSS contributors. *Proceedings of the ACM on human-computer interaction* 4, CSCW2 (2020), 1–22.
- [54] Bianca Trinkenreich, Klaas-Jan Stol, Anita Sarma, Daniel M German, Marco A Gerosa, and Igor Steinmacher. 2023. Do i belong? modeling sense of virtual

- community among linux kernel contributors. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, Piscataway, NJ, USA, 319–331.
- [55] vitessio/vitess GitHub Project. 2025. Vitess. <https://github.com/vitessio/vitess>. Accessed: 22 October 2025.
- [56] Likang Yin, Mahasweta Chakraborti, Yibo Yan, Charles Schweik, Seth Frey, and Vladimir Filkov. 2022. Open source software sustainability: Combining institutional analysis and socio-technical networks. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (2022), 1–23.
- [57] zellij-org/zellij GitHub Project. 2025. Zellij. <https://github.com/zellij-org/zellij>. Accessed: 22 October 2025.
- [58] Minghui Zhou and Audris Mockus. 2014. Who will stay in the floss community? modeling participant’s initial behavior. *IEEE Transactions on Software Engineering* 41, 1 (2014), 82–99.