International Journal of the Commons

Sustaining Open Source as a Digital Commons: Evidence from the Apache Software Foundation (ASF) Incubator Program

RESEARCH ARTICLES

LIN ZHOU
CHARLES SCHWEIK
CURTIS ATKISSON
BRENDA BUSHOUSE
MAHASWETA CHAKRABORTI
SETH FREY
ȘTEFAN STĂNCIULESCU
LIKANG YIN
VLADIMIR FILKOV

*Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

Open source software (OSS) now underpins critical digital infrastructure, yet many projects struggle to sustain the volunteer effort, technical maintenance, and governance work required to persist over time. This paper develops a Socio-Technical-Governance (STG) framework to explain OSS sustainability as a digital commons problem in which social dynamics, technical production, and rule-based governance jointly shape outcomes. Building on socio-technical systems scholarship in Software Engineering and Ostrom's Institutional Analysis and Development (IAD) tradition in the social sciences, the framework conceptualizes OSS communities as nested action situations governed by working rules that allocate decision authority, distribute costs and benefits, and structure participation. We illustrate the framework's utility through an integrative synthesis of previously published evidence from the Apache Software Foundation (ASF) Incubator, leveraging an ecosystem of digital trace data (e.g., version control, issue tracking, and mailing-list discussions) and five previously published studies of the ASF Incubator: a shared longitudinal dataset covering 269 incubator projects, and four empirical analyses that draw overlapping subsets from it. Across technical, social, and governance dimensions, the four source studies report (1) early and sustained growth in technical activity as a distinguishing feature of projects that graduate, alongside code-quality signals associated with retirement; (2) mentorship configuration as a meaningful social correlate of outcomes; (3) governance patterns consistent with cost–benefit balancing at the program level in ASF's Incubator context; and (4) measurable differences in institutional discourse and socio-technical network trajectories that emerge months before final outcomes. The STG Framework offers a theory-grounded approach for comparing governance arrangements across OSS contexts and for developing candidate indicators to support earlier diagnosis and intervention in at-risk projects.

CORRESPONDING AUTHOR:
Lin Zhou
University of Massachusetts Amherst, United States
linzhou@umass.edu

KEYWORDS:
Open source software; Digital commons; Socio-technical-governance (STG) framework; Sustainability and project outcomes; Apache Software Foundation

TO CITE THIS ARTICLE:
Zhou, L., Schweik, C., Atkisson, C., Bushouse, B., Chakraborti, M., Frey, S., Stănciulescu, Ș., Yin, L., & Filkov, V. (2026). Sustaining Open Source as a Digital Commons: Evidence from the Apache Software Foundation (ASF) Incubator Program. *International Journal of the Commons*, 20(1), pp. 576–595. DOI: https://doi.org/10.5334/ijc.1727

1. INTRODUCTION

Open source software (OSS) represents one of the most significant forms of digital commons in the contemporary era. While the concept of "commons" traditionally referred to shared common-property natural resources such as forests, fisheries, and irrigation systems, the digital age has given rise to new forms of commons that exist in the realm of information and software (Hess and Ostrom, 2007; Schweik and English, 2012, 2013; Frischmann, Madison and Strandburg, 2014). OSS projects, where source code is openly accessible and modifiable under specific licensing arrangements, exemplify this transformation of commons into the digital sphere. OSS communities face commons-like governance challenges because the inputs required to produce and maintain the shared codebase—developer time, attention, review capacity, infrastructure, and decision authority over changes—are scarce and often rivalrous. As a result, sustaining OSS depends on how communities craft and enforce rules that allocate responsibilities and coordinate contributions over time.

The importance of OSS to global digital infrastructure cannot be overstated. Active and sustained OSS projects dominate the Internet and other critical operations, with prominent examples including the Apache web server, the Linux operating system, Android mobile platform, and LibreOffice productivity suite. Beyond these well-known projects, OSS libraries and frameworks underpin virtually every sector of the modern economy, from financial services to healthcare, from telecommunications to government operations. Surveys indicate that open source has become a best practice among large companies (Hecht and Clark, 2018), with organizations increasingly relying on OSS not merely as a cost-saving measure but as a strategic approach to software development that enables innovation, interoperability, and community-driven quality assurance.

However, despite this widespread adoption and critical importance, OSS projects face a significant sustainability challenge that threatens the stability of the digital infrastructure upon which modern society depends. Large-scale empirical work has long documented that abandonment, rather than sustained development, is a common fate for OSS projects: in the first population-scale study of its kind, Schweik and English (2012) classified over 100,000 SourceForge-hosted projects into successful and abandoned trajectories and found abandonment to be widespread, including among projects that never produced a first release. Nor is this vulnerability confined to obscure or hobbyist efforts. Studying 104 failed projects drawn from the 5,000 most popular repositories on GitHub, Coelho and Valente (2017) document nine recurring reasons for failure, most commonly being usurped by a stronger competitor, functional obsolescence, and lack of time or interest of the main contributor, and report that 13% of these highly popular repositories showed no commit activity in the preceding year. These findings reflect the reality that while it is relatively easy to initiate an OSS project, maintaining it over time—recruiting and retaining contributors, managing technical debt,[1] responding to security vulnerabilities, and adapting to changing user needs—requires sustained collective effort that many projects fail to mobilize. Given the global reliance on OSS coupled with how commonly projects are abandoned, as well as growing concerns around maintenance and security of OSS components, ensuring the sustainability of OSS-based digital infrastructure represents a pressing grand challenge for researchers, practitioners, and policymakers alike.

Understanding OSS sustainability requires grappling with the inherent complexity of these projects, which exist at the intersection of technology, community, and governance. OSS development involves three interconnected dimensions that must be analyzed in conjunction rather than in isolation. First, there is a *social dimension* encompassing the communities of contributors who participate in development, ranging from individual volunteers motivated by learning or reputation to paid employees of corporations and government agencies with strategic interests in the software. These contributors bring diverse skills, motivations, and expectations, and their interactions—patterns of communication, collaboration, and sometimes conflict—shape project trajectories. Second, there is a *technical dimension* involving the source code itself, the programming languages employed, version control systems, testing frameworks, and other infrastructure that enables distributed collaborative development. The architecture of the software, its modularity, documentation quality, and technical accessibility all influence the ease with which new contributors can join and existing contributors can work effectively. Third, there is a *governance dimension* consisting of the rules, norms, and decision-making processes that structure how participants interact and make collective choices about the project's direction (Schweik and English, 2012). Governance encompasses both formal mechanisms such as licensing arrangements and voting procedures and informal norms around code review, communication etiquette, and conflict resolution.

This paper addresses two primary research questions: (1) how can socio-technical systems and institutional analysis be integrated into a common framework for studying OSS sustainability as a digital commons problem? (2) how can the scientific study of OSS projects be organized to provide useful and practical insights for OSS communities

seeking to ensure that their projects are sustained? This methodological question concerns the development of an analytical framework that can guide interdisciplinary research and translate findings into actionable guidance for practitioners.

To address these questions, we introduce the Socio-Technical-Governance (STG) Framework that integrates perspectives from the fields of Software Engineering and Institutional Analysis. The framework combines the Socio-Technical Systems (STS) approach, which has a long history in organizational studies and has been applied extensively in software engineering research, with the Institutional Analysis and Development (IAD) Framework developed by Elinor Ostrom and colleagues to study governance of common-pool resources. By combining these complementary perspectives, the STG Framework provides a comprehensive lens for analyzing OSS projects as complex adaptive systems where technical artifacts, social relationships, and governance structures co-evolve over time. We then use this framework to organize and critically synthesize previously published evidence from the Apache Software Foundation Incubator (ASFI), drawing on multiple empirical studies that examine different components of the framework using a dataset of 269 projects that have passed through the incubation process.

This study is an integrative framework and synthesis paper rather than a new empirical study. Existing research has established OSS as a digital commons and has separately applied socio-technical and institutional perspectives to OSS governance. Here we significantly extend the STG Framework, the skeleton of which was posited in a companion paper (Yin et al., 2026). The distinctive contribution of the STG Framework is threefold: it integrates social, technical, and nested governance attributes within a common action-situation model; it distinguishes digitally observable micro-outcomes from longer-term macro-outcomes; and it provides a common architecture for connecting heterogeneous empirical studies across levels and over time. The previously published ASF studies are used to illustrate and critically assess this architecture, not as new empirical findings of the present article.

The paper proceeds as follows. Section 2 reviews the theoretical foundations underpinning the STG Framework, including the Socio-Technical Systems approach from organizational studies and software engineering and the Institutional Analysis and Development Framework from political science and economics. Section 3 presents the STG Framework in detail, explaining its core dimensions, the concept of action situations adapted from IAD, and the feedback loops that connect different levels of analysis. Section 4 describes the Apache Software Foundation and its incubation program as an empirical context, explaining why it provides an ideal setting for studying OSS sustainability. Section 5 synthesizes evidence from five interrelated, previously published source studies and maps them retrospectively onto the STG Framework, covering technical metrics, social attributes including mentorship, governance structures, and the integration of all three dimensions through computational methods. Section 6 discusses the implications of these findings for understanding OSS as a digital commons and for practitioners seeking to enhance project sustainability and concludes with reflections on the contributions of this research and directions for future work.

2. THEORETICAL FOUNDATIONS

2.1 SOCIO-TECHNICAL SYSTEMS

The Socio-Technical Systems (STS) approach originated in organizational studies during the 1950s, emerging from research at the Tavistock Institute of Human Relations in London that examined the relationship between social and technical components in industrial work settings (Rice, 1953; Trist, 1981). The foundational insight of STS is that organizations and work systems consist of two interdependent subsystems: a social system comprising the people who do the work, their relationships, skills, and interactions, and a technical system comprising the tools, technologies, equipment, and infrastructure used to accomplish work tasks. The key theoretical claim is that optimal performance emerges from the joint optimization of both subsystems rather than from optimizing either in isolation. Attempts to redesign technical systems without attention to social implications, or to restructure social arrangements without considering technical constraints, tend to produce suboptimal outcomes and unintended consequences.

In the field of software engineering, the STS approach has been applied extensively to understand how software development occurs through the interaction of social dynamics and technical artifacts (Ducheneaut, 2005). Researchers have examined how developer networks—patterns of communication, collaboration, and code co-authorship—relate to software architecture and quality. Studies have traced how code structures and developer social structures co-evolve over time, with changes in one domain triggering or constraining changes in the other. This socio-technical perspective has proven particularly valuable for studying OSS, where development is distributed across geographically dispersed contributors who coordinate primarily through digital communication channels such as mailing lists, chat systems, and code

review platforms, and through shared code repositories that serve as both technical infrastructure and social coordination mechanisms.

From an STS perspective applied to OSS, the social system includes the network of developers, users, and other stakeholders who participate in or are affected by the project. These participants occupy various roles—core developers with commit access, occasional contributors who submit patches, users who report bugs and request features, documentation writers, translators, and community managers—and interact through multiple channels. The patterns of these interactions, who communicates with whom and how frequently, can be analyzed as social networks that reveal the structure of collaboration and information flow. The technical system includes the source code, programming languages, libraries, development tools, testing frameworks, and collaboration infrastructure such as version control systems and issue trackers. These technical elements both constrain and enable particular forms of social interaction; for example, modular code architectures may facilitate parallel development by independent contributors, while monolithic designs may require tighter coordination.

