Studying OSS Sustainability via Socio-technical Structure and Institutional Governance

By

Likang Yin

Dissertation

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Vladimir Filkov, Chair

Premkumar Devanbu

Charles Schweik

Committee in Charge

2023

This dissertation is dedicated to my wife and my parents. For their endless love, support, and encouragement.

Contents

List of Figures

List of Tables

Abstract

Studying OSS Sustainability via Socio-technical Structure and Institutional Governance

Open Source Software (OSS) projects have become an integral part of our digital landscape, revolutionizing the way we develop, distribute, and consume software in various sectors. From operating systems to complex data analysis tools, OSS projects are the backbone of the global digital infrastructure. They offer a collaborative platform for developers worldwide, fostering the development and maintenance of high-quality software by leveraging the collective knowledge and expertise of a diverse, global community. However, despite their undeniable potential and benefits, OSS projects often face challenges related to sustainability, requiring effective governance, and a committed community of volunteering contributors. Understanding and addressing these challenges is crucial for maintaining the sustainability of OSS projects, ultimately benefiting the broader digital ecosystem.

This thesis aims to investigate the dynamics of OSS projects to understand the underlying factors contributing to their sustainability or lack thereof. This investigation primarily provides insights into the following research questions: How effective can we predict sustainability based on socio-technical traces of OSS projects? Can we identify the determinants for OSS sustainability along with their weights and directions? And, are there temporal associations between socio-technical structure and institutional governance? Answers to the above questions can help us design tools and methodologies to forecast the sustainability trajectory of OSS projects. This would allow stakeholders, such as project managers, contributors, and sponsors, to make informed decisions about resource allocation, project involvement, and risk management.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Professor Vladimir Filkov, for the continuous support of my Ph.D. study and related research. Your guidance, patience, and intellectual insights have been invaluable. Your mentorship has nurtured my scientific thinking and has equipped me with the necessary skills to undertake and complete this journey.

I would also like to thank the members of my committee, Professor Premkumar Devanbu, and Professor Charles Schweik, for their insightful comments and encouragement, which expanded my perspective on this research. Your comprehensive expertise and valuable feedback were instrumental in shaping this dissertation.

Finally, my sincere appreciation goes to my colleagues and friends in the research groups at UC Davis and UMASS for their camaraderie and collaborative spirit.

Preface

All chapters in this dissertation (except the introduction and conclusion chapters) correspond to work that has been published through peer review or will be submitted soon. The following are said works:

1. L. Yin, V. Filkov. Team Discussions and Dynamics During DevOps Tool Adoptions in OSS Projects. The 35th IEEE/ACM International Conference on Automated Software Engineering (**ASE 2020**).

2. L. Yin, Z. Chen, Q. Xuan, V. Filkov. Sustainability Forecasting for Apache Incubator Projects. The 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (**ESEC/FSE 2021**).

3. L. Yin, M. Chakraborty, Y. Yan, C. Schweik, S. Frey, V. Filkov. Open Source Software Sustainability: Combining Institutional Analysis and Socio-Technical Networks. The 25th ACM Conference on Computer-Supported Cooperative Work And Social Computing (**CSCW 2022**).

4. L. Yin, X. Zhang, V. Filkov. On the Self-Governance and Episodic Changes in Apache Incubator Projects: An Empirical Study. The 45th IEEE/ACM International Conference on Software Engineering (**ICSE 2023**).

5. A. Ramchandran*, L. Yin*, V. Filkov. Exploring Apache Incubator Project Trajectories with APEX. The 19th IEEE/ACM International Conference on Mining Software Repositories (***Co-first Author, MSR Tool Demo 2022**).

6. L. Yin, C. Atkisson, M. Chakraborty, C. Schweik, S. Frey, V. Filkov. How do OSS projects govern themselves in the wild? An Empirical Study on *Governance.md* Markdown Files.

Chapter 1

Introduction

Throughout history, from ancient bridge builders to modern software engineers, human beings have always aspired to create artifacts that embody long-term sustainability [1, 2]. Traditionally, research in software engineering has been primarily oriented toward the technical aspects of the software, e.g., algorithmic performance [3], software security [4], and code quality [5, 6], with technical metrics serving as the primary predictors of successful software [7]. However, the landscape has been progressively shifting, recognizing that the production of software requires developers' interactions [8]. This shift is partially attributed to the prevalence of Open Source Software (OSS) projects, which rely heavily on voluntary participation, e.g., feature suggestions, bug reports, and documentation of individuals from diverse backgrounds [9]. In this context, the path to a sustainable software product involves several stages that require not only proficient coding but also developers' interaction [10].

1.1 From Socio-technical to Institutional

Enhancing the technical parts of the software contributes directly and additively to the software's functionality and makes it more competitive, while undeniably crucial, often yields a 'one-time' increment in the software's overall value. However, investing in the social aspects of a project, such as fostering a healthy community and facilitating effective communication among team members, can amplify the value of the technical work done. This is because an engaged and cohesive community can lead to network effects where the value of the project grows exponentially with the number of active participants. Each new participant brings unique perspectives and skills, which can lead to novel solutions, more thorough testing, faster bug identification and resolution, and a richer and more comprehensive knowledge base. In this sense, improving the social aspects of software projects can be viewed as boosting the social capital that strengthens the technical efforts undertaken [11]. Therefore, while it's important not to neglect the technical aspects of software engineering, it's equally important to invest in the social infrastructure of the project to fully leverage the potential of network effects [12]. Similar belief has guided Apache Software Foundation (ASF), one of the most impactful OSS foundations, that 'community over code'[1]. This simple yet profound statement encapsulates a perspective that is becoming increasingly significant in the realm of software development, particularly in OSS projects [13].

Therefore, the socio-technical modeling approaches have been shown to be highly predictive of project outcomes over the past years [14, 15, 16]. On one hand, the technical aspect is typically gleaned from developers' code commits [17]. These commits represent the tangible contributions of OSS developers, showing how they collaboratively write, modify, and enhance the code. By analyzing the pattern, frequency, and nature of these commits, one can gain valuable insights into the technical progression of the project. On the other hand, the social aspect is often captured through the communication between OSS developers, predominantly through their email exchanges [8]. These discussions form the backbone of the social interactions within the project and can encompass a wide array of topics - from solving technical issues and proposing new features to strategizing project direction and coordinating collaborative efforts [18]. These email threads are informative data, revealing the social structures, relationships, and communication dynamics within the project [19]. As contributors engage with the complexities of code, they simultaneously navigate evolving social interactions, shaping unique configurations that exert a substantial influence on the project's outcomes [20]. Therefore, the integration of data from code commits and email communications in socio-technical models provides a holis-

[1]Apache Software Foundation 'Community Over Code': Link

tic view of the OSS project. This combined perspective enables the models to accurately capture the interplay between the social and technical elements of the project and thus yields a high predictive power concerning the project's outcome. Thus, socio-technical modeling offers a robust analytical framework for understanding, predicting, and guiding the success of OSS projects.

However, OSS projects, while often starting as small-scale endeavors, have the potential to scale dramatically [21]. This rapid growth with a high abandon rate brings sustainability to the forefront, shifting the focus from immediate success to long-term sustainability [22]. To study this, we draw upon the theoretical framework posited by Elinor Ostrom for shared resource systems [23]. Ostrom's design principles, grounded in the governance of common-pool resources such as forests [24] and fisheries [25], underline the imperative of established rules, norms, and governance mechanisms for their efficacious stewardship and long-term sustainability [26]. The significance of governance in managing such traditional commons elucidates its potential importance in the sphere of digital commons, with OSS representing a prime exemplar [27]. Therefore, it becomes evident that institutional governance could be instrumental in overseeing and ensuring the sustainability of OSS projects.

Approaching this from a metaphorical perspective, we can conceive the institutional rules and governance of OSS projects as the anchor to ships, keeping OSS projects within safe boundaries amid the fluctuating digital tides. We consider the integral role of governance in setting the course, standardizing the conduct, and structuring collaborative development, ensuring the project stays its course and remains robust under varying conditions. On the other hand, the socio-technical structure can be likened metaphorically to a weather vane to the ship. Just as a weather vane is finely attuned to changes in wind and provides immediate weather insight, the socio-technical structure is acutely sensitive to alterations within the project and its surrounding environment. Moreover, the project's socio-technical structure allows for the effective detection of shifts in collaboration dynamics, fluctuations in contribution patterns, or the advent of new technological trends. Thus, the project's socio-technical framework delineates the trajectory for cooperative development and interaction, it is the incorporation of robust governance mechanisms that emerges as a crucial counterpart, underpinning the long-term sustainability of OSS projects [28].

1.2 Summary of Contributions

This subsection serves to succinctly encapsulate the motivations underpinning each individual work incorporated in this thesis. Given that every piece of work included herein has undergone the rigor of peer review for submission to conferences in software engineering and human-computer interaction, readers might find some repetitions in the contextual description and methodological elaborations in each chapter. This repetition is intentional, designed to allow each chapter to stand on its own merit, thereby facilitating independent comprehension. Detailed expositions of all research activities described in this thesis are presented comprehensively in the respective chapters.

1.2.1 Chapter 2: Team Discussions and Dynamics During DevOps Tool Adoptions in OSS Projects

In this chapter, we delve into the ongoing adoption events of DevOps tools in Open Source Software (OSS) development. These practices are progressively moving away from de novo programming and towards the utilization of pre-made tools. This trend can largely be attributed to the popularity of the DevOps software development movement which advocates for rapid production changes, without compromising software quality. This is achieved primarily through automation of processes for building, testing, and deploying software, negating the need for coding from scratch.

However, DevOps engineers often lack the necessary support to make informed decisions about the tools they should use, due to a lack of comprehensive empirical evidence on the effectiveness of DevOps practices. Consequently, decision-making often relies on scattered knowledge, anecdotal evidence, and team members' individual strengths and experiences. Previous related studies have been conducted in narrower settings and are often based on a small number of commercial software projects, making their findings hard to generalize, especially for OSS projects. These studies relied heavily on surveys, thus being dependent on individual experiences and feedback.

Motivated by the availability of large, comprehensive datasets on tool adoptions from various GitHub projects, we undertake a deep investigation into the social determinants of team discussions and dynamics leading up to tool adoption events in OSS projects. Through our research, we treat the adoption of a tool by a project team as the adoption of an idea by a community, utilizing the diffusion of innovation theory and social judgment theory to discern the significant factors underlying tool adoptions. Using a dataset of social and technical traces from numerous GitHub projects, along with project-tool adoption data, we undertake exploratory data analyses, and in-depth case studies, and build regression models. We find that initiating a project-wide change, such as adopting a new tool, is also a complex social process, especially when the team is diverse in terms of developer tenure, prior exposure to the tool, and day-to-day involvement in the project. Moreover, we find substantial changes in team dynamics around the time of DevOps tool adoptions, which are further associated with discussion length and tool adoption likelihood.

In summary, this chapter presents a comprehensive look at DevOps tool adoptions, focusing not just on the effects of tool choices on software engineering outcomes but also on the underlying social dynamics and discussion patterns within development teams.

1.2.2 Chapter 3: Sustainability Forecasting for Apache Incubator Projects

OSS has revolutionized the software development landscape, attracting developers for various reasons, from honing their skills to contributing to significant goals. However, more than 80% of OSS projects are abandoned over time. This chapter seeks to address the crucial question: Why do some nascent OSS projects succeed while others do not, and can the latter be helped?

To answer the questions, we turn to Apache Software Foundation's (ASF) Incubator, which provides mentorship and guidance to nascent projects, helping them adhere to ASF rules and regulations and progress towards self-sustainability. However, even with this support, many projects fail due to challenges in understanding and meeting the graduation requirements, which demand not just coding skills, but effective teamwork and sustainable community development.

This chapter, therefore, aims to model the process that ASF incubator projects follow towards becoming sustainable and predict project binary outcomes (graduation/retirement) early on. Using the socio-technical modeling perspective and historical data from the ASF incubator, we identify features that distinguish graduating projects from retiring ones and build temporal forecasting models to predict sustainability outcomes at any point in project development. Our methodology includes gathering project technical and social traces, constructing temporal social and technical networks for each project, and conducting exploratory data analyses and case studies to build a forecast model and offer timely advice. Our contributions include a novel longitudinal dataset of OSS projects under ASF regulation, the first OSS project sustainability predictor with over 90% of F-1 score with only the first 8 months of the project's data, findings about the project socio-technical metrics associated with sustainability, and a strategy for real-time monitoring of sustainability forecasts.

In summary, the chapter demonstrates that the end results of OSS projects' sustainability trajectories can be effectively forecasted and potentially corrected in time.

1.2.3 Chapter 4: Open Source Software Sustainability: Combining Institutional Analysis and Socio-technical Networks

In this chapter, we first explore two prominent perspectives that attempt to explain OSS sustainability: a software engineering angle, which investigates the socio-technical aspects serving as the abstraction of organizational structure, and a management perspective, which emphasizes institutional designs serving as the abstraction of institutional governance. We utilize the ASF incubator dataset that comprises historical traces of OSS developers' activities. We use these digital traces to analyze the differences between successful (graduated) and unsuccessful (retired) projects from both the socio-technical and institutional governance perspectives.

Then, motivated by Ostrom's Nobel prize-winning theory, we merge these two approaches in order to understand and address the sustainability challenges faced by OSS projects in a holistic way. On the socio-technical side, we construct longitudinal social and technical networks for each project and compute various measures that describe the socio-technical networks. Simultaneously, on the institutional governance front, we trained a sentence-level classifier on manual annotations of institutional statements in the ASF participants' email exchanges. We find that successful projects tend to be more socially active, often showing active contributions to documentation and communicating policy guidance. Additionally, we find that the institutional governance in OSS projects, such as rules, norms, and regulations, serve as guiding structures that facilitate temporal collective actions in OSS development.

In closing, we bridge the gap in previous studies by investigating the relationship between socio-technical systems and institutional governance in OSS projects. We believe that the simultaneous analysis of structural and institutional factors will pave the way for novel approaches to studying emergent properties like OSS sustainability.

1.2.4 Chapter 5: On the Self-Governance and Episodic Changes in Apache Incubator Projects: An Empirical Study

The chapter stresses that many OSS projects fail to maintain their sustainability due to various factors, it is underscored that efficient governance can significantly impact their socio-technical trajectory through episodic changes, which are intentional, periodic, and intermittent.

To illustrate this point, the chapter provides a motivating example of how the Apache Software Foundation project OpenWhisk managed to avert a decline in its active developer base and potential abandonment by proactively enforcing regulations and engaging more volunteers. The chapter hypothesizes that sustainable OSS projects can translate governance more efficiently into a socio-technical structure than those that are not, through episodic change intervals.

The chapter proceeds to validate this hypothesis by analyzing data from hundreds of OSS projects in the Apache Software Foundation (ASF) incubator, a perfect exemplary foundation that fosters sustainability through robust institutional policies and governance. The sustainability status will be assigned to a project upon its exit from the incubator and is determined through an evaluation by ASF committees, an external approach that provides valuable insights into the concept of sustainability.

An empirical study was conducted to identify patterns in sustainable and unsustainable projects by comparing episodic changes in their socio-technical structure with institutional discussions. This exercise was operationalized by matching episodic time-series events with sentence-level institutional discussions. We find significant associations between episodic changes in the socio-technical structure and the sentiment attached to institutional discussions. It was also observed that episodic changes impacted both individuals and projects differently, based on their sustainability levels. Notably, we find that sustainable projects proved more adept at translating institutional rules into practice during periods of episodic changes.

In summary, the chapter concludes by emphasizing the novelty of this unified analytical approach toward understanding the episodic changes in OSS projects and their self-governance.

1.2.5 Chapter 6: Exploring Apache Incubator Project Trajectories with APEX

In this chapter, we introduce an online, interactive, dashboard-like tool, 'APEX' [2] to empower OSS projects to act proactively and adjust their sustainability forecasts.

The tool APEX is designed for nascent projects housed in the ASFI incubator, enabling them to monitor their sustainability trajectories over time. This leads to timely course corrections and increases the likelihood of projects graduating into the ASF incubator. APEX, while tailored for ASF projects, is generically designed, allowing for easy adaptation to accommodate data from other repositories.

We contrast APEX with existing tools, such as the ASF's Clutch tool, demonstrating how APEX complements Clutch by offering more comprehensive analytics into the socio-technical facets of projects and actionable insights. By integrating our work that showed early predictions of OSS project sustainability, in comparison to other tools outside the ASF domain like Augur and GrimoireLab, APEX stands out by providing a synthesis of metrics into longitudinal sustainability forecasts and permitting deeper analysis of socio-

[2]APEX is available online: https://ossustain.github.io/APEX/

technical features.

To demonstrate the functionality and utility of APEX, we provide a detailed introduction of the tool, followed by use cases showcasing its use in monitoring events in ASF projects, pinpointing long-term engagements in developers, and enabling within-ASF project comparisons.

1.2.6 Chapter 7: How do OSS projects govern themselves in the wild? An Empirical Study on `Governance.md` Markdown Files

In this chapter, we examine the increasing prominence of (OSS) repositories on GitHub as digital commons, bearing resemblances to traditional commons such as fisheries, forests, and irrigation systems. However, OSS projects present unique challenges regarding sustainability, necessitating innovative approaches due to their undegradable nature. For this purpose, we draw upon Elinor Ostrom's governance theories, typically employed for managing natural resources, to interpret the dynamics and organizational mechanisms of OSS projects.