2.2 THE INSTITUTIONAL ANALYSIS AND DEVELOPMENT FRAMEWORK

The Institutional Analysis and Development (IAD) Framework was initially introduced by Ostrom (1982) and subsequently refined and operationalized by a large number of social scientists over several decades (see Ostrom, 2009 for a comprehensive treatment). The original goal was to help integrate work undertaken by political scientists, economists, anthropologists, lawyers, sociologists, psychologists, and others interested in how institutions affect the incentives confronting individuals and their resultant behavior (Ostrom, 2009: 9). Institutions are defined as the prescriptions—rules, norms, and shared strategies—that humans use to organize all forms of repetitive and structured interactions. Understanding how institutional arrangements emerge, how they are maintained and changed, and how they shape individual and collective outcomes has been a central concern of the IAD research program.

The IAD Framework identifies several key components and their relationships. Exogenous variables that shape the context of action include biophysical and material conditions (the characteristics of the resources or goods being produced or managed), attributes of the community (such as shared norms, social capital, and demographic characteristics), and rules-in-use (the working rules that actually govern behavior as opposed to formal rules that may exist on paper but not in practice). These exogenous factors shape the structure of action situations—the social spaces where individuals interact, make decisions, and produce outcomes. The Action Situation is the core analytical unit, where participants in particular positions take actions based on available information, expecting particular outcomes to which costs and benefits are attached. The patterns of interaction within action situations produce outcomes, which then feed back to influence the exogenous variables, potentially changing biophysical conditions, community attributes, or rules over time. This feedback structure means that the IAD Framework conceives of institutional systems as dynamic and evolving rather than static.

The IAD Framework has been particularly influential in studying common-pool resources—resources that are difficult to exclude people from using but where one person's use subtracts from what is available to others—and understanding how communities develop institutional arrangements to govern shared resources sustainably (Ostrom, 1990). Ostrom's research on commons governance demonstrated that, contrary to the predictions of tragedy of the commons theory, communities around the world have developed diverse and often highly effective institutional arrangements for managing shared resources without resort to either privatization or state control. Key Ostrom (1990) Design Principles associated with successful commons governance include clearly defined boundaries, congruence between rules and local conditions, collective choice arrangements allowing participation of affected parties, monitoring mechanisms, graduated sanctions for rule violations, conflict resolution mechanisms, and recognition of rights to organize by higher-level authorities.

Whether these principles travel beyond the settings that produced them, however, remains contested. Madison, Frischmann and Strandburg (2010) caution against treating them as a ready-made template for knowledge and digital commons: because cultural commons are deliberately constructed against a backdrop of intellectual property law rather than found in nature, and because their resources must be created before they can be shared, inquiry should proceed inductively through structured case studies rather than by transposing frameworks developed for natural resource settings—a critique that helped motivate the Governing Knowledge Commons framework introduced below. Hess and Ostrom (2007) likewise stress the distinct provision and enclosure problems that knowledge commons raise.

The application of IAD to OSS builds on the recognition that OSS projects constitute a form of commons—shared resources that are neither purely private goods controlled by individual owners nor purely public goods provided by the state (Schweik and English, 2012). OSS projects involve

collective production of a shared resource (the software) by a community of contributors who must coordinate their efforts and manage potential conflicts. The open-access nature of OSS means that the code is available for anyone to use, but the community must still develop and maintain institutional arrangements governing who can contribute, how contributions are evaluated and integrated, how decisions about the project's direction are made, and how disputes are resolved. Another IAD-derived analytical framework, the Governing Knowledge Commons (GKC) framework, extends this analysis specifically to knowledge resources and information goods, providing additional conceptual tools for analyzing how communities construct and maintain knowledge commons including software (Frischmann, Madison and Strandburg, 2014). Both IAD and GKC inform what we present next.

The commons tradition is not the only body of scholarship on OSS governance that informs the framework. A parallel stream of organizational research asks how governance in OSS communities emerges rather than how it is designed. O'Mahony and Ferraro's (2007) longitudinal study of Debian traces a community that constructed a formal, bureaucratic basis of authority and then deliberately limited it through democratic mechanisms, such that governance evolved with members' changing conceptions of legitimate authority. Surveying this literature, de Laat (2007) shows that projects settle into markedly different configurations, from autocratic-mechanistic to democratic-organic, cautioning against the assumption that any single governance model characterizes OSS. This work also foregrounds an exit option with no counterpart in natural resource commons: a dissatisfied group may fork, departing with a copy of the code, and Gamalielsson and Lundell (2014) document that the community formed around such a fork can outlive the project it left—making forking a distinctive accountability mechanism that disciplines incumbent governance. The Governance Attributes of the STG Framework are accordingly designed to accommodate this diversity of arrangements rather than to presuppose the foundation-hosted model of the empirical setting examined below.

3. INTRODUCING THE SOCIO-TECHNICAL-GOVERNANCE FRAMEWORK

Figure 1 presents the Socio-Technical-Governance (STG) Framework, which synthesizes the STS approach from software engineering with the IAD Framework from the social sciences to provide a comprehensive lens for analyzing OSS sustainability. We do not take the introduction of yet another framework lightly, but taking a field-converging approach to the study of OSS sustainability makes it necessary. This convergence-driven meta-theoretical framework enables interdisciplinary research teams to systematically examine how social, technical, and governance factors interact to influence project outcomes, while providing a common vocabulary that allows researchers from different disciplinary backgrounds to communicate effectively and integrate their findings. The framework emerged from our recognition that understanding OSS sustainability requires conceptual tools that bridge the disciplinary divide between software engineering, which has deep expertise in technical aspects

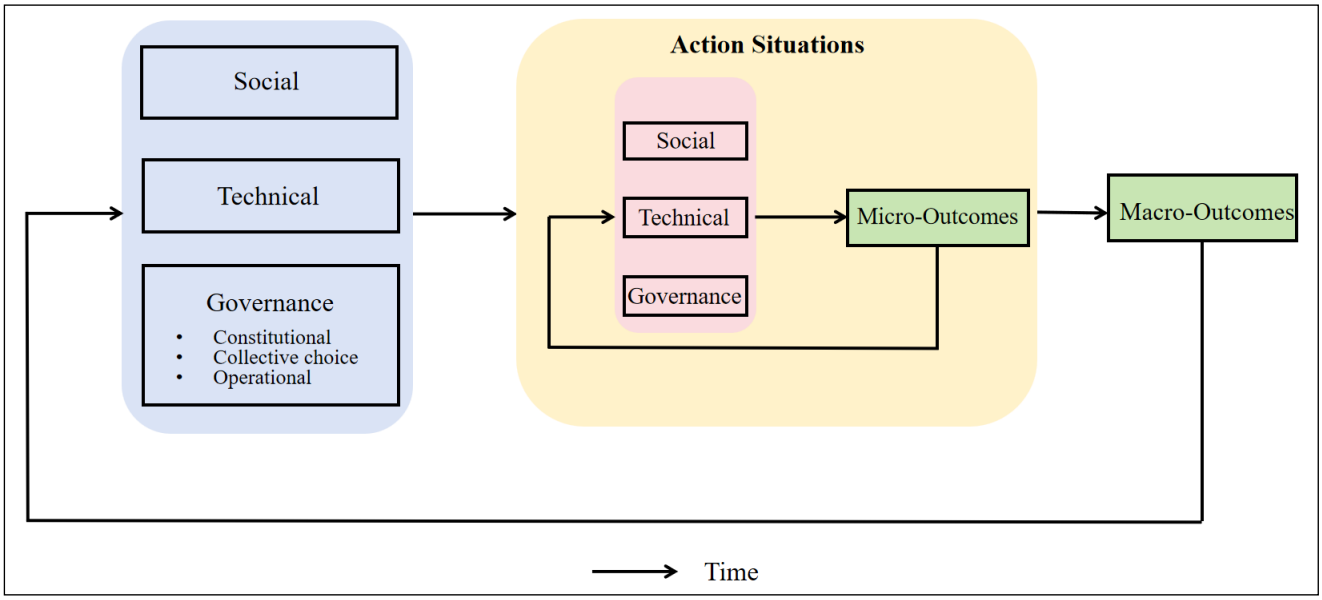


Figure 1 The Socio-Technical-Governance (STG) Framework for the Study of a Single OSS Project.

of software development, and institutional analysis, which provides sophisticated frameworks for understanding governance and collective action. Neither perspective alone is sufficient; the STG Framework provides the integration necessary for comprehensive analysis.

3.1 CORE DIMENSIONS OF THE FRAMEWORK

Social Attributes. The social dimension of the STG Framework encompasses variables related to the people engaged in an OSS project, corresponding to the "Attributes of Community" in the IAD Framework. This includes characteristics of individual participants such as whether they volunteer or are compensated for their contributions, the natural and programming languages they use, their prior experience and expertise, and their motivations for participating. These motivations have been the subject of a substantial literature: reviewing a decade of this research, von Krogh et al. (2012) classify contributors' motives as intrinsic (e.g., enjoyment), internalized-extrinsic (e.g., learning, reputation, reciprocity), and extrinsic (e.g., pay), while a related body of work turns from why individuals join to why they stay. Steinmacher et al. (2015) document the social barriers that drive newcomers out of projects around their first contribution, and Zhou and Mockus (2015) show that a newcomer's initial behavior, together with the attention received from the community, predicts long-term participation—evidence that sustaining a project depends as much on retaining contributors as on recruiting them. It also includes collective and relational properties such as team size and composition, the diversity of organizational affiliations represented, levels of trust and social capital among team members, and communication patterns and network structures. From an STS perspective, social attributes are often captured as dynamic networks recording interactions between participants—who communicates with whom, through what channels, how frequently, and in what capacity (Ducheneaut, 2005; Yin et al., 2021a). When modeled as graphs, these interaction patterns reveal structural properties such as degree centrality, clustering, core-periphery structures, and tie persistence that are empirically linked to project vitality and resilience. Two key groups typically emerge in OSS projects: contributors who submit code, documentation, bug reports, and other artifacts either regularly or occasionally, and decision-makers who are responsible for setting and enforcing project policies, resolving conflicts, and making binding choices about the project's direction. These groups overlap—many decision-makers are also active contributors—but the distinction between contribution and governance roles is analytically important.

Technical Attributes. Aligned with the "Biophysical/Material Conditions" category in the IAD Framework and the "Resource Characteristics" category in the GKC framework (Frischmann, Madison and Strandburg, 2014), Technical Attributes capture the interplay between software artifacts and the developers who create and maintain them. At its core, this dimension encompasses three key components. First, software artifacts include the source code files, libraries, and dependencies that constitute the project's tangible output. Characteristics such as code volume, complexity, modularity, documentation quality, and test coverage all influence project sustainability. Second, development infrastructure includes version control systems, issue trackers, continuous integration pipelines, and communication platforms that serve as the backbone for coordinating distributed development efforts. The choice and configuration of these tools shapes what kinds of collaboration are possible and how visible the project's activities are to potential contributors. Third, software metrics and practices include standards for code quality, review processes, release management procedures, and approaches to managing technical debt. Sustainable projects tend to maintain structured codebases, adopt scalable architectures, implement rigorous review processes, and invest in documentation and testing, thereby reducing single points of failure and ensuring effective knowledge transfer when contributors depart (Stănciulescu et al., 2022). These technical choices do not exist in isolation; collaboration norms, governance rules, and tool accessibility all influence which architectural decisions are feasible and how easily new contributors can participate effectively.

Governance Attributes. Governance in the STG Framework corresponds to "Rules-in-Use" or "Institutional Attributes" in the IAD and GKC Frameworks (Ostrom, 2009, 2011; Frischmann, Madison and Strandburg, 2014). Institutions—the prescriptions that humans use to organize all forms of repetitive and structured interactions—significantly influence day-to-day project management and long-term viability, and are found at three nested levels. Constitutional-level rules capture high-level aspects of governance including the choice of copyright licensing (such as Apache 2.0 or GNU GPL), foundational positions and roles within the project, legal structures, and rules about who has authority to change lower-level rules. Collective-choice level rules specify decision-making processes, defining who has the authority to change operational-level rules and through what procedures such changes occur. Operational-level rules guide daily practices, including version control permissions, code review requirements, communication protocols, and informal norms about expected behavior. Some OSS projects operate independently with self-governing arrangements, while others affiliate with nonprofit foundations such as

the Apache Software Foundation or Linux Foundation that provide funding, legal assistance, trademark protection, and overarching policies. In such cases, governance is nested, with project-level rules operating within the constraints of foundation-level requirements.

3.2 ACTION SITUATIONS

At the heart of the STG Framework lies the concept of an Action Situation, adapted from the IAD Framework. Following Ostrom (2009: 32), an Action Situation forms when "two or more individuals are faced with a set of potential actions that jointly produce outcomes." The Action Situation is the core component of the IAD Framework, the arena in which individuals acting on their own or as agents of organizations observe information, select actions, engage in patterns of interaction, and realize outcomes from their interaction (McGinnis, 2020). Within the STG Framework, Action Situations in OSS projects are where individuals engage in activities including software development, code review, bug triage, project governance decisions, conflict resolution, and community engagement. These recurring interactions form the basis through which project sustainability is fostered or undermined.

3.3 MICRO-OUTCOMES AND MACRO-OUTCOMES

It is here that we diverge from past articulations of IAD. In our STG Framework, the outcomes of interactions within Action Situations can be categorized at two levels, a distinction that is particularly important for analyzing OSS projects where digital trace data capture fine-grained activities over time. Micro-outcomes are the immediate, small-scale effects of individual actions: a single bug fix, a code commit, a merged pull request, a resolved discussion, a new contributor's first contribution, or a minor adjustment to team processes. These micro-level activities leave digital traces in version control systems, issue trackers, mailing list archives, and other collaboration infrastructure. Over time, micro-outcomes may aggregate into macro-outcomes that define the long-term trajectory of the project. Macro-outcomes include major software releases, significant governance transitions such as moving from individual maintainership to collective governance or affiliating with a foundation, successful recruitment of a sustainable contributor base, or alternatively, declining participation leading to project abandonment. A primary reason we add this micro- and macro-outcome distinction is because in contexts like OSS the scientist can often find "digital traces" representing very fine-scaled or minor activities by OSS developers. In the older days of IAD-based empirical work, scientists in the field would rarely have time to focus on such micro-level actions. In digital commons settings, these actions are often open-access and traceable and allow the analyst to study the internals of an Action Situation.

Scaling out more broadly in Figure 1, understanding Action Situations enriches the analysis of OSS sustainability by revealing the mechanisms through which sustainability emerges or deteriorates. Within an action situation, patterns of interaction may support or undermine project objectives. Supportive patterns are characterized by active engagement, ongoing technical improvement, and transparent governance processes, and these positive micro-level interactions may aggregate into positive macro-outcomes such as community growth, sustained innovation, and long-term project viability. Conversely, patterns of interaction may be marked by declining participation, unresolved conflicts, technical stagnation, or governance dysfunction. These negative patterns can accumulate into macro-outcomes including community fragmentation, mounting technical debt, and eventual abandonment. By applying the STG Framework and analyzing both positive and negative patterns within action situations, researchers and practitioners can identify candidate early-warning signs of sustainability challenges and formulate intervention hypotheses for prospective testing.

3.4 FEEDBACK LOOPS AND NESTED GOVERNANCE

The STG Framework conceives OSS ecosystems as nested feedback loops operating at multiple levels. At the micro-level, each project operates an inner loop where contributors, technical artifacts, and project-specific governance rules continually influence one another. Decisions about coding standards, review workflows, or release criteria emerge from this loop, and their effectiveness is immediately felt in day-to-day collaboration. For projects affiliated with a foundation, there is an outer loop defined by the broader organizational environment that provides resources, normative guidance, and legitimacy. This outer environment receives performance signals from individual projects—contributor churn, release cadence, security incidents, community health indicators—and may respond with incentives or constraints such as mentoring programs, funding decisions, or compliance requirements. Importantly, the rules operating at these different levels are not identical: foundation-level policies may mandate release management guidelines, voting thresholds, or licensing constraints, while individual projects maintain their own processes for merging contributions or resolving disputes.

Because these loops are coupled, changes at one level propagate to others. A project that adopts containerized development environments to simplify onboarding

(a technical change) may spark increased newcomer participation (a social effect), prompting clarification of contributor agreements and committer promotion criteria (governance adjustments). Success at the project level may be recognized by the foundation through additional resources or prominent positioning, while persistent issues such as unresolved security vulnerabilities or community conflicts can trigger corrective interventions from the outer loop. Framing OSS sustainability in terms of these intertwined loops has several practical benefits: it draws attention to potential misalignments between project and foundation expectations before they undermine viability, it guides the design of interventions targeting the appropriate level, and it establishes an empirically testable research agenda for tracing how signals such as contributor activity, architectural changes, and governance discussions propagate across levels.

4. THE APACHE SOFTWARE FOUNDATION INCUBATOR

The Apache Software Foundation (ASF) is one of the oldest and most established nonprofit organizations supporting OSS development, having operated since 1999. The ASF hosts over 350 top-level projects and provides governance, legal, and infrastructure support to a broad ecosystem of software communities. For more than twenty years, nonprofit foundations like ASF have emerged to provide legal protection, technical infrastructure, and financial support to OSS projects (O'Mahony, 2005; Riehle and Berschneider, 2012). Izquierdo and Cabot (2020) reported a survey of 101 software foundations, suggesting that foundation-hosted OSS is an increasingly common organizational form. The ASF represents a particularly well-established example with mature processes and extensive documentation of its governance practices.

Figure 2 presents our STG Framework representing a variety of OSS projects with their own individual social, technical and governance attributes but associated with and working with the ASF nonprofit foundation. Figure 2 also showcases four key Action Situations within the ASF context: (1) new potential project – or what ASF calls "Podling" – incubation; (2) decisions around ASF governance; (3) the management of OSS projects that successfully incubate, what ASF calls "Top Level Projects"; and (4) maintenance of OSS project copyright licensing issues.

In the sections that follow, we synthesize evidence from five previously published source studies of the Podling Incubation ASF Action Situation: one shared dataset paper and four empirical analyses. In essence, ASF operates an Incubator program through which new projects can affiliate with the foundation and receive support for their development. Projects enter incubation as "podlings" after submitting a proposal reviewed by the Incubator Project Management Committee (IPMC). During incubation, podlings work to meet specific requirements while receiving mentorship from experienced ASF members who provide guidance on technical, social, and governance

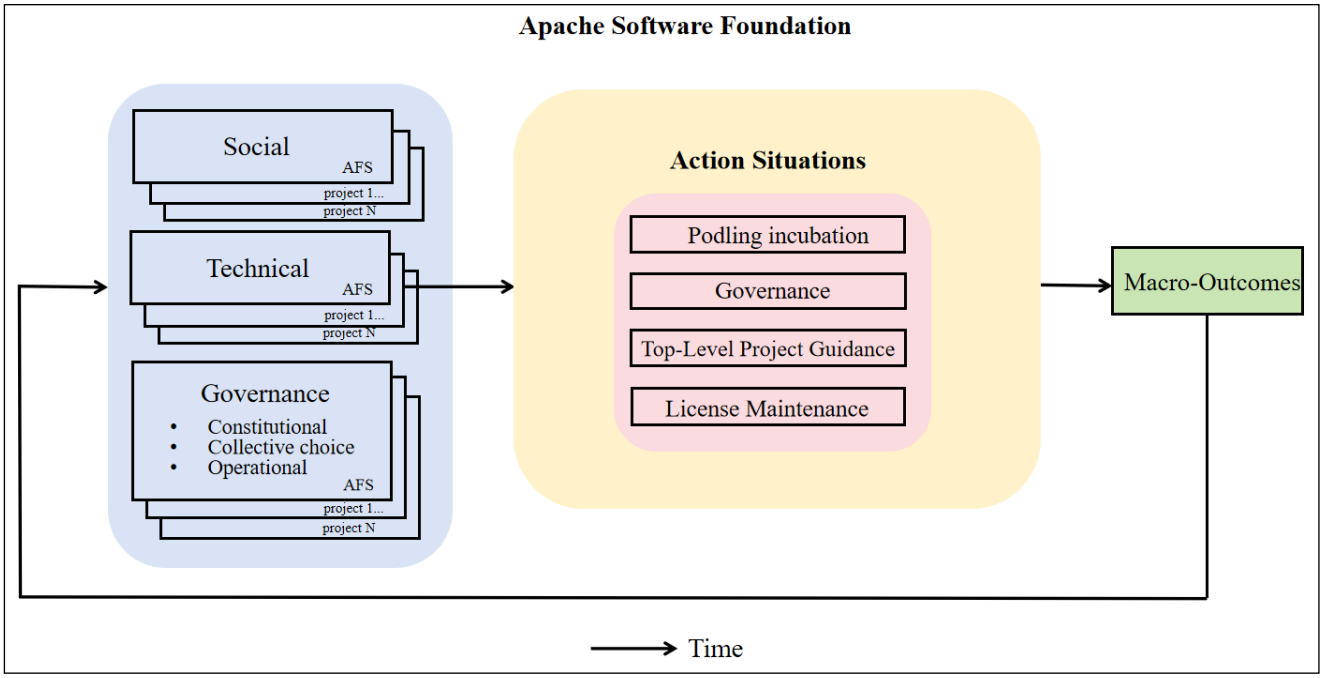


Figure 2 The Four Apache Software Foundation Action Situations.

matters. Projects that successfully meet requirements graduate to become Top-Level Projects (TLPs) with full ASF membership and governance autonomy. Projects that fail to meet requirements or become inactive are retired from the incubator with no continuing ASF affiliation. This binary outcome—graduation versus retirement—provides a clear macro-level measure of project success within the incubation context. We treat this outcome throughout the paper as an observable proxy for sustainability within the incubation context rather than as a direct measure of long-term project survival or abandonment, since a retired podling may continue development outside the ASF, a caveat that the source studies themselves note (e.g., Yin et al., 2022).