Recognizing the need to bridge the gap between Ostrom's governance theory and its application to OSS, we harness the rich timestamped data from OSS repositories, and specifically, the `GOVERNANCE.md` files. These markdown files contain a recorded history of project rules, enabling a deep understanding of a project's governance evolution. The overarching proposition of our research is the criticality of striking a balance between under-governance and over-governance in OSS projects, similar to the dosage effects in medicine. We aim to guide software communities in creating optimal governance rules by drawing on insights from Ostrom's theory.

The merits of this research extend to both management science and software engineering fields. We gathered project historical data, including code commits, issues, and comments, from 703 GitHub repositories containing `GOVERNANCE.md files`. Using a manually annotated dataset, We develop a high-performing sentence-level classifier capable of identifying seven different types of Ostrom's design rules within the `Governance.md` files, facilitating an automated, large-scale analysis of OSS project governance mechanisms.

Our findings provide insights into the evolution of governance rules within a project, offering guidance for software engineers on when and how to modify rules for the project's benefit.

In conclusion, this chapter represents an innovative endeavor to apply Ostrom's governance rule to study the sustainability of OSS projects. While our research focuses on GitHub projects, the implications could prove significant for numerous other software projects seeking sustainability in the digital commons.

Chapter 2

Team Discussions and Dynamics During DevOps Tool Adoptions in OSS Projects

2.1 Introduction

OSS software development practices are evolving away from *de novo* programming and toward adopting pre-made tools for various tasks, which are then integrated into existing development and production pipelines [29, 30]. Part of the reason for this has been the popularity of the DevOps software development movement, which seeks to bring changes into production as quickly as possible without compromising software quality, primarily by automating the processes of building, testing, and deploying software. In practice, DevOps is supported by a multitude of configuration management, cloud-based continuous integration, and automated deployment tools, short-circuiting the need for coding from scratch [31, 32]. Using pre-made tools can shorten the development process, so long as an appropriate set of tools is used and properly integrated into a development pipeline.

Tool adoption decisions, however, are often not well informed. DevOps engineers frequently lack the decision-making support to help them discern the best choices among the many tools available [32]. In large part, that's because current empirical evidence on the effectiveness of DevOps practices is, at best, fragmented and incomplete. While questions about best tools and practices in DevOps abound in online forums, the existing answers are typically generic rules of thumb, or dated advice, mostly based on third-party experiences, often non-applicable to the specific context. While they likely consider that scattered knowledge, teams seem to leverage their strengths and experiences in making tool adoption decisions. Some tool adoption events in projects are preceded by a discussion among team members on the issues involved [32], but it is not clear what is discussed in them among team members and how those discussions correlate with adoption decisions.

Moreover, instituting a project-wide change is a complex social process when there are many stakeholders [33]. Adopting a new tool, e.g., can require a team-wide adjustment in practices that affect every developer–thus they are all stakeholders in the adoption decision. This is especially true when the team is more diverse in terms of developer tenure with the project, their prior exposure to the tool being considered for adoption, and their day-to-day involvement in the project. Naturally, supporters and detractors can and do arise over decisions when developers espouse different views toward a tool, resulting in champions and detractors, and sometimes arguments can get emotional [34, 35]. Many of these individual and team-level factors may contribute to the ultimate adoption decision, but which ones actually do? And in what proportion?

Inspired by the availability of large, comprehensive data sets on tool adoptions from diverse GitHub projects, here we undertake both qualitative and quantitative methods to uncover the social determinants of team discussions and dynamics leading to tool adoption events in OSS projects. We operationalize our study at the team-level instead of at the individual level. Central to our study is the analogy that a project team adopting a tool is akin to a community adopting an idea. We use theories on diffusion of innovation and individual and team social judgment to guide us in discerning the important factors underlying tool adoption.

We start from a data set of social and technical traces from a large number of GitHub projects, together with project-tool adoption data for each. Then, we perform exploratory data analyses, do deep-dive case studies, and built regression models to determine how team properties and their communication behaviors are associated with tool adoptions. We find that:

- Team dynamics changes around adoption time in substantial ways, and some of those changes remain with the project.
- Project teams undergo meaningful discussions and exhibit significant dynamics changes in the period before and after a new tool is adopted.
- Influencer developer's participation is associated with shorter discussion length, and likelier tool adoption.
- New developers are positively associated with longer discussion length and lower adoption success.

Related questions have been asked before, in narrower settings. Zhu et al. [36] use code accept/ignore rate to compare the goodness between issue tracker and pull request systems, but less focus on the adoption dynamics. Using surveys Xiao et al. [37] find that coworker recommendation is a significant determinant of security tool use. Witschey et al. [38] find that the strongest predictor of using security tools is the ability to observe their coworkers using those tools. These findings are based on a small number of commercial software projects and can be difficult to generalize, especially to OSS projects. Moreover, surveys, by their nature are based on individual experience and feedback. Kavaler et al. [32] have looked more comprehensively at a large swath of JavaScript projects, but have focused more on the effects of tool choices on software engineering outcomes, and found that some choices are better than others.

2.2 Background and Theories

Here we position our current work in the space of representative prior work on tools, tool adoptions, and OSS teams, as well as present two most relevant sociological theories that will guide our hypotheses and research questions.

2.2.1 Tools, Teams and Adoptions

Software developers use various techniques to implement and maintain software, including, at times, adopting new technology [39]. T. Gorschek et al. [40] describe five different stages in technology transfer, including identifying potential problems, formulating issues, proposing candidate solutions, validations, and releasing solutions. S.L Pfleeger [41] proposed a model of technology transfer that can be tailored to a particular organization's needs. Riemenschneider et al. [42] find that opinions of developers' coworkers and supervisors on using some technology matters to an individual developer when they consider whether to use this method. Our paper extends such implications to tool adoptions in OSS projects.

Marlow et al. [43] find that length of tenure may be associated with attitudes toward new tools. More senior developers may get more attached to a working style, thus making them less flexible to adopt new tools. New developers [44], in contrast, need less time to adapt to a new working style due to lack of history in the project, and eagerness to learn/follow technology trends. Xiao et al. [37] find developers are more likely to adopt a tool if their peers or co-workers are using or have used it. Also, if experienced individuals join a project, their knowledge and social ties move with them to the project [45, 46]. Thus, the sum-knowledge of the whole team is constantly changing as the project evolves. Likewise, previous research has found that emotional contagion from co-workers can percolate within the team [47, 17] causing a slowdown in communication and productivity.

With publicly available traces of working records and discussions, it is possible for researchers to study team dynamics before, during, and after a change. However, studying tool adoptions is complex because of the variety of contributing factors. Previous work [48] has found that changing previous practices can force software developers out of their safe zone, resulting in nonconformity with the methodology. Zelkowitz et al. [49] find that many software professionals are resistant to change, and that timing is of importance when adopting new technology. Johnson et al. [50] consider that having intuitive defect presentation may contribute to the willingness of using static analysis tools. Related literature [37, 38] shows that developers' social networks (e.g., through discussions about tools [51]) benefit the spread of tools. Poller et al. [52] pointed out correlations between organizational factors and the success of security practices.

2.2.2 Diffusion of Innovations Theory

Diffusion of innovations (DOI) theory was first proposed by Rogers in 1983 [53], and has since become popular in socio-technical fields. Rogers considered diffusion as the process by which an innovation unfolds over time through communication channels among the population, and found that some properties of innovation are crucial to an adoption: perceived usefulness, perceived complexity, peer pressure, etc. The innovation diffusion process at an organizational level can be summarized as having two stages [53, 54]: initiation (perceiving the issues, and knowing they can be addressed by adopting certain innovations), and implementation (adopting the innovation and customizing it to fit own culture if necessary, then the team-collected information reinforces/devitalize the adoption until the innovation becomes a part of the organization).

In the context of software engineering, DOI theory can be of vital value to offer understandings of the phenomenology and consequences. E.g., tool builders want to know if anyone will use the tool they built, and how to persuade the community to use it. For general developers, they want to know if there exist tools to help them be more efficient. However, even if a tool can be beneficial, individual resistance may still exist. To reduce such individual resistance, organizational mandate has been shown to have influence on adoptions [48, 42]. However, those authors also find that the organizational mandate is not sustainable compared to other factors. A catalog of non-coercive adoption patterns has been proposed [54], to help organizations achieve successful software engineering practices in a more persuasive manner.

In GitHub OSS projects, Bertram et al. [55] found that communication and knowledge sharing exist for coordinating work in issue trackers. This suggests that developers within a project communicate and learn from each other [56]. Moreover, knowledge sharing exists even across projects. Singer et al. [54] noted that some developers use GitHub to learn how other projects use the same tools in their projects. Thus, developers in the GitHub ecosystem, and in its tighter sub-ecosystems, share and distribute knowledge about tools through participation in different projects, thus creating a diffusion process.