The ASF Incubator provides an ideal empirical context for studying OSS sustainability for several reasons. First, it offers clear macro-outcome measures—e.g., in the incubation action situation, the outcomes are graduation versus retirement—enabling comparative analysis of project trajectories leading to different outcomes. Second, the ASF maintains comprehensive open-access records of project activities including code commits, email discussions, governance documents, and status reports, providing rich digital trace data for analysis. Third, it exhibits the nested governance structure characteristic of foundation-hosted OSS, where individual projects maintain self-governing arrangements while also being subject to ASF-level requirements known as "The Apache Way." Fourth, the incubation process creates a structured Action Situation with defined expectations, milestone requirements, and regular review, enabling focused analysis of how projects navigate institutional requirements.

Upon acceptance into incubation, projects work to meet three primary expectations that reflect the ASF's emphasis on sustainable community development. First, they must ensure that their code meets ASF copyright license requirements, typically involving license header verification and contributor license agreements that protect both the foundation and downstream users. Second, they must demonstrate competence in software release management by successfully completing multiple release cycles during incubation, showing that the team understands ASF release processes and quality standards including cryptographic signing and verification procedures. Third, they must provide evidence of ability to recruit new developers and build a sustainable community, with particular emphasis on "diversity" meaning contributors from at least three distinct organizations to reduce dependence on any single company and enhance project resilience. The IPMC evaluates progress against these criteria through regular status reports and votes on whether projects have met graduation requirements.

5. SYNTHESIS OF PUBLISHED EMPIRICAL FINDINGS

With the STG Framework fully presented, we now present five interrelated, previously published studies, conducted by various members of a broader multidisciplinary research team, that analyze different aspects of the Podling Incubation Action Situation within the Apache Software Foundation. The source studies were independently designed and previously published; here, we map them retrospectively onto the STG Framework to examine whether it can organize complementary evidence across dimensions. Together, these studies demonstrate how social, technical, and governance attributes can be operationalized and measured, and how they relate to project outcomes. They also demonstrate how, thanks to the availability of open access longitudinal digital information in these OSS settings, researchers can study the micro-outcomes that occur within Action Situations and ultimately their macro-outcomes (Figure 1). Importantly, although this section summarizes individual studies published elsewhere, including one in a previous issue of IJC, the original contribution here lies in presenting them together: (1) we demonstrate how each study informs or analyzes particular elements of an STG-based study, and (2) we provide an example of multi-disciplinary, convergence-related research.

Specifically, the studies build on a shared dataset infrastructure while focusing on different framework components. The four empirical analyses examine technical code metrics, social dynamics including mentorship, governance policies and their distributional implications, and the integration of all three dimensions through computational methods. After presenting the common dataset structure, we summarize these four analyses to show how independently conducted studies can be connected within a larger STG-based computational social science synthesis. Figure 3 maps these studies onto the STG Framework, coding each element with the section in which the study examining it is summarized.

5.1 BUILDING THE INITIAL ASFI DATASET

As just mentioned, an important foundation for the four empirical studies described in Sections 5.2–5.5 was the development of a comprehensive longitudinal dataset of developer activities across 269 projects that have passed through the ASF Incubator (Yin et al., 2021b). This 269-project dataset is best understood as the shared data infrastructure from which the four empirical analyses below drew. This substantial open-access data was scraped from ASF web infrastructure and organized to support a wide variety of studies examining OSS sustainability from

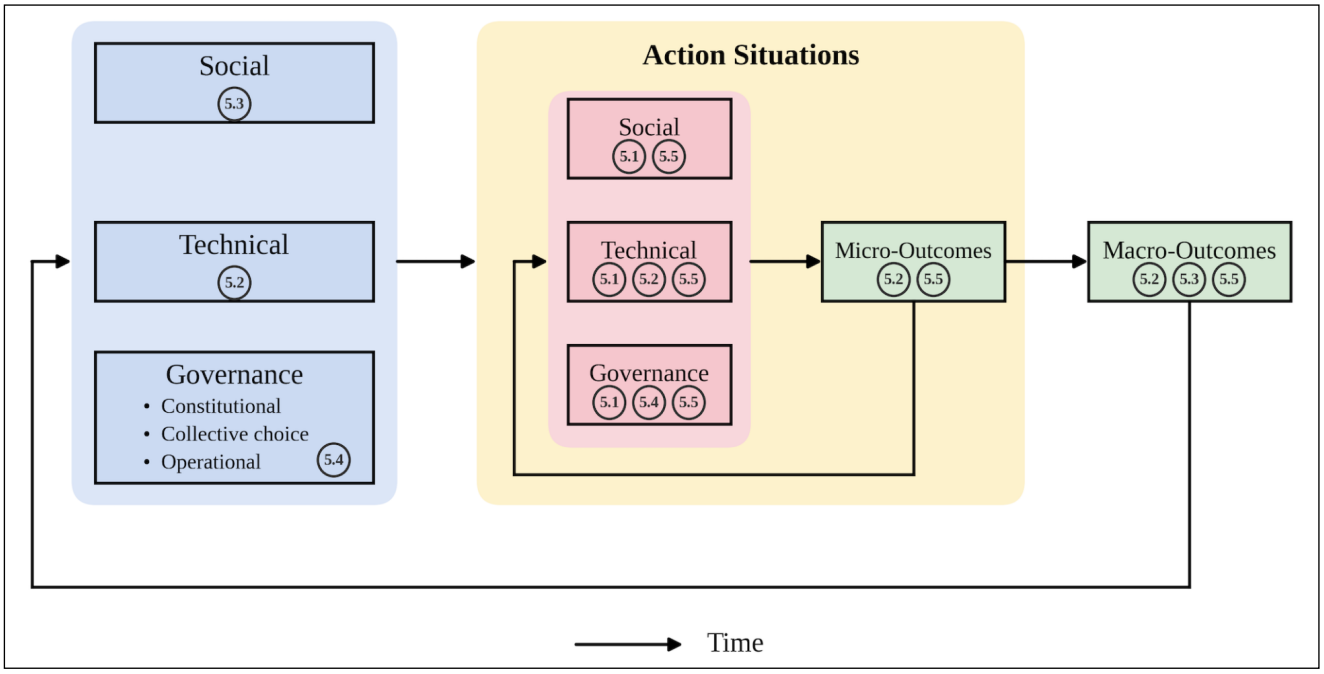


Figure 3 Mapping the five studies onto the STG Framework. Codes indicate the section in which the study examining each element is summarized.

multiple perspectives. The dataset contains historical trace data of Social (software developer) activities, including emails sent to project mailing lists where ASF communities conventionally document and deliberate decisions, code commits to version control repositories, linked sponsor and mentor role records, project status reports, and incubation outcomes (graduated or retired). Embedded in these records are digital traces of both micro-level activities (individual commits, email messages, review comments) and macro-level events (releases, graduation votes, retirement decisions). Technical aspects are captured by references to certain project file repositories shown under the commits and filelist components of the database schema (Figure 4).

This schema also captures the complex Social and Technical relationships between projects, developers, communications, and code contributions that emerge during the incubation process. Governance aspects of the STG Framework are somewhat concealed in this schema, but are embedded in the communication (e.g., email discussions) that are captured in this schema as will be described more fully in Section 5.5. Sen, Atkisson and Schweik (2022) augment this with the development of additional Governance information related to the ASFI, as described in Section 5.4.

Taken as a whole, the dataset enables STG-based longitudinal analysis tracking how projects evolve from entry into incubation through graduation or retirement, supporting both retrospective comparisons of successful and unsuccessful projects and prospective modeling of project trajectories. Further, with its longitudinally fine-temporal scale and extensive data, the database gives the scientist substantial data to open up the Action Situation and study micro-level outcomes (center, Figure 1). With this database in place, researchers can investigate a variety of STG-based research questions related to projects operating in the ASF Podling Incubator Action Situation both at the micro- and macro-Action Situation outcome scales. The four summaries of published studies that follow illustrate this.

The five studies were designed and published independently, and their analytic samples differ accordingly (Table 1). Units of analysis vary: two model project-months, one models mentor-project pairs, and Sen, Atkisson and Schweik (2022) code policy text and contribute no project-level sample. The project-level samples overlap without nesting, reflecting three design choices: the closing date of each archive snapshot (the shared dataset's collection ends in August 2019, while the four analyses drew on later snapshots extending into 2021), the data-availability filter applied (a retrievable correspondence archive for the mailing-list studies, a GitHub repository for the code-metrics study, which excludes older SVN-era podlings), and the treatment of projects still in incubation. The synthesis below thus draws convergent evidence from overlapping windows on a single population rather than repeated measurement of one fixed sample.

SOURCE STUDY	UNIT OF ANALYSIS	SAMPLE	TIME WINDOW	INCLUSION AND EXCLUSION FILTERS
Yin et al. (2021b), Section 5.1	Project (shared data infrastructure)	269 podlings (179 graduated, 46 retired, 44 still in incubation at collection)	Mailing-list and commit archives collected from March 2003 to August 2019	Podlings with retrievable records on ASF web infrastructure; reports no findings of its own
Stănciulescu et al. (2022), Section 5.2	Project-month	218 projects (184 graduated, 34 retired); 5,365 observations, 5,245 after removing the top 3% of outliers	podlings.xml downloaded June 2021	236 of the 328 listed podlings had a Git repository on GitHub; SVN-only projects and projects whose repository could not be located were excluded, then 18 projects with no activity or incomplete data were dropped
Atkisson (2022), Section 5.3	Mentor-project pair	303 mentors and 286 projects in the regression step; descriptive statistics reported for 264 projects (204 graduated, 60 retired)	Full history of the ASF Incubator, scraped in 2021	Mentor lists taken as the union of the program overview page and each individual project page; no archive-availability filter applied
Sen, Atkisson and Schweik (2022), Section 5.4	Institutional statement in policy text	8 policy documents; 259 operational-level statements (1,405 coded statements in total)	ASF policy documents as posted in 2021	Formal policy documents plus the documents to which new podlings are directed on entry; no project-level sample
Yin et al. (2022), Section 5.5	Project-month	253 projects (204 graduated, 49 retired); 1,330,003 emails collected, 1,201,746 retained after removing bot-generated messages	29 March 2003 to 1 February 2021	Graduated and retired projects only; longitudinal comparisons restricted to the first 24 incubation months

Table 1 Reconciliation of the samples analyzed in the five source studies.