2.2.3 Social Judgment Theory

Social Judgment Theory (SJT) [57] was proposed to study how people self-persuade to adopt a new idea when they encounter it. According to SJT, when people are exposed to new information or a new environment, they tend to consider three things. First is their previously formed attitude, or *anchor* to which they compare the new idea [58]. Then they look at available *alternatives*. In this process, people recognize themselves, form their views, and express their ideas. Finally, there is the *ego involvement* or the centrality of the issue being considered to a person's life, which can explain why people can accept some ideas and novelty easier than others. Individuals with high ego involvement on an issue tend to be more passionate on the topic and are more likely to evaluate all possible positions [59]. High-ego individuals also have a larger latitude of rejection, and it is difficult to persuade them to adopt a new idea. In contrast, low-ego individuals tend to have a larger non-commitment latitude, meaning they often do not take a stance on an issue, and they do not care much about the arguments.

There are several ways to aggregate the judgments of individuals from a group into a *group judgment* [60, 61]. *Mathematical aggregation* amounts to simple counting/averaging of individual judgment. On the other hand, *behavioral aggregation* is the outcome of group members agreeing after discussing the matter. Experimental evidence suggests that group judgment is generally more accurate than individual judgment, and how it is measured can be significant to the outcome [60]. However, others found that the superiority of group judgment is due to reliability produced by larger samples [62].

In the context of OSS projects, team discussions on tool adoptions transpire during which possible options are proposed. These discussions can be considered a form of behavioral aggregation of individual opinions. However, group judgment may get biased, as high-ego individuals with strong opinions can potentially sway a group decision in their direction, even if it offers no overall benefit.

2.3 Hypotheses and Research Questions

Central to this paper is the analogy that adopting a tool is akin to adopting an idea. An OSS project involves a social organization in which activities are coordinated through communication to achieve both individual and collective goals. By coordinating activities, the organizational structure has to be created to assist individuals to communicate. In GitHub projects, in particular, the two main communication channels are through code committing and issue posting. Moreover, contributing code changes and approving others' pull requests suggests that the developers are mutually aware of each other, and this forms the basis for communications and discussions.

The DOI theory suggests that innovations and new technologies can be spread to teams through diffusion. For tool adoption in GitHub projects, this diffusion happens through information exchange within and between projects, through their developers. One mechanism is through reading/participating in discussions that are accessible to all team members in a project. The publicly available traces of commits and comments allow us to study discussion dynamics of how tools are perceived, discussed, and then finally adopted. Thus, a hypothesis arising from DOI is that developers who have previously had exposure to certain tools will be more knowledgeable of them and contribute to a discussion on it, to make the adoption process smoother and faster.

On the other hand, SJT suggests that since it is likely some developers prefer adopting a tool more strongly than other developers do, the former naturally will have high-ego on adoptions. A hypothesis arising is that developers who post more comments on tools than others will be the influencers in the discussions, and will affect the direction of the adoption. Moreover, contributing to cognitive dissonance, people will react more strongly to negative information than positive information [63]. Therefore, another hypothesis is that the people's discussion sentiment (positive or negative) may correlate with eventual tool adoption.

We formalize the above into our Research Questions (RQs), as follows. First, we seek to uncover the patterns and changes happening at team-level, in the period surrounding tool adoption events.

RQ$_1$: What is the team dynamics when a project team goes through the process of adopting a new tool?

Then, we want to understand what goes on in the discussions before and after the tools are adopted.

RQ$_2$: What topics are discussed during tool adoptions? How are people's sentiments evolving toward the tools they are adopting?

Next we look at notable individuals in the discussions. According to the aforementioned theories, people who care the most and comment voluminously, i.e. influencers, may play a significant role in the innovation diffusion process. Namely,

RQ$_3$: Are tool adoption events associated with influencers? How much does their opinion weigh in on others?

In the final thrust, to comprehensively understand tool adoption and discussions, we quantitatively model adoptions and discussion length in terms of our chosen variables using multiple regression.

RQ$_4$ What are the quantitative determinants of project-wide tool adoption and the preceding discussion length?

2.4 Data

We start from a data set by Trockman et al.[64] of GitHub *npm* projects that have adopted any of 19 different DevOps tools [1]. They were collected by gathering tool badges from the projects' `README.md` files. Some projects use badges to signal important information [65], e,g, code coverage percentage codecov 100%, number of downloads per month downloads 1.1k, package dependencies npm v7.18.3, and continuous integration status build passing. The tool adoption data is current as of Jan 2019. The data set consists of 52,923 distinct GitHub projects, the adopted tools, and the adoption dates. In total, 96,176 tool adoptions are identified, or about 2 tool adoptions per project on average. Among them, 28,430 projects adopted only one tool, 11,272 projects adopted two tools, 8,900 projects adopted three tools, and 3,378 projects adopted four tools. Projects which have adopted more than four tools are fewer than 2% of the total.

[1]Data and code is available at `https://github.com/lkyin/tool_adoptions`

The collected tools can be classified into the following six categories according to their functionality. We illustrate each category with an example tool. Browser testing: e.g., *Selenium* is an automated testing framework for automated testing of web applications and User Interfaces (UIs) [66]. Test Coverage: e.g., *Coveralls* measures software quality in terms of test cases line coverage, function coverage, branch coverage, and statement coverage [67]. Minifier: e.g., *Uglifyjs* is a JavaScript compressor tool used to merge and minimize JS resources by removing blank rows, shorten the variables and functions names to make the web applications load faster [68]. Testing: e.g., *Mocha* has good support for testing asynchronous code, allowing any use of failed exception test libraries [69]. Linters: e.g., *ESLint* is a plug-in JavaScript code style/error detection tool [70], thereby achieving effective control of the quality of the project. Dependency managers: e.g., *Snyk* helps developers track the dependency tree to find which module introduces the vulnerability [71, 72]. The categories and the summary statistics of adoption data is given in Table 2.1.

2.4.1 Data Collection and Cleaning

We use the *GitHub API (v3)* [73] to collect and extract historical records of the commits and discussions from the GitHub projects. For commits data, the author is the one who made the changes, while the committer is the one who committed the changes to GitHub. Thus, we align the 'author' with each commit. For discussions, since the commit messages are usually not meaningful [74], we only collect and use issues and comments as their discussions.

Some projects have very low levels of activity, while others have team sizes that may be too small to study their dynamics. To avoid those, in this paper commits and comments of only the durable and persistent projects are collected. We set three requirements: (1) *Durable*: The projects that have commit records spanning at least two years. (2) *Persistent*: The projects having at least *6* months of commit history before and after the tool adoption event and have at least *50* total commits. (3) *Related*: The projects having at least one comment related to the adopted tools. After filtering, the final data set consists of 684 distinct GitHub projects.

Since we focus on discussion comments which specifically associate with tools, we

Table 2.1. Tool adoption event summary

Tool	Task Class	Per Tool	Per Tool Category
karma	Browser	4116	6157
sauce		1654	
selenium		387	
coveralls	Coverage	14430	21626
codecov		4239	
codeclimate		2731	
codacy		213	
coverity		13	
uglify	Minifier	2018	2124
minimist		62	
minifier		44	
mocha	Testing	33280	47119
istanbul		11864	
jasmine		1975	
eslint	Linter	10978	18969
standard		4827	
jshint		3164	
snyk	Dependence Manager	179	181
gemnasium		2	

filter out all discussion comments which do not explicitly mention a tool name (of the 19 tool names), with the following exception. If an issue starts a thread and mentions a tool name, the replies to that thread are likely a part of the discussion, therefore, all comments following the tread are also included even if they do not contain a tool name. On the other hand, some issues are automatically posted, e.g., by continuous integration (CI) tools; we identify the comments by checking the title of the comments (e.g., '[Snyk Update]') and exclude the comments from the data set.

We identify and remove the following strings from comments that do not comprise text, as they would bias downstream analysis: non-ASCII characters (by checking if all characters are from 'u4e00' to 'u9fff'), code snippets (by checking if the comments are enclosed with single or triple backticks), URLs (by using the regular expression from the *re* module in *Python 3.7* with pattern 'https?://S+'), and emojis by checking the encoding of the characters. Finally, since some comments mention multiple tools at the same time, therefore, they are counted multiple times. To avoid such duplication by those comments (about 1.77% of the total), we manually re-label these comments with only one tool, we achieve this mainly by referring the outcome of the adoption (which one is finally adopted) and context of the discussion.

2.4.2 Variables Used

The variables that we use in this study have been identified based on our discussion and consideration of the underlying theories. They include the following.

Outcomes: Adoption success and discussion length. *Adoption success (adoption_success)* is a binary variable (0="No" or 1="Yes") indicating whether a tool was adopted in a project. A successful adoption is if a tool is being used in a project, regardless of whether discussions on it ever happened. An unsuccessful adoption is if a project's team had a discussion on a tool yet they never adopted it. *Discussion length (discussion_length)* for a project and a tool is calculated as the number of months from the first day the tool was mentioned in the project discussion, until the tool was adopted. The discussion length suggests how long it takes for teams to reach an agreement and eventually adopt the tool, although in practice can be much longer if the team keeps coming back to discussing a tool after longer breaks without mentioning it. To address those cases we introduce a control variable *num_mentions*, see below, which measures the volume of tool mentions in a discussion.