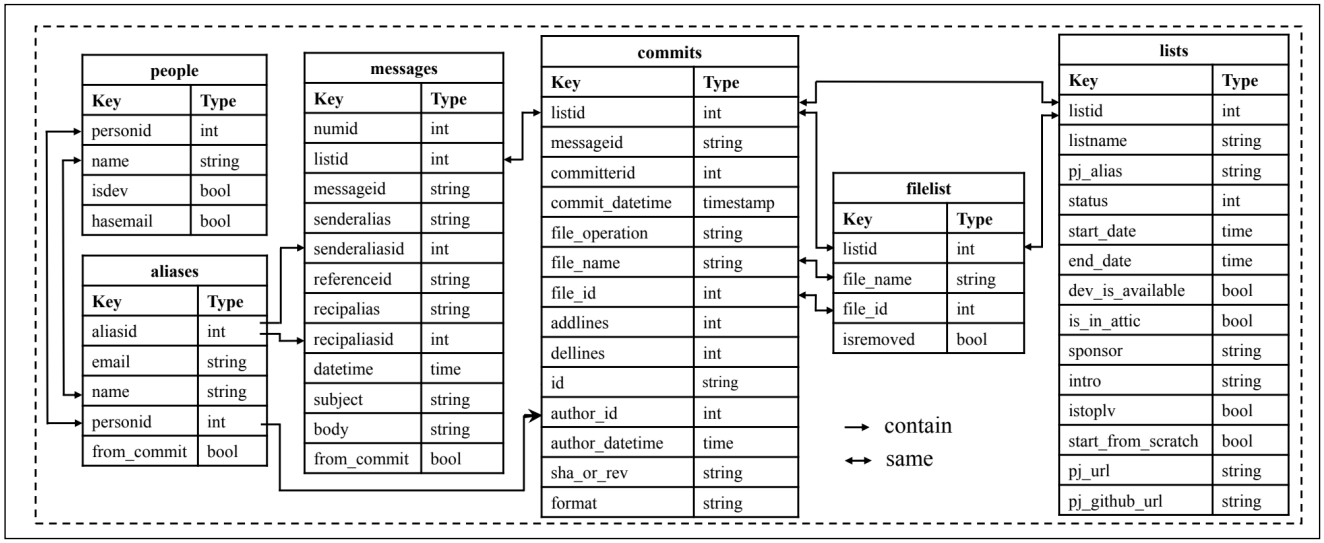


Figure 4 The Data Schema of the ASF Incubator Dataset (*Source*: Yin et al., 2021b).

5.2 EXAMPLE 1: EXAMINING TECHNICAL ATTRIBUTES AND ACTION SITUATION MICRO AND MACRO PROJECT OUTCOMES

Stănciulescu and colleagues (2022) investigated the relationship between technical metrics and podling incubation outcomes in their paper "Code, Quality and Process Metrics in Graduated and Retired Projects," focusing on the Technical Attributes component of the STG Framework. The study addressed three research questions: (RQ1) How do graduated and retired projects differ across code, process, and quality metrics? (RQ2) Are project trajectories different along these metrics? (RQ3) What are the code, process, and quality metrics determinants of whether a project graduates or retires? The analysis covers 218 podlings (184 graduated, 34 retired) and 5,245 project-month observations, restricted to the 236 of 328 listed podlings for which a Git repository was available on GitHub (Table 1).

The study developed 23 independent variables representing three categories. Code metrics included lines

of source code without comments (SLOC), number of code directories, and number of top-level directories. Process metrics included measures of major contributors, minor contributors, new contributors, files added, files deleted, average files modified per commit, and the number of days per month with at least one commit (commit-active days). Quality metrics included SLOC of test code as a proxy for project maturity, numbers of functions with medium or high risk McCabe complexity index, and the number of very large files (over 1000 SLOC) known to contain more problems.

Results for RQ1 showed that graduated and retired projects differentiate themselves across multiple dimensions. Compared to retired projects, graduated projects have fewer lines of code and fewer directories per developer (code metrics), more activity in terms of code commits and file deletions (process metrics), and more complex code as well as more lines of test code (quality metrics). For RQ2, longitudinal analysis of the Podling Incubation Action Situation revealed that graduated and retired projects follow different trajectories after entering the incubator (Figure 5). Graduated projects demonstrated significantly more active commit days throughout their incubation period compared to retired counterparts. Some projects entered well-positioned and graduated quickly; others required longer periods to build momentum. Retired projects demonstrated higher burden per developer due to fewer developers overall and fewer newly recruited contributors—precisely the growth that incubation requirements emphasize.

For RQ3, Stănciulescu and team ran Generalized Linear Regression (glm) and Generalized Mixed Effect Regression (glmer) models on Java projects. The Java-only model rests on 3,795 project-month observations; a companion mixed-effects model covering all languages (4,970 observations) yields coefficients in the same directions but attributes most of the explained variance to the dominant programming language. The mixed-effects diagnostics (marginal R^2 = 0.154 against conditional R^2 = 0.844) indicate that language choice absorbs most of the explained variance, so the coefficient directions summarized here follow the Java subsample and should not be generalized to non-Java projects. Results showed that increases in lines of code, major and minor contributors, feature commits, corrective commits, medium complexity functions, and very large functions were all associated with higher graduation odds. Conversely, increased top-level directories, average files touched per commit, very large files, and code duplication percentages were associated with higher retirement odds. These estimates are associational rather than causal: projects approaching graduation may already be changing their behavior in anticipation of the outcome,

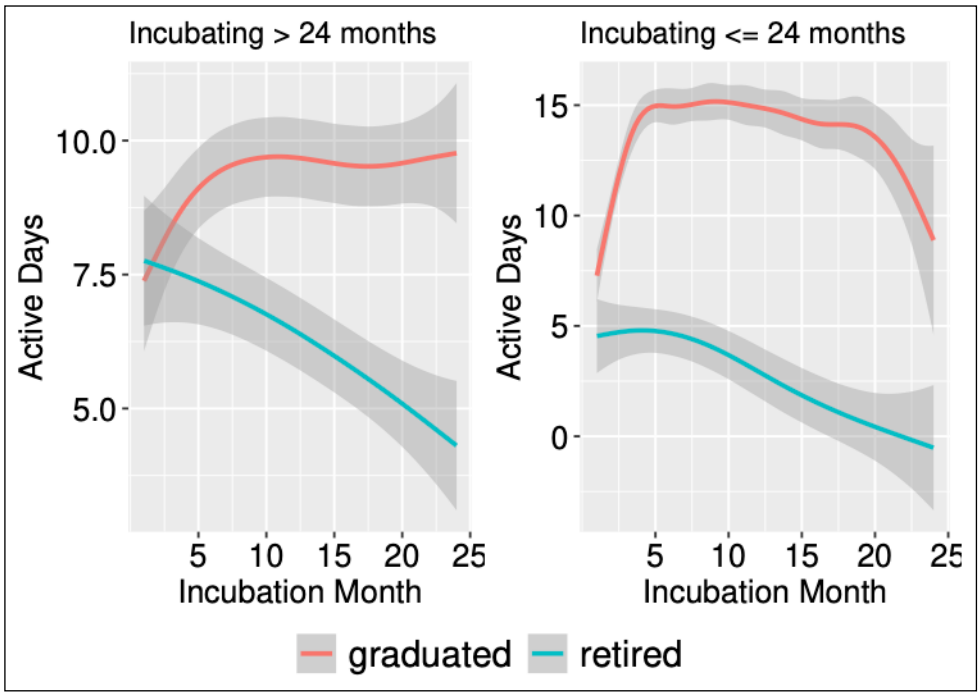


Figure 5 An example of the internal dynamics of the ASF Incubation Action Situation. The number of commit-active days per incubation month for graduated (red) versus retired (blue) podlings, drawn from the 218-project sample (184 graduated, 34 retired); curves are smoothed means with shaded standard-error bands. The x-axis is the incubation month; the left panel shows projects that incubated more than 24 months, the right panel those that incubated 24 months or less, with the first 24 months displayed (*Source*: Stănciulescu et al., 2022).

so the determinants are best read as predictive signals of the macro-outcome rather than as levers that produce it. Further caveats apply: the retired group is small (34 projects) relative to the graduated group, the top 3% of observations were removed as outliers, estimates across the 23 metrics were not adjusted for multiple testing, and greater code complexity may reflect project size or functionality rather than superior quality.

5.3 EXAMPLE 2: EXAMINING THE SOCIAL ATTRIBUTES OF THE ASF INCUBATOR ACTION SITUATION AND THE MACRO-OUTCOME OF GRADUATION OR RETIREMENT – THE ROLE OF MENTORSHIP

Atkisson (2022) examined whether mentor assignment matters for project outcomes in "Mentors Matter: Association of Mentors with Project Success in the Apache Software Foundation Incubator," addressing the Social attributes component of the STG Framework. The ASF Incubator assigns experienced ASF members as mentors to guide podling projects through the incubation process. The study modeled the mentoring relationships between 303 mentors and 286 incubated projects over the incubator's history, and reported descriptive statistics for 264 of these projects (204 graduated, 60 retired, an empirical graduation rate of 77%). Of the 303 mentors, 128 had mentored exactly one project while 175 had mentored two or more, with some mentors overseeing many projects over extended periods. The research question asked: Does who a project has as a mentor matter for the project's graduation or retirement from the ASF Incubator?

Using binomial regression with LASSO variable selection (due to more predictors than observations), the analysis found that mentor identity is meaningfully associated with graduation likelihood. According to one ASF Incubator Project Management Committee member, between three and five mentors is considered ideal. Figure 6 shows the relationship between number of mentors and graduation rate, with dot size representing the number of projects at each intersection point. Having more mentors is associated with a higher probability of graduating and becoming a Top-Level Project. Atkisson (2022) reports a Tjur's coefficient of discrimination (a pseudo-R^2 for binary outcomes) of 0.45 for the post-LASSO binomial regression, meaning that the mean fitted graduation probability for graduated projects exceeded that for retired projects by 0.45 in the analyzed sample; this in-sample statistic should not be interpreted as 45% of outcome variance explained. LASSO shrinkage biases coefficient magnitudes, so only the direction and relative size of mentor effects are interpretable; effects distinguishable from zero were found for only 33 of the 303 mentors; and because the analysis is observational, unmeasured project quality or resources may confound the association, ASF mentors may select into projects they expect to succeed, and reverse causation is not ruled out—for example, projects that appear more

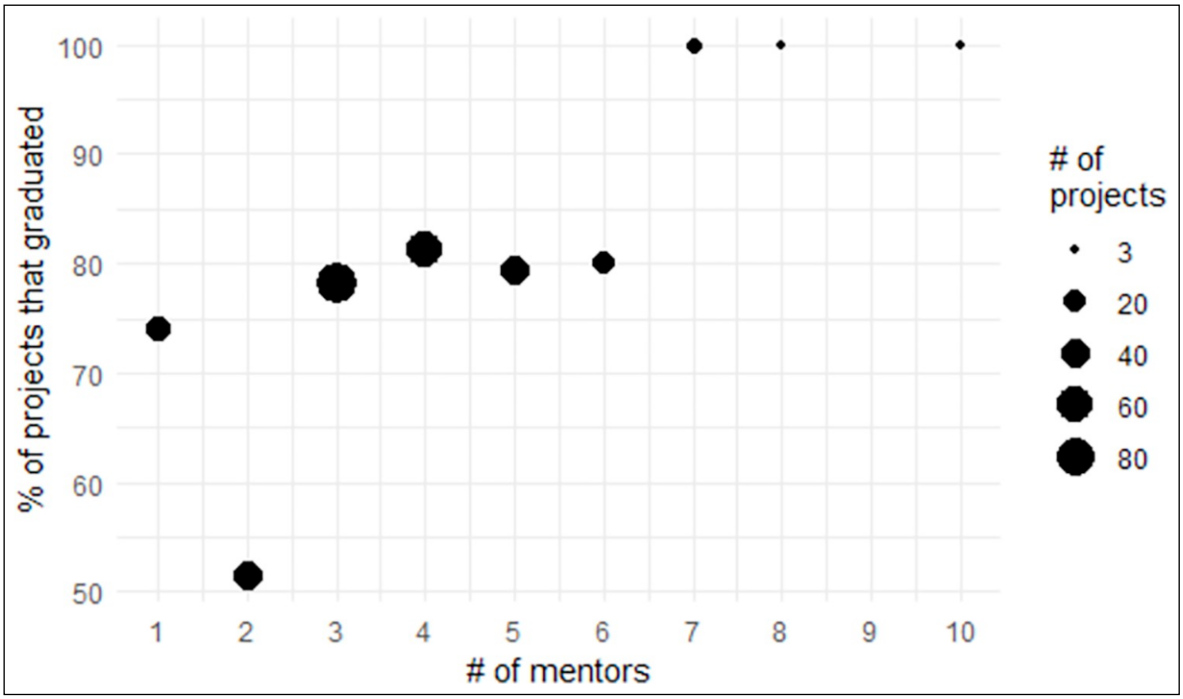


Figure 6 Examining the ASFI Macro-outcome of Graduation. The y-axis gives the percentage of projects that graduated and the x-axis the number of mentors assigned, over the 264 graduated or retired projects; dot size encodes the number of projects with that many mentors (*Source*: Atkisson, 2022).

likely to graduate may attract additional mentors. The estimates should therefore not be read as causal effects of mentoring. As Atkisson (2022: 11) explains, "The ability of creators and maintainers [of OSS software] to manage the complex needs of the projects will impact their long-term sustainability. Advice from experienced managers of OSS projects on these aspects of the projects, which is what mentors do in the context of ASF, can help ease the transition of developers to become managers of complex projects." This highlights mentoring as strongly associated with the graduation or retirement macro-outcome in the Podling Incubation Action Situation.

5.4 EXAMINING GOVERNANCE: DISTRIBUTION OF COSTS AND BENEFITS

Sen, Atkisson and Schweik (2022) investigated the governance dimension of the STG Framework and in the Podling Incubation Action Situation in "Cui Bono: Do Open Source Software Incubator Policies and Procedures Benefit the Projects or the Incubator." The study analyzed operational-level policies guiding incubating projects, specifically examining how costs and benefits are distributed between two actor types: project developers and ASF officials. The team systematically reviewed eight ASF policy documents that define the rules of the incubation action situation: Incubation Policy (overarching reference document), Apache Incubator Cookbook (information for prospective projects to assess fit), Podling Project Management Committee (PPMC) guide, and several others. Through careful examination and coding statements following Ostrom's Institutional Grammar (Crawford & Ostrom, 1995), they identified 259 operational-level institutional statements—prescriptions that specify required, prohibited, or permitted actions. Since the unit of analysis here is policy text rather than projects, this study contributes governance measures rather than a project-level sample to the synthesis (Table 1).

Each institutional statement was coded along three dimensions: the focus area (such as risk mitigation, release management, community development, or graduation criteria), which actor type primarily bears the costs of complying with the statement (project developers or ASF officials), and which actor type primarily receives the benefits. Two independent coders classified all 259 statements, yielding a Krippendorff's alpha of 0.66 (95% CI = [0.63, 0.69]). This value sits just below the 0.667 benchmark that Krippendorff (2019) proposes for drawing tentative conclusions and is better described as moderate than as strong agreement; the published analysis accordingly rests on a discussion-resolved final coding (one residual disagreement among 1,405 coded statements) and reports a robustness measure for each table, with R = 0.054 for the aggregate distribution indicating that roughly 14 statements would need to be recoded to overturn the symmetry finding reported below. The coding also weighs statement counts rather than the magnitude of the costs and benefits each statement carries, so the balance at issue concerns who bears formal obligations, not their size. The aggregate analysis did not detect asymmetry in how costs and benefits fall between ASF actors and OSS project developers (Figure 7). Chi-squared testing showed a symmetric distribution ($\chi^2 = 0.075$, $p = 0.78$), indicating little evidence of an association between the actor classified as bearing a formal obligation and the actor classified as receiving its benefit. Because the analysis counts policy statements rather than the magnitude or distributional importance of their consequences, it does not establish equivalent burdens, substantive fairness, or historical evolution toward equity; given the moderate independent coder agreement noted above, the symmetry finding should also be regarded as tentative. The pattern is consistent with—but does not test—Ostrom's (1990) proposition that proportionality between costs and benefits supports successful commons governance.

5.5 INTEGRATING SOCIAL, TECHNICAL, AND GOVERNANCE DIMENSIONS

Yin and colleagues (2022) conducted the most integrative study of the Podling Incubation Action Situation in "Open Source Software Sustainability: Combining Institutional Analysis and Socio-Technical Networks," by combining all three STG dimensions and developing machine learning methods to identify governance discussions in project communications and linking these to socio-technical network dynamics. The ASF's open communications policy states that "as a virtual organization, the ASF requires all communications related to code and decision-making to be publicly accessible to ensure asynchronous collaboration."

		BENEFIT		TOTAL
		PROJECT	ASF	
Cost	project	101	60	161
	ASF	59	39	98
Total		160	99	259

Figure 7 Examining Governance in the ASF Incubation Action Situation. Cells cross-tabulate the 259 operational-level policy statements by which party bears the cost (rows) and which receives the benefit (columns). Chi-squared test statistic = 0.075, p-value = 0.78, R = 0.054 (*Source*: Sen et al., 2022).

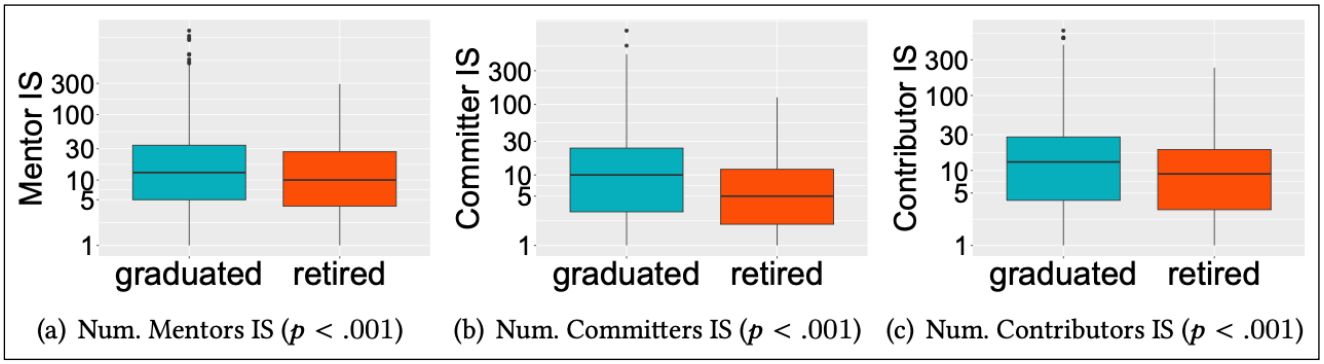


Figure 8 Examining the ASFI macro outcome of Graduation and Retirement – Comparing graduated and retired projects on institutional statement discussions by mentors, committers, and contributors. Boxplots give the monthly count of institutional statements per project (log-scaled y-axis) by sender role, for graduated (blue) versus retired (red) projects in the 253-project sample (204 graduated, 49 retired); Mann-Whitney U test p-values indicate significant differences (*Source*: Yin et al., 2022).

Project mailing lists are archived and publicly accessible, and private decisions on code, policies, or project direction are disallowed. This enables study of both social dynamics and governance discussions.

Three research questions were posed: (RQ1) whether institutional statements are contained in ASF Incubator project email discussions and, if so, whether they can be effectively identified; (RQ2) whether OSS project evolution toward sustainability, operationalized in that study as graduation rather than retirement from the incubator, is readily observable through the dual lenses of institutional and socio-technical analysis, and how such temporal patterns differ; and (RQ3) whether periods of increased institutional-statement frequency are followed by changes in project organizational structure, and vice versa. For RQ1, the team leveraged 1,330,003 emails from 253 projects (204 graduated, 49 retired) covering 03/29/2003 to 02/01/2021. After removing automatically generated emails, Yin and colleagues trained a BERT-based classifier (Cohan et al., 2019) to detect discussions about rules or "institutional statements" (IS) in email texts. Training combined 273 coded email threads with 328 policy statements from Sen et al.'s work, achieving an F1 score of 0.674.[2] Applied to 1,201,746 developer emails, it identified 313,140 sentence-level institutional statements, an average of 0.261 per email. An F1 of 0.674 implies a non-trivial share of false positives and false negatives among these detected statements, so the IS variables enter the analyses below as noisy proxies for governance discussion rather than as ground truth.

For each project-month, variables counted IS occurrences by actor type: mentors (num_IS_mentor), committers (num_IS_committer), and contributors (num_IS_contributor). Mentors showed the highest mean IS count, suggesting they advise developers about ASF rules and norms. Analysis showed mentors of graduated projects sent significantly more institutional statements than mentors of retired projects (Figure 8). Similar patterns appeared for committers and contributors. Latent Dirichlet Allocation topic modeling identified twelve distinguishable topics, with IS discussions centered on progress reports, project releases, documentation requirements, and copyright license alignment—directly aligned with ASF requirements.

To address RQ2, socio-technical networks were also constructed for each project-month. Social networks used weighted directed graphs with nodes representing developers and edge weights representing communication frequency. Technical networks had nodes for both developers and files, with edge weights for committing frequency. For social networks, s_num_nodes counted unique developers communicating each month (mean ~13 people across all projects). For technical networks, t_num_dev_nodes counted active committers (mean 1–2 per month) and t_num_file_nodes counted files touched (mean 61 per month).

Figure 9 shows longitudinal trajectories across incubation – another example of longitudinal analysis of micro-outcomes within an Action Situation. Mentors send more institutional statements to projects that eventually graduate from the earliest months onward; counts decline for both groups through roughly month five and then rise again, which the source study reads as mentors stepping in when projects encounter difficulties. Committers in graduating projects maintain higher IS discussions throughout. Social network analysis showed graduated projects exhibiting a clear and increasing trend in communicating participants over the first six months, while retired projects remained constant. Throughout incubation, graduated projects maintained higher participant engagement. Technical network analysis showed graduated projects with more active committers and more files touched per month than retired projects across the observed window. This integrated analysis suggests that distinct trajectories across all three