Controls: Project Age, Number of Commits, Number of Comments/Mentions. *Project age (project_age)* at the time of adoption of a tool is the number of months from the first commit date in the project to that specific tool's adoption date. *Number of Commits (num_commits)* at the time of adoption is the number of commits made during the discussion (based on the discussion length above) on that tool, in the project. *Number of Comments (num_comments) / Mentions (num_mentions)* is defined as the number of comments (including all follow-up comments in the same thread) made in that tool's discussion interval, while the number of mentions only counts the number of comments which explicitly mention the tools, for a given project. *Tool (tool)* is the full name in lowercase of the corresponding tool (e.g., eslint).

Team Metrics: We calculate team measures for each project, for each tool discussion. *New Developers (num_new_dev)* at time t in the project are those developers who have made their first contribution (either by committing code changes or participating in discussions) within the *3* months prior to t. For convenience, we refer to all other developers who are not new developers as *Senior developers*. *Developers with prior tool exposure (num_w_tool_expos)* are the developers who had already been in a project before time t that had used that tool, and have committed code changes before time t to current project (but after their contributions to the other, tool using project). In contrast, the developers without prior exposure are the ones who did not participate in a project that had used the tool before time t. *Involved Developers (num_involved_dev)* are the developers who have been involved (i.e., participated) in the tool discussion. *Positive Developers (num_pos_dev)* are the developers who have posted overall more comments with positive sentiment than negative sentiment in the tool discussion. While *Negative Developers (num_neg_dev)* are the opposite. The descriptive statistics for the metrics are shown in Table 2.2.

2.5 Methods

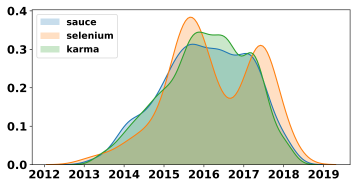
2.5.1 Topics Identification

We use Latent Dirichlet Allocation (LDA) [75] to study topics in discussions. LDA is a statistical technique used to identify topics in large documents and high-frequency words associated with the topics. LDA yields a topic probability distribution for each document, enabling topic clustering and/or text classification across all documents in a set.

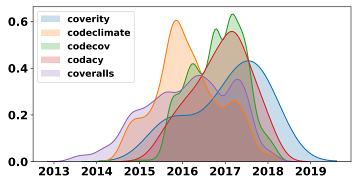
Before training the LDA model, we pre-processed the GitHub comments by tokenizing with the Apache Open NLP library [76], and stemming with the Porter stemmer (this removed all stop-words from the comments). Due to the large number of discussions we have, many high-frequency words are not very meaningful. To get a corpus with a higher

Table 2.2. Summary statistics for our 10 variables over all 1,085 adoption events (after removal of top 2% as outliers)

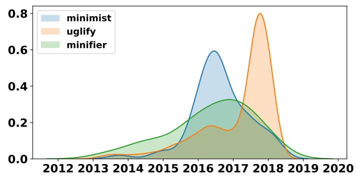
Statistic	Mean	St. Dev.	Min	Max
adoption_success	0.798	0.402	0	1
discussion_length	9.927	11.428	0.033	56.633
project_age	33.542	20.688	6	100
num_comments	15.852	32.792	1	272
num_commits	338.246	588.521	0	4,163
num_new_dev	18.852	32.933	0	252
num_w_tool_expos	7.724	13.973	0	133
num_involved_dev	2.373	2.377	1	23
num_neg_dev	0.678	1.077	0	7
num_pos_dev	0.641	0.975	0	8



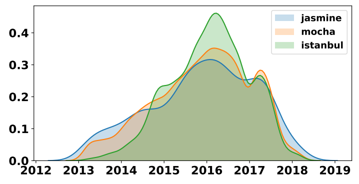
(a) Browser



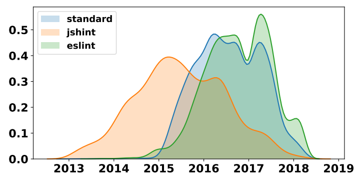
(b) Coverage



(c) Minifiers



(d) Testing



(e) Linter

Figure 2.1. The adoption time distributions of tool adoption events of five tool categories.

concentration of topics [77], we removed both the 5% words with the lowest frequency (e.g., user names) and the 5% words with the highest frequency (e.g., bugs). After this, the LDA model is more able to distinguish topics from each other.

2.5.2 Sentiment Analysis

A sentiment analysis tool identifies the emotional characteristic of a text, typically in terms of its aggregate positivity or negativity. Many sentiment analysis tools used in software engineering are trained solely on social media corpora, e.g., Twitter, Facebook, and Yelp. However, some words in the context of software engineering can represent a different meaning compared to social media corpus [17, 78]. For example, to ‘kill a process’ is neutral in the context of programming, while some sentiment analysis predictors trained on other media corpus, treat it as a strong indicator for negative emotion. To avoid such issue, we use Senti4SD to predict the sentiment of the GitHub comments. It has been trained on Stack Overflow annotated comments, and it has been shown to be more accurate in the software engineering domain [79, 80]. Senti4SD yields ternary sentiment for each comment, i.e., positive, neutral, or negative. The following are examples from our data of positive, neutral, and negative comments, as per Senti4SD (sensitive words are anonymized and replaced by ¡notation¿). *Positive*: “sure ¡user¿! appreciate your point, thanks for the suggestion.” *Neutral*: “Let’s have this project actually be ¡tool¿ and not try to duplicate on our own what ¡tool¿ does internally. Let’s just let ¡tool¿ handle all merging itself.” *Negative*: “BTW, ¡tool¿ failed because you added a function without tests.”

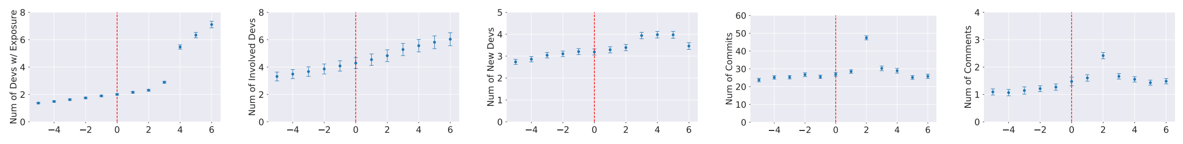
GitHub discussion comments can still be different from the Stack Overflow comments. To verify that Senti4SD can effectively identify sentiment in comments on GitHub, we selected a random set of 50 comments and via observation determined them to contain 25 comments of neutral, 13 comments of negative, and 12 comments of positive sentiment. Then we ran them through Senti4SD and found that 6 were mispredicted by Senti4SD, showing the accuracy of 88% (2 neutral comments were deemed negative, while 1 positive and 3 negative comments were deemed neutral).

We aligned each post with their Senti4SD derived ternary sentiment (i.e., positive, neutral, and negative). We find that 23.8% of the discussions are positive, 48.4% of the comments are neutral, and 27.8% of them are negative. The average length of the comments is 194 characters for neutral, 242 characters for positive, and 422 characters for the negative. This suggests that the negative comments may carry more information than other two types.

2.5.3 Linear Mixed-effect Regression

We use Generalized Linear Mixed Effect Regression (GLMER) models (*glmer* package in *R*) to study the contribution of our independent variables to explain the variability in the outcome variable, while mixing fixed and random effects. The *tool* is used as the random effect in the models.

We use logistic regression for modeling the adoption success and generalized regression with the Poisson family for the modeling discussion length. To avoid convergence issue we use the *bobyqa* optimizer. We use *scale()* function to z-normalize each variable as they vary across orders of magnitude between variables. To avoid influential points caused by outliers in the fixed effects, we remove the projects with top 2% *num_commits*, *num_comments*, and with top 1% of the rest variables. We also use nonlinear *log* transform for two variables: *num_new_dev* and *num_involved_dev*, containing extreme values and high variance. We use the Variance Inflation Factor (VIF) to check whether multicollinearity exists between the independent variables in the regression models, which can lead to regression coefficients that are difficult to interpret. Typically, if the VIF values are smaller than 5 then multicollinearity is not significant. This is the case with all our models. To describe the goodness of fit of our models, we use the *squared GLMM()* function in *R* to report two pseudo-R^2 values: the marginal R^2, interpreted as the variance solely described by the fixed effects, and the conditional R^2, interpreted as the variance introduced by both fixed and random effects in the model [81]. We also report the standard goodness of fit measures of log-likelihood and the Bayesian information criterion, the latter often used for model choice.



(a) Number of Devs /w Exposure (b) Number of Involved Devs (c) Number of New Devs (d) Number of Commits (e) Number of Comments

Figure 2.2. The monthly aggregated numbers of five variables (x-axis unit is in month), relative to the adoption month ($x = 0$), over all projects. Error lines show the ± 1 standard error away from the mean.