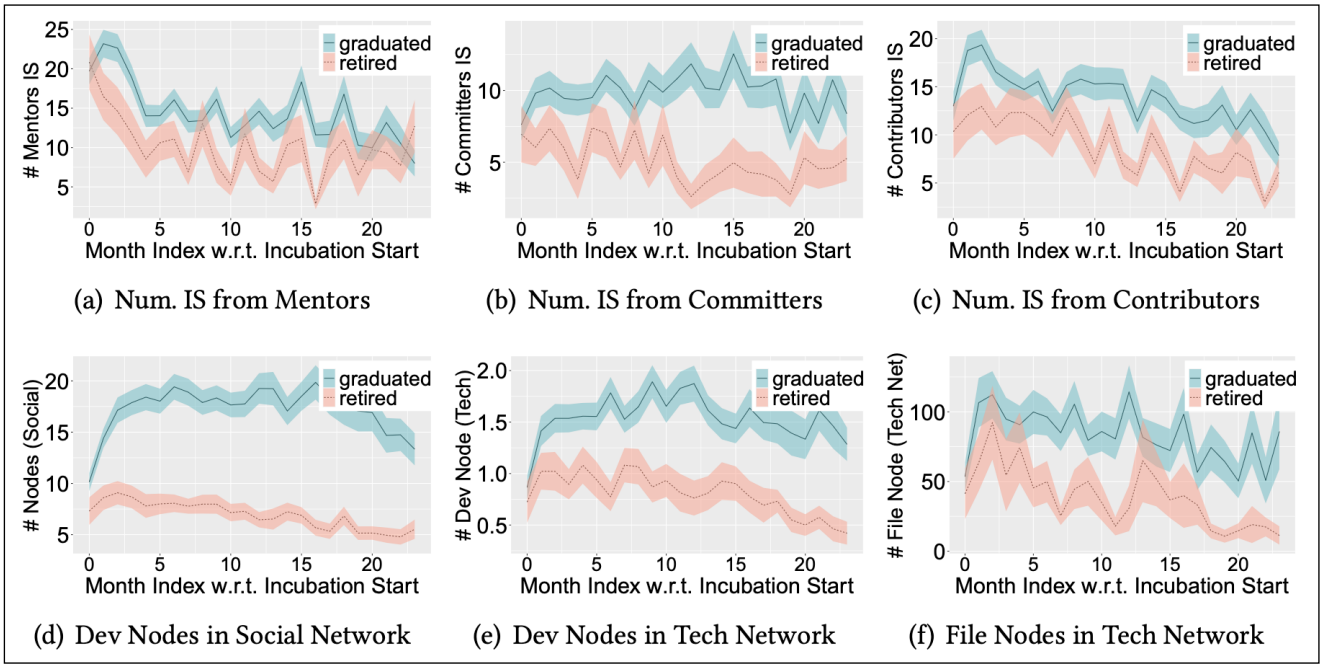


Figure 9 Examining the longitudinal dynamics of micro-outcomes within ASF's Incubation Action Situation. Panels show monthly means of institutional-statement counts by sender role (top row) and socio-technical network measures (bottom row) for graduated (blue) versus retired (red) projects over the first 24 incubation months; month index 0 is the incubation start, and shaded areas indicate one standard error from the mean (*Source*: Yin et al., 2022).

STG dimensions become observable within the first year of incubation, potentially enabling early identification of projects requiring intervention.

RQ3 turned to whether these dimensions move together over time. Yin and colleagues ran pairwise Granger causality tests with a lag of two months across all institutional and socio-technical variable pairs, separately for graduated and retired projects, adjusting for multiple testing with the Benjamini-Hochberg procedure and treating $p < 0.001$ as significant. They report 31 Granger-causal relationships among the 72 pairs tested, 15 within the graduated set and 16 within the retired set, of which 8 are bidirectional. Institutional statements from mentors, committers, and contributors precede changes in the technical network measures, whereas paths into the social network measures are largely absent, and graduated projects show fewer paths running from socio-technical variables back to institutional ones. The authors read this sparse two-way pattern as evidence that project organizational structure and project governance form a loosely coupled system, which is the empirical claim most directly relevant to the feedback loops posited in the STG Framework. Granger causality establishes temporal precedence rather than causation and reports neither effect size nor sign, so these results indicate co-evolution of the governance and socio-technical dimensions without establishing the direction of influence.

Two measurement caveats bound the integrated reading of this section and the broader governance synthesis. First, the governance measures carry distinct sources of uncertainty. The Section 5.5 institutional-statement classifier has an F1 of 0.674, so misclassification enters the IS counts used here. Separately, the Section 5.4 cost-benefit analysis rests on moderate independent coder agreement ($\alpha = 0.66$), making its symmetry finding tentative. These uncertainties affect different governance measures and should not be treated as a single error pipeline. Second, the socio-technical network measures are computed from raw commit and email traces and are unaffected by classifier error. The integrated claims therefore rest on convergence across independently measured technical, social, and governance indicators, and the IS-based trajectory contrasts should be interpreted cautiously, especially in magnitude.

6. DISCUSSION AND CONCLUSIONS

In the above sections, we presented the full development of the Socio-Technical-Governance (STG) Framework, which integrates the Socio-Technical Systems perspective from Software Engineering with the Institutional Analysis and Development Framework. We then synthesized five source studies—one shared dataset and four empirical analyses—and mapped them retrospectively onto the STG Framework. Building on the initial articulation of the framework in Yin et al. (2026), this full development and synthesis-based assessment is a core original contribution of the present paper.

The reader might be asking: Why is yet another theoretical framework needed, when we have many already we can draw on, like IAD? The broader project supporting this work and the five source studies cited here was funded by the US National Science Foundation's Growing Convergence Research program (GCR). The general description of the GCR program (National Science Foundation [NSF], 2024) states: 'A distinct characteristic of convergence research, in contrast to other forms of multidisciplinary research, is that from the inception, the convergence paradigm intentionally brings together intellectually diverse researchers and stakeholders to frame the research questions, adopt common frameworks for addressing them, and create and implement innovative scientific approaches for their solution.' In short, as a 'Convergence Team' of scientists from a variety of disciplines, we found that to develop a common understanding we needed to build a theoretical framework that was useful and acceptable to the multi-disciplinary team trying to pull theories, methods and empirical work together. After several years of effort, the STG Framework is where we as a convergence team landed.

Turning to the empirical studies we synthesize, their findings are consistent with conceptualizing OSS as a form of digital commons requiring careful governance for long-term sustainability. Like traditional commons involving natural resources, OSS projects face collective action challenges including free-riding, coordination problems, and the ongoing need to maintain shared resources. However, digital commons also present unique characteristics: software is non-rivalrous, and digital communication enables coordination across geographic boundaries while introducing challenges of cultural diversity. The STG Framework provides analytical tools for understanding how these challenges manifest across social, technical, and governance dimensions. The importance of nested governance structures, evident in project-foundation relationships, parallels patterns observed in successful natural resource commons (Ostrom, 1990).

Our synthesis of the five source studies illustrates how independently conducted studies can be organized within the STG Framework and also suggests several practical implications for OSS practitioners, foundation administrators, and policymakers concerned with digital infrastructure sustainability. First, mentor selection and support deserve strategic attention given strong associations with outcomes; foundations may wish to match projects with appropriate mentors based on track record and expertise, invest in mentor development, and ensure adequate mentor bandwidth by monitoring workloads. Second, candidate early-warning indicators appear across all STG dimensions, although they await prospective validation; projects showing declining commit activity, shrinking communication networks, or reduced governance engagement may warrant closer assessment before problems become intractable. Third, cost-benefit balance in governance policies merits explicit consideration during policy design and review; arrangements that disproportionately burden one stakeholder group may undermine sustainability by creating resentment, discouraging participation, or concentrating knowledge among a shrinking group. Fourth, distinct trajectories observable within the first incubation year suggest value in early assessment and graduated intervention rather than end-of-process evaluation only.

A further contribution of this paper is that it illustrates the promise of using computational social science methods—machine learning, natural language processing, and network analysis—to study longitudinal properties of both macro- and micro-outcomes in collective-action settings. The methods demonstrated across the five synthesized studies illustrate an emerging research direction at the intersection of computational social science and institutional analysis, which we describe as computational institutional analysis: the application of computational text analysis, network methods, and longitudinal trace-data analysis to the study of rules, roles, authority, and institutional change—the study of 'Institutions for Collective Action,' as scholar De Moor (2021) refers to them. The ability to extract institutional statements using machine learning, combined with network analysis of social and technical interactions, enables new forms of empirical institutional analysis at scale. This direction builds directly on the work of many other scientists, including leaders such as Frantz and Siddiki (2022) and others[3] working on computational social science methods around the Institutional Grammar, other scholars working to develop methods for the extraction of institutional statements contained within rich bodies of text (e.g., Bushouse, et al. 2026; Frantz, 2025; and Rice et al., 2021); scholars applying the Institutional Grammar directly to OSS governance documents to extract and compare how projects codify roles and authority (e.g., Oliveira et al., 2026); and a larger body of work on computational text-as-data in fields of political science and communications (Grimmer and Stewart, 2013; Wilkerson and Casas, 2017; Grimmer et al., 2022). In these approaches, institutional arrangements are extracted and structured using coding schemes like Ostrom's rule types or the more detailed Institutional Grammar (Frantz and Siddiki, 2022). Advances such as these are broadening the reach of institutional analysis: institutional arrangements can now

be analyzed computationally, enabling comparison across contexts, tracking change over time, and identifying patterns associated with success and failure. This creates possibilities for developing candidate early-warning indicators that, once prospectively validated, could alert stakeholders to emerging problems in governance-related operations before they become critical.

Several limitations bound these contributions and together define an agenda for future work. Conceptually, the three dimensions of the STG Framework are analytically useful but not empirically independent, since technical architecture can encode governance choices and governance roles often emerge through patterns of contribution. The framework also foregrounds contributors and formal decision-makers, leaving users, downstream adopters, package maintainers, corporate sponsors, and other affected groups underdeveloped, even though adoption, user feedback, and dependency relationships may influence sustainability independently of internal project activity. STG at present therefore organizes candidate mechanisms rather than specifying a complete causal theory, and the relationships and feedback loops it represents require comparative and longitudinal testing across OSS settings. Empirically, the evidence synthesized here comes from a single setting, the ASF Incubator, whose formal mentorship structure and explicit graduation criteria make its working rules highly legible and therefore well suited to STG analysis. The governance measures also carry distinct measurement uncertainties: the Section 5.4 cost-benefit coding rests on moderate inter-rater agreement, while the Section 5.5 institutional-statement classifier has an F1 of 0.674. These limitations bound the precision of the governance-related conclusions and should be considered when interpreting the integrated evidence. Open source is governed through a wide range of arrangements, however, so the associations reported here await testing in other foundations and in independent projects. A first step in this direction is taken by Ashok et al. (2026), who model sustainability across multiple OSS ecosystems beyond the ASF. Future research should extend this STG Framework to other foundations and independent projects, incorporate user and adoption dynamics, prospectively validate the candidate indicators identified here before their use in monitoring tools, deepen theoretical integration with collective action theory and organizational behavior, and refine computational methods for institutional analysis across diverse application domains. The ultimate aspiration is providing OSS communities – or other Action Situations in other commons settings – with evidence-based guidance for enhancing sustainability, contributing to both scholarship and practice in supporting the digital commons upon which modern society increasingly depends.

NOTES

1 Defined as the future costs associated with relying on shortcuts or suboptimal decisions made during software development.
2 The 328 statements used as classifier training data by Yin et al. (2022) were drawn from an earlier SSRN working-paper version of Sen, Atkisson and Schweik (2022), before the corpus was finalized, and are therefore not interchangeable with the 259 operational-level statements reported in the published article, which we cite throughout.
3 See https://institutionalgrammar.org/resources/published-research/ for a list of the many contributions of scholars in this area.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge this support. The authors also thank the anonymous reviewers for their valuable comments and suggestions, which helped improve the quality of this paper.

FUNDING INFORMATION

This research was supported by the National Science Foundation under the Growing Convergence Research (GCR) program, grant numbers 2020751 and 2020900.

COMPETING INTERESTS

Charles M. Schweik serves on the Editorial Board of the International Journal of the Commons but was not involved in the editorial handling, peer review, or decision-making process for this manuscript. The other authors declare no competing interests.