2.6 Results and Discussion

2.6.1 RQ1: Tool Adoption and Team Dynamics

First, to study tool adoption events over time, we align all projects around their adoption dates and plot those adoptions for each tool. The results, per tool category and per tool are shown in Figure 2.1. The figures suggest that adoptions in OSS projects spread in a non-constant speed, some group of people adopt a tool much sooner than the average adoption time, while others adopt it only if they are fully convinced, i.e., have an adoption lag. This fits well within the predictions of the DOI theory. Note that the tool category *Dependence manager* is not included because of lacking enough data points.

Second, to study the team dynamics around adoption events, we examine the temporal data of our five team-level metrics: the number of developers with prior exposure, new developers, involved developers, comments associated with tools, and commits at monthly intervals, as illustrated in Figure 2.2.

We see from Figure 2.2 that the average number of involved developers is almost linearly increasing over time, likely correlating with the general GitHub trend. This implies that more developers participate in the discussions of adopting tools.

We also observe a significant discontinuity in the steady numbers of commits and comments just before and even more so after the adoption event, in the positive direction. This is arguably associated with increased activities related to the adoption. Also notable, is that the number of developers with prior exposure to the tool is steadily growing in the period before the adoption, thereby likely increasing the adoption chances.

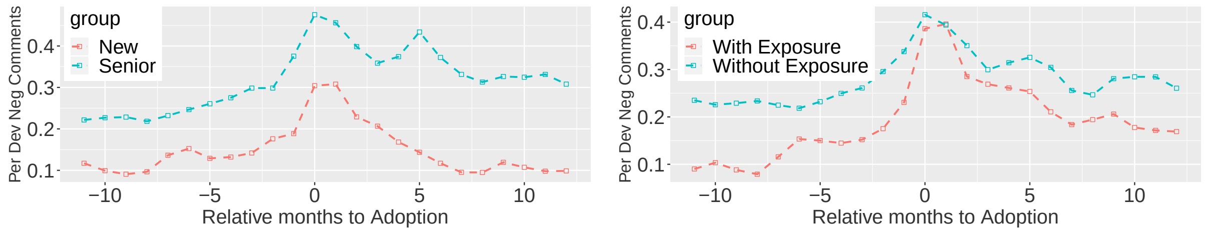


Figure 2.3. The per-developer negative comments posted by new developers and senior developers (left), and developers with prior tool exposure and developers without prior tool exposure (right). Adoption event happens at month x = 0.

Table 2.3. Topics Discovered in Discussions

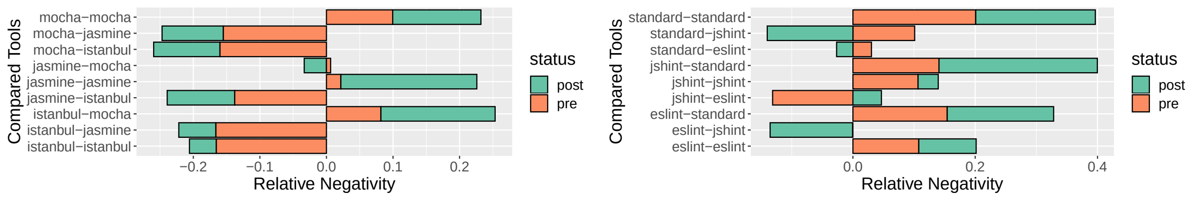
	Topic	**Sample vocabulary**
1	Testing	run, test, use, report, case, mocha
2	Development	js, setup, window, browser, npm
3	Debugging	fail, stack, error, timeout, check
4	General ideas	work, support, dependency, need
5	Integration	CI, function, module, nodejs, client

Answer to **RQ**[1]: Successful adoption distributions are in line with DOI theory, with some projects adopting early and others late. We observe that new developer numbers increase slower than involved developer numbers but significantly more so after the tool adoption.

2.6.2 RQ[2]: Discussion Topics and Sentiment

Pre- and Post-adoption Discussions LDA is regularly used to reveal the frequent topics in text information. However, different from the corpus gathered from social media, the comments from GitHub, by their nature, are in a much narrow domain. Therefore, the number of different discussion topics on GitHub is much fewer compared to social media corpus. Even though, many topics still overlap with each other. We uncover the top-ranked topic features and their associated sample vocabulary in the tool discussions, and summarized them in Table 2.3.

To understand and identify topics and emerging patterns in the discussions, we go



(a) Testing Tools (b) Linter Tools

Figure 2.4. Negativity toward tools in the same category, illustrated both before the adoption (red) and after adoption (green).

through 3,342 discussion comments within the 6 months prior to the adoption date. By using the topic-vocabulary pairs as the keywords, we find the following three topics in tool discussions are prominent:

Perceiving Demands of Tools: ”right now the ¡file¿ is extremely distracting and the test output is impossible to read (just the result). We need a ¡tool¿ PR to solve this.”

Choosing One Tool Over Another: ”¡dev 1¿ thanks. I should have done this, to begin with. I set up ¡tool 1¿ because that’s what ¡tool 2¿ was using before. But you are correct, that is better for testing.”

Deciding When to Adopt: ”Thank you for sending us these contributions! Moving to ¡tool¿ is, in fact, something we have hoped to do, I just wanted to let you know it might take us a few more days before we’re ready to engage.”

We also analyze 4,278 discussions from the same projects set for the 6-month post-adoption period. The two identified prominent topics of discussions are about:

Adoption Feedback: ”Not sure how I to write a test for this. I can’t figure out how to get the output from ¡tool¿”

Switching Tools: ”But when comparing/choosing a testing framework you will definitely need to say one is better for you or a project at some point. I would be open to using ¡tool¿.”

In summary, we find the following prominent patterns exist in the tool-related discussions: a discussion seems to be initiated by an individual who found an issue and asked for addressing such issue with a tool, and it ends with an individual who has previously used similar tools presenting their experience, by them recommending a tool to adopt. We also find that discussion threads are goal-oriented, well structured, and proceed logically, and they are likely to be beneficial for developers to decide on which tool to adopt.

Developer Sentiment To test our hypotheses from SJT about ego involvement and from DOI about tool adoption, here we compare the sentiment in tool adoption discussions between the comments of a) new developers and senior (i.e., not new) developers, and b) between developers with prior tool exposure and the ones without prior tool exposure. To do that, we compare the negative comments posted separately by both new developers and senior developers. As shown in Figure 2.3(a), before the adoption, new developers are much less negative to adopting new tools than senior developers, even though their negativity significantly grows just before and more so after the adoption events, if only for a short time. This is consistent with SJT: new developers have less involvement in the projects and thus lower emotional attachment than senior developers, while the latter have to (perhaps begrudgingly) adapt to the changes in the project. Even more interestingly, the negative sentiment of the senior developers persists, which can in the longer term affect project cohesion and effective management.

On the other hand, as shown in Figure 2.3(b), the per-developer negativity of developers with prior exposure is much lower than the developers without exposure, for all time-bins. And after implementing tool adoptions, the negativity of developers with prior exposure is very close to developers without exposure, however, the negativity of developers with prior exposure drops faster right after the adoption and remains lower in the long term. This validates the assumption that developers with prior exposure are less negative toward the tools than the ones without exposure. Moreover, we find that the curves of the two types of developers are similar to some degree, suggesting both are reacting to the same events, or are communicating together.

Relative sentiment. To understand why developers choose one tool t_1 over another tool t_2 from the same category (e.g., both t_1 and t_2 are linters), we compare the relative negativity of their corresponding discussions. First, we calculate the baseline negativity for each project (i.e., the ratio of negative comments over all comments). Then we calculate the aggregated negativity of a tool as the ratio of negative over all comments in the discussion, minus the project's baseline negativity. In Figure 2.4, each row represents a pair of two tools ($t1$-$t2$) from the same category. The bars show the aggregated (team-level) negativity to tool $t2$ of the projects before (red) and after (green) adopting tool $t1$. If tool $t1$ and $t2$ are the same (i.e., $t1 = t2$), the bar simply represents the change of negativity after adopting the tool $t1(i.e., t2)$. As shown in Figure 2.4(a), the bar *Jasmine-Jasmine* shows that, the post-negativity becomes more negative than pre-negativity, suggesting that developers find *Jasmine* more difficult to use than expected. In Figure 2.4(b), *standard-jshint* bar shows that after adopting tool *standard*, the sentiment toward *jshint* becomes less negative (i.e., more positive).

Answer to $\mathbf{RQ}_2$: We identify three most significant scenarios in the tool discussions: perception of tools, choosing a tool over another, and deciding when to adopt a tool. We also find that sentiment toward a tool is associated with developer seniority and prior exposure to the tool. Finally, sentiment toward a tool is associated with adoptions, and it can change after adoptions.

2.6.3 RQ$_3$: Influencers and Adoptions

In the sentiment data set, we find that 5% of developers posted 35% of total negative comments, indicating there might exist some strong ego-involvement and possibly influencers during the discussion period. Here, we seek to answer if influencers exist, identify them, and see how they affect tool adoptions. We first define *influencers* as those developers who post more than 50% of the total comments in the tool adoption discussions. To have reliable data to study developers who frequently comment in each discussion, we decided to only consider adoption discussions with more than ten comments. That filter leaves us 592 distinct adoption events to study. Among them, 379 adoptions are successful, and 213 adoptions are unsuccessful. We define the Successful Adoption Rate (SAR) as the ratio of the successful adoptions to the total, i.e., $379/592 = 64.02\%$.