AUTHOR AFFILIATIONS

Lin Zhou orcid.org/0000-0002-2812-1096
University of Massachusetts Amherst, United States
Charles Schweik orcid.org/0000-0002-0199-3740
University of Massachusetts Amherst, United States
Curtis Atkisson orcid.org/0000-0003-3575-6871
University of Washington, United States
Brenda Bushouse orcid.org/0000-0003-2079-1949
University of Massachusetts Amherst, United States
Mahasweta Chakraborti orcid.org/0000-0002-2431-7681
University of California, Davis, United States
Seth Frey orcid.org/0000-0002-5298-5089
University of California, Davis, United States
Ștefan Stănciulescu orcid.org/0000-0003-1569-3535
Mysten Labs, United States
Likang Yin orcid.org/0000-0002-4731-2672
Google, United States
Vladimir Filkov orcid.org/0000-0003-0492-4393
University of California, Davis, United States

REFERENCES

Ashok, A., Khan, N. I., Singhvi, S., Stanciulescu, S., Wang, Z., & **Filkov, V.** (2026). Multi-ecosystem modeling of OSS project sustainability. *Journal of Systems and Software*, 112985. https://doi.org/10.1016/j.jss.2026.112985

Atkisson, C. (2022). Mentors matter: Association of mentors with project success in the Apache Software Foundation Incubator. *Plos one*, *17*(8), e0272764. https://doi.org/10.1371/journal.pone.0272764

Bushouse, B. K., Virgüez-Ruiz, S., Chakraborti, M., & **Frey, S.** (2026, February). Advancing Text Analysis for Nonprofit Research: Using Semantic Role Labeling to Automate Institutional Grammar Coding of Nonprofit Laws and Policies. In *Nonprofit Policy Forum*, De Gruyter, Vol. 17, No. 1–2, pp. 323–341. https://doi.org/10.1515/npf-2024-0048

Coelho, J., & **Valente, M. T.** (2017). Why modern open source projects fail. In *Proceedings of the 2017 11th Joint meeting on foundations of software engineering* (pp. 186–196). https://doi.org/10.1145/3106237.3106246

Cohan, A., Beltagy, I., King, D., Dalvi, B., & **Weld, D. S.** (2019, November). Pretrained language models for sequential sentence classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3693–3699). https://doi.org/10.18653/v1/D19-1383

Crawford, S. E., & **Ostrom, E.** (1995). A grammar of institutions. *American political science review*, *89*(3), 582–600. https://doi.org/10.2307/2082975

De Moor, T. (2021). Three waves of cooperation: A millennium of institutions for collective action in historical perspective. In *Institutions of International Economic Governance and Market Regulation*. Oxford University Press. https://doi.org/10.1093/oxfordhb/9780190900571.013.8

de Laat, P. B. (2007). Governance of open source software: State of the art. *Journal of Management & Governance*, *11*(2), 165–177. https://doi.org/10.1007/s10997-007-9022-9

Ducheneaut, N. (2005). Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work (CSCW)*, *14*(4), 323–368. https://doi.org/10.1007/s10606-005-9000-1

Frantz, C. K. (2025). IG Parser: A Software Package for the Encoding of Institutional Statements using the Institutional Grammar. *arXiv preprint arXiv:2505.13393*.

Frantz, C. K., & **Siddiki, S.** (2022). *Institutional grammar*. Springer International Publishing. https://doi.org/10.1007/978-3-030-86372-2

Frischmann, B. M., Madison, M. J., & **Strandburg, K. J.** (Eds.). (2014). *Governing knowledge commons*. Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199972036.001.0001

Gamalielsson, J., & **Lundell, B.** (2014). Sustainability of Open Source software communities beyond a fork: How and why has the LibreOffice project evolved? *Journal of systems and Software*, *89*, 128–145. https://doi.org/10.1016/j.jss.2013.11.1077

Grimmer, J., Roberts, M. E., & **Stewart, B. M.** (2022). *Text as Data: A New Framework for Machine Learning and the Social Sciences*. Princeton University Press.

Grimmer, J., & **Stewart, B. M.** (2013). Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, *21*(3), 267–297. https://doi.org/10.1093/pan/mps028

Hecht, L., & **Clark, L.** (2018). Survey: Open source programs are a best practice among large companies. *The New Stack*.

Hess, C., & **Ostrom, E.** (Eds.). (2007). *Understanding knowledge as a commons: From theory to practice*. MIT Press. https://doi.org/10.7551/mitpress/6980.001.0001

Izquierdo, J. L. C., & **Cabot, J.** (2020). A survey of software foundations in open source. *arXiv preprint arXiv:2005.10063*.

Krippendorff, K. (2019). *Content analysis: An introduction to its methodology* (Fourth edition). SAGE. https://doi.org/10.4135/9781071878781

Madison, M. J., Frischmann, B. M., & **Strandburg, K. J.** (2010). Constructing commons in the cultural environment. *Cornell L. Rev.*, *95*, 657.

McGinnis, M. (2020). "Updated Guide to IAD and the Language of the Ostrom Workshop: A Simplified Overview of a Complex Framework for the Analysis of Institutions and their Development." Available at: https://mcginnis.pages.iu.edu/iad_guide.pdf.

National Science Foundation. (2024). *Growing Convergence Research (GCR)* (Program solicitation NSF 24–527). https://www.nsf.gov/funding/opportunities/gcr-growing-convergence-research/nsf24-527/solicitation

Oliveira, P., Conte, T., Gerosa, M., & **Steinmacher, I.** (2026, April). Governance in Practice: How Open Source Projects Define and Document Roles. In *Proceedings of the 19th International Conference on Cooperative and Human Aspects of Software Engineering* (pp. 255–266). https://doi.org/10.1145/3794860.3794911

O'Mahony, S. (2005). *Nonprofit foundations and their role in community-firm software collaboration*. https://doi.org/10.7551/mitpress/5326.003.0028

O'Mahony, S., & **Ferraro, F.** (2007). The emergence of governance in an open source community. *Academy of Management Journal*, *50*(5), 1079–1106. https://doi.org/10.5465/amj.2007.27169153

Ostrom, E. (Ed.). (1982). *Strategies of political inquiry*. Beverly Hills, California: Sage Publications.

Ostrom, E. (1990). *Governing the commons: The evolution of institutions for collective action*. Cambridge university press. https://doi.org/10.1017/CBO9780511807763

Ostrom, E. (2009). *Understanding institutional diversity*. Princeton university press. https://doi.org/10.2307/j.ctt7s7wm

Ostrom, E. (2011). Background on the institutional analysis and development framework. *Policy studies journal*, *39*(1), 7–27. https://doi.org/10.1111/j.1541-0072.2010.00394.x

Rice, A. K. (1953). Productivity and social organization in an Indian weaving shed: An examination of some aspects of the socio-technical system of an experimental automatic loom shed. *Human Relations*, *6*(4), 297–329. https://doi.org/10.1177/001872675300600402

Rice, D., Siddiki, S., Frey, S., Kwon, J. H., & **Sawyer, A.** (2021). Machine coding of policy texts with the Institutional Grammar. *Public administration*, *99*(2), 248–262. https://doi.org/10.1111/padm.12711

Riehle, D., & **Berschneider, S.** (2012, September). A model of open source developer foundations. In *IFIP international conference on open source systems* (pp. 15–28). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-33442-9_2

Schweik, C. M., & **English, R. C.** (2012). *Internet success: a study of open-source software commons*. MIT Press. https://doi.org/10.7551/mitpress/9780262017251.001.0001

Schweik, C. M., & **English, R.** (2013). Preliminary steps toward a general theory of internet-based collective-action in digital information commons: Findings from a study of open source software projects. *International Journal of the Commons*, *7*(2). https://doi.org/10.18352/ijc.397

Sen, A., Atkisson, C., & **Schweik, C.** (2022). Cui bono: Do open source software incubator policies and procedures benefit the projects or the incubator? *International Journal of the Commons*, *16*(1). https://doi.org/10.5334/ijc.1176

Stănciulescu, Ș., Yin, L., & **Filkov, V.** (2022, November). Code, quality, and process metrics in graduated and retired asfi projects. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 495–506). https://doi.org/10.1145/3540250.3549132

Steinmacher, I., Conte, T., Gerosa, M. A., & **Redmiles, D.** (2015, February). Social barriers faced by newcomers placing their first contribution in open source software projects. In *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing* (pp. 1379–1392). https://doi.org/10.1145/2675133.2675215

Trist, E. L. (1981). *The evolution of socio-technical systems* (Vol. 2, p. 1981). Toronto: Ontario Quality of Working Life Centre.

von Krogh, G., Haefliger, S., Spaeth, S., & **Wallin, M. W.** (2012). Carrots and rainbows: Motivation and social practice in open source software development. *MIS Quarterly*, *36*(2), 649–676. https://doi.org/10.2307/41703471

Wilkerson, J., & **Casas, A.** (2017). Large-scale computerized text analysis in political science: Opportunities and challenges. *Annual Review of Political Science*, *20*, 529–544. https://doi.org/10.1146/annurev-polisci-052615-025542

Yin, L., Atkisson, C., Chakraborti, M., Ruiz, S. V., Bushouse, B., Schweik, C., Frey, S., & **Filkov, V.** (2026). Governing the digital commons at scale: Detecting Ostrom rule types in OSS. *Journal of Systems and Software*, 113034. https://doi.org/10.1016/j.jss.2026.113034

Yin, L., Chakraborti, M., Yan, Y., Schweik, C., Frey, S., & **Filkov, V.** (2022). Open source software sustainability: Combining institutional analysis and socio-technical networks. *Proceedings of the ACM on Human-Computer Interaction*, *6*(CSCW2), 1–23. https://doi.org/10.1145/3555129

Yin, L., Chen, Z., Xuan, Q., & **Filkov, V.** (2021a). Sustainability forecasting for apache incubator projects. In *Proceedings of the 29th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering* (pp. 1056–1067). https://doi.org/10.1145/3468264.3468563

Yin, L., Zhang, Z., Xuan, Q., & **Filkov, V.** (2021b). Apache Software Foundation Incubator project sustainability dataset. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)* (pp. 595–599). IEEE. https://doi.org/10.1109/MSR52588.2021.00081

Zhou, M., & **Mockus, A.** (2015). Who will stay in the FLOSS community? Modeling participant's initial behavior. *IEEE Transactions on Software Engineering*, *41*(1), 82–99. https://doi.org/10.1109/TSE.2014.2349496

TO CITE THIS ARTICLE:
Zhou, L., Schweik, C., Atkisson, C., Bushouse, B., Chakraborti, M., Frey, S., Stănciulescu, Ș., Yin, L., & Filkov, V. (2026). Sustaining Open Source as a Digital Commons: Evidence from the Apache Software Foundation (ASF) Incubator Program. *International Journal of the Commons*, 20(1), pp. 576–595. DOI: https://doi.org/10.5334/ijc.1727

Submitted: 01 March 2026 **Accepted:** 05 October 2026 **Published:** 18 November 2026

COPYRIGHT:
© 2026 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See https://creativecommons.org/licenses/by/4.0/.

International Journal of the Commons is a peer-reviewed open access journal published by Ubiquity Press.