A very common situation is that a project member wants to have a testing tool for their project, and the developers create an issue to query other's opinions on whether to adopt this tool. If the majority of main contributors agree on adopting a testing tool, then they would be discussing which tool to adopt. Our hypotheses, arising from the ego-involvement considerations in SJT, are that: 1) a project with an influencer has a higher likelihood to have adopted a tool, and 2) the higher the negativity of a strong influencer, the less likely it is for the tool to have been adopted.

We test the first hypothesis by comparing successful adoption rates in projects with strong influencers to those without. We find that for the former, the SAR is 68.86% and for the latter 57.75%.

To test the second hypothesis, among the adoptions that have had an influencer, we found that if the sentiment of the strong influencer was positive on the tool, the SAR is 72.18%, while if the sentiment of the strong influencer was negative, the SAR is 64.71%.

In contrast, and as predicted by SJT, without an influencer integrating the democratic opinions into a consensus decision may be more difficult and take longer, while the influencer can speed up the adoption process. We find that without an influencer, the average discussion length is about 15 months, while with an influencer, the length decreases to 13 months, on average.

Answer to $\mathbf{RQ}_3$: We identify strong influencers in the projects, and show that projects having strong influencers have more successful adoptions and shorter adoption discussions. Moreover, the sentiment of the strong influencer correlates with the adoptions.

2.6.3.1 Case Study

We give examples from two comparably sized projects *testem/testem* and *bower/bower* to illustrate how a strong influencer can be of help in tool adoptions. The project sizes are similar to each other (number of commits: 2,305 v.s. 2,726; number of contributors: 157 v.s. 209).

With a strong influencer In the project *testem/testem*, the following discussion transpired, on using a tool to automate testing, where [Dev 2] is the strong influencer in the team.

[Dev 1]: ”... I think it would be convenient to render the page as template and pass there...”

[Dev 2] ”Why do you need this? Please give more details about your use case.”

[Dev 1] ”... I use testem not just to run unit tests to see standard test report page, but as a watching tool that automatically reloads my web application when sources changed ...”

[Dev 2] ”I get most of what you are saying. Are you using [tool 1]? The hash issue I think I need to rethink how to handle that ...”

[Dev 1] ”Now I switched to [tool 2] (as it really more robust and convinient), I used [tool 1] as well it doesn't actually matter. And I use dependency management tool to load scripts ...”

Without a strong influencer In contrast, the project *bower/bower* encountered an issue when trying to use a tool for automated testing. A developer asked for help from the community, however, no one presented strong opinions in favor of continued use of this tool. One member suggested to not use ¡tool¿ anymore and switch to another tool.

[Dev 1]: ”Should we have a common way to declare the tests for any component? For example, I'd specify [tool] and file in [file]. I'd then run [command], which would open the [tool] page in a browser. Thoughts?”

[Dev 2]: ”what we could support is something similar to ¡package name¿ which is common scripts specified in the [file] ... what do [Dev 3] and [Dev 4] think?”

[Dev 4]: ”[Dev 5] has said it was a mistake to make them all globals. they should be triggered with [file] ...”

[Dev 6]: ”... Any component with unit tests that I've written just ends up using a separate node module (like *[tool]*) to automate the test workflow. I'd be in favor of closing this.”

The two decisions are in contrast. In the first project, tool adoption happened after a strong influencer insists on it. In the second, multiple project members are involved, and the project adopted a different tool than the one discussed.

2.6.4 RQ4: Adoption & Discussion Determinants

In the previous RQs, we conducted exploratory and qualitative studies of team discussion and dynamics before and following an adoption event. Here we triangulate those with quantitative studies, to understand the determinants, as well as the direction of their effects, on adoption success and discussion length.

Our data is naturally hierarchically organized based on the tool being discussed. We use *tool* as a random effect in our models, allowing all projects adopting a specific tool to have the same random intercept. All other variables are used as fixed effects in our mixed effect models.

We model each of the two outcomes with a base model, comprised of the control variables, and a full model, which adds to the base the complement of team variables. We perform the likelihood ratio test between the base and full models using the $anova()$ function, and present both models for each outcome variable.

Modeling Adoption Success. The results are shown in Table 2.4. We see from the AIC that the full model fits the data significantly better than the base model, with the three significant team variables explaining about 5% of the total variance, as per the marginal R^2. Overall, the fixed effects alone do not present a good model, but together with the random effect, the model is much better, at 48% conditional R^2. This, together with the vif's being smaller than 5, gives us confidence that we can interpret the coefficients of the variables.

Of the controls, the variables *discussion_length*, *num_comments* and *num_commits* have significant, sizeable negative effect on tool adoptions, holding all else constant. This is consistent with the SJT prediction that group judgment needs more time to form in larger teams, and that an adoption may be more difficult to succeed since an agreement is needed from more people. *num_mentions*, on the other hand, shows a sizeable positive effect, which makes sense from a DOI perspective, that an adoption needs a wider spread to succeed.

Of the team variables, *num_involved_dev* is positively associated with adoption success, all else held constant. When multiple developers are highly involved in the project, they may all be on the same page concerning the project's needs. This is consistent with SJT, as aligned egos will easier agree. The predictor *num_w_tool_expos* also has a significant, sizable positive effects on adoption success. One possible reason for this is that

Table 2.4. *adoption_success* glmer model, *tool* as random effect.

	Base Model	Full Model
scale(discussion_length)	-0.477^{***} (0.110)	-0.484^{***} (0.129)
scale(project_age)	-0.051 (0.105)	-0.115 (0.108)
scale(num_mentions)	0.612^{***} (0.150)	0.448^{**} (0.214)
scale(num_comments)	-0.183^{*} (0.110)	-0.268^{**} (0.115)
scale(num_commits)	-0.208^{**} (0.097)	-0.222^{**} (0.108)
scale(log(num_new_dev + 0.1))		-0.464^{***} (0.163)
scale(num_w_tool_expos)		0.617^{***} (0.141)
scale(log(num_involved_dev + 0.1))		0.419^{**} (0.181)
scale(num_pos_dev)		-0.126 (0.141)
scale(num_neg_dev)		-0.120 (0.140)
Constant	1.264^{***} (0.410)	1.434^{***} (0.399)
Observations	1,085	1,085
Log Likelihood	-407.502	-391.491
Akaike Inf. Crit.	829.003	806.981
Bayesian Inf. Crit.	863.929	866.854
Marginal R^2	9.94%	14.79%
Conditional R^2	46.72%	47.97%

Note: *p<0.1; **p<0.05; ***p<0.01

the developers who had previously been active in projects that have used the tool, are more familiar with the tool. Consistent with DOI theory, those are the developers that contribute to the diffusion (spread) of information on the tool in their new projects, which can lead to successful adoptions. Refined temporal diffusion models can offer a more detailed, temporal view of this diffusion process, and are left for future work. An interesting finding is that *num_new_dev* is significant and negatively associated with adoptions. We can see several explanations. First, new developers may not feel comfortable to state their opinions publicly, which practically, may amount to a negative overall opinion. Second, as they do not understand the ins and outs of the projects yet, they may not perceive the need for the change, especially since they have just recently started contributing. Also, we find that neither *num_neg_dev* nor *num_pos_dev* have a significant effect on adoptions. We see this in the context of SJT: high ego developers are likely to be the ones participating in the discussions, and their arguments, emotional or not, are unlikely to change the opinions of other high ego developers.

Modeling Discussion Length. As we see in Table 2.5, the AIC tells us that the full model fits the data significantly better than the base model, with the three significant team variables explaining about 40% of the total variance, as per the marginal R^2. Overall, the fixed effects alone present a good model, but together with the random effect, the model is excellent, at 91% conditional R^2. This, together with the vif's being smaller than 5, gives us confidence that we can interpret the variables coefficients and trust the model.

Of the control variables only *project_age* has a significant, sizeable positive effect. It is in line with expectations: older active projects will have more participants and this likely longer discussions. In the team variables, *num_new_dev* is sizeable and positively associated with the discussion length, all else kept constant. Both DOI and SJT are consistent with these findings, as the new people, who have little ego involvement, can be the ones with questions or comments about tools and the project. The other team variables are negatively associated with the discussion length, but their effects are small. In particular, *num_involved_dev* and *num_w_tool_expos* are negatively associated with discussion length, in line with expectations that involved developers and those exposed to the tool previously may not need long discussions to decide. The *num_pos_dev* variable has a small, but a significant negative effect on discussion length, suggesting that more positive developers can be beneficial to shortening discussions.

Table 2.5. *discussion_length* glmer model, *tool* as random effect.

	Base Model	Full Model
scale(project_age)	0.360*** (0.010)	0.242*** (0.010)
scale(num_mentions)	0.038*** (0.009)	0.059*** (0.014)
scale(num_comments)	0.072*** (0.009)	−0.072*** (0.010)
scale(num_commits)	0.183*** (0.008)	−0.006 (0.009)
scale(log(num_new_dev + 0.1))		0.922*** (0.020)
scale(num_w_tool_expos)		−0.062*** (0.009)
scale(log(num_involved_dev + 0.1))		−0.038** (0.016)
scale(num_pos_dev)		−0.030** (0.013)
scale(num_neg_dev)		−0.013 (0.012)
Constant	1.838*** (0.155)	1.644*** (0.098)
Observations	1,085	1,085
Log Likelihood	−5,868.721	−4,451.806
Akaike Inf. Crit.	11,749.440	8,925.611
Bayesian Inf. Crit.	11,779.380	8,980.494
Marginal R^2	31.50%	77.99%
Conditional R^2	86.71%	91.33%
Note:		*p<0.1; **p<0.05; ***p<0.01

Answer to **RQ**$_4$: For adoption success, the positive significant variables are the number of mentions, developers with prior exposure, and involved developers. As for discussion length, the number of new developers seems to be the most significant indicator to extend the discussion, while the exposure factor has a positively sizable effect on shorting the discussion.

2.7 Takeaways for Practitioners

Here we distill from our findings some practical takeaways and suggestions. Generally, OSS project team discussions are helpful for community building. But they are sorely lacking during tool adoptions, and since we also found that the discussions tend to be goal-oriented and rational, we recommend that they should be encouraged in the OSS community.

Our finding that having more people on the project with prior exposure to a tool is associated with successful adoption is evidence toward proceeding with adoptions after a team has multiple members with prior exposure. However, longer discussions can be distracting to a team, and we found they are not associated with better adoption outcomes; on the other hand, having an influencer as a champion for the adoption may help.

We also found that some tools are associated with longer discussions than others, perhaps because they demand certain prior exposure and further research. Setting expectations for the team ahead of time can limit feelings of frustrations arising out of lengthy discussions. And while we found no association between negative (or positive) developers and adoption success, discussions tend to be shorter as the number of positive developers increases. Thus, being more positive than negative may help keep things shorter.

Further, more new developers are associated with lower adoption and longer discussions. While the number of new developers cannot be modulated much, perhaps timing tool adoptions during periods of low influx of new people may help the proposed tool adoption. Finally, more commits and comments are associated with lowered success of adoption, thus, planning tool adoptions away from busy project times may result in more successful adoptions.

2.8 Threats to Validity and Conclusion

Threats. Both adoption data and commits/comments data were gathered from GitHub, thus, generalizing the results beyond GitHub, or even beyond the gathered corpus, carries some risk. However, the projects were selected randomly (with some minimum activity requirements), thus lowering this risk. Also, we do not consider any offline/in-person communication channels between developers except through GitHub. Previous work has found that there exists a notable decrease in communications associated with same company affiliation, implying that developers may share their opinions offline. Also, Senti4SD is trained on communications among developers in Stack Overflow. GitHub comments can be different than Stack Overflow comments, though our small sample study here capped that to 12%.

Conclusion. Motivated by the availability of multiple tools per use category in DevOps settings, and the general lack of guidance about their appropriateness for specific projects, we studied team level determinants of tool adoptions and discussions.

In terms of the relative timing of tool adoptions, we found that there is a significant difference between the distribution of adoption times for tools in the same use category, making it challenging to choose which tool to adopt for projects that are laggards.

We considered tool adoption as a project-wide phenomenon, affecting every member in the group. But also depending on the opinion of many of them, including their prior impressions of tools and linguistic sentiment, we demonstrated that the involvement, tenure, and more importantly, prior exposure to the tool play significant roles in the discussion. We also find that strong influencers are associated with more, successful adoptions.

We find that the attitude towards adoptions varies across different groups of people, and that a team's relative negativity is tool-specific, and can change after adoption, suggesting that the usability of tools can be over- and underestimated. We conducted topic analysis, and in-depth case studies on how and why some similar projects choose one tool over another one. We conclude that tool adoption is akin to a reasonable team negotiation, that proceeds through multiple phases. We hope our results can be of help in future tool adoption decisions.

Chapter 3

Sustainability Forecasting for Apache Incubator Projects

3.1 Introduction

Open source has democratized software development. Developers flock to OSS projects hoping to add certain functionality, contribute to a worthy goal, and sharpen their skills. However, more than 80% of OSS projects become abandoned over time, especially the smaller and younger projects [82]. Certainly not all OSS projects are meant to be widely used or even to persist beyond a college semester. However, even the large, popular OSS software, widely used in our daily lives and by fortune 500 companies, started out as small projects. Thus, from a societal perspective, it is important to ask: Why do some nascent OSS projects succeed and become self-sustaining while others do not [83]? And can the latter be helped?

To aid developer communities to build and maintain sustainable OSS projects, non-profit organizations like the Apache Software Foundation (ASF) have been established. ASF runs the ASF Incubator (ASFI) [84], where nascent projects aiming to be a part of the ASF community are provided with stewardship and mentor-like guidance to help them eventually become self-sustaining and even top-level projects in ASF. Projects in ASFI, called *podlings*, are required to adhere to ASF rules and regulations, including keeping all commits and emails public. When certain conditions are satisfied, project developers and ASF committees decide if a podling should be *graduated*, referred to as a *successful* sustainability outcome. Otherwise they get *retired*. Per ASFI, *'A major criterion for graduation is to have developed an open and diverse meritocratic community. Graduation tests whether a project has learned enough and is responsible enough to sustain itself as such a community'*[1].

Podlings in ASFI receive a mentor, file monthly reports, and get feedback. In spite of this support, many podlings fail. Most ASFI committers do not lack coding expertise, but graduating from the incubator requires more: it asks for effective teamwork and sustainable, community development. These requirements are most challenging to meet. From comments in ASFI, we see that developers are confused by the expectation of 'The Apache Way', especially initially. E.g., from project *Flex* on 05 Jan 2012, '...I think there is a need to keep things as simple as possible for people who [are] already confused with what's going on with the move to Apache...' The frustrations sometimes persist beyond the initial period. A comment from project *Flex* on 06 Jan 2012, states '...many people are confused and lost as to what to do. Who is providing that direction for them?' In large part, understanding how to achieve the graduation requirements seems to be the culprit, likely due to the abstract nature of those concepts. Another reason is that comparison to others is difficult. From project *Rave* on 28 June 2011: '[sporadic adherence to requirements] makes it very confusing and difficult to compare against what other projects as some are doing too little while others are doing too much...'

Thus, there is a need to connect the proverbial dots on how to get from the point of entry into ASFI to checking all the graduation requirement boxes. That brings us to the motivation of our paper. The extrinsically labeled ASFI dataset offers heretofore unavailable, fine-grained records of historical trajectories of projects under policies and regulations of ASFI. We posit that:

> The process that ASFI projects follow toward becoming sustainable can be modeled as a function of a small set of project features, so that the outcome (graduation/retirement) can be predicted early on, from the successes and failures of others, allowing for trajectory adjustment if needed.

[1] `https://incubator.apache.org/guides/graduation.html`

To deliver on that, in addition to the historical, outcome-labeled ASFI data, we also need a theoretical framework that can capture the complexity of OSS development. Over the past 30 years research in organizational behavior and management has documented the evolution in project management practices [85], as they have moved toward more successful models [86]. The *socio-technical* view of an organization [87] has emerged as one of the more successful hybrid models recognizing the benefits that integrated treatment of the technical aspect (code, machines, device, etc.) and the social aspect (people, communication, well-being, etc.) has on an organization [88]. Likewise, OSS projects have been effectively studied from the *socio-technical system* (STS) perspective [89], with the social side capturing humans and their communication channels, and the technical capturing the content and structure of the software [90].

Here, inspired by the socio-technical systems modeling perspective, and the availability of extrinsically labeled historical data of project sustainability from ASFI, our goals are: (1) to identify socio-technical features distinguishing projects that graduate from the ASF Incubator from those that do not (i.e., find the determinants of OSS project sustainability), and (2) to build temporal forecasting models that can predict sustainability outcomes at any time point in the project development, and thus (3) to offer practical and timely advice on intervening to correct a project's course, especially early on. To approach these goals, we conduct a mix of quantitative and qualitative empirical studies. We start by gathering project technical traces (commits) and social traces (emails) from the ASF Incubator website [2]. From those, we construct the temporal social and technical networks for each project, and perform exploratory data analyses, deep-dive case studies, build an accurate forecasting model, and finally implement the interpretable model presenting timely advice. We illustrate our workflow in Figure 3.1.

Our contributions in this paper are:

- We provide a novel longitudinal dataset of hundreds of OSS projects' development traces under ASF regulation, with extrinsically labeled project sustainability status.

[2]`http://incubator.apache.org/projects/`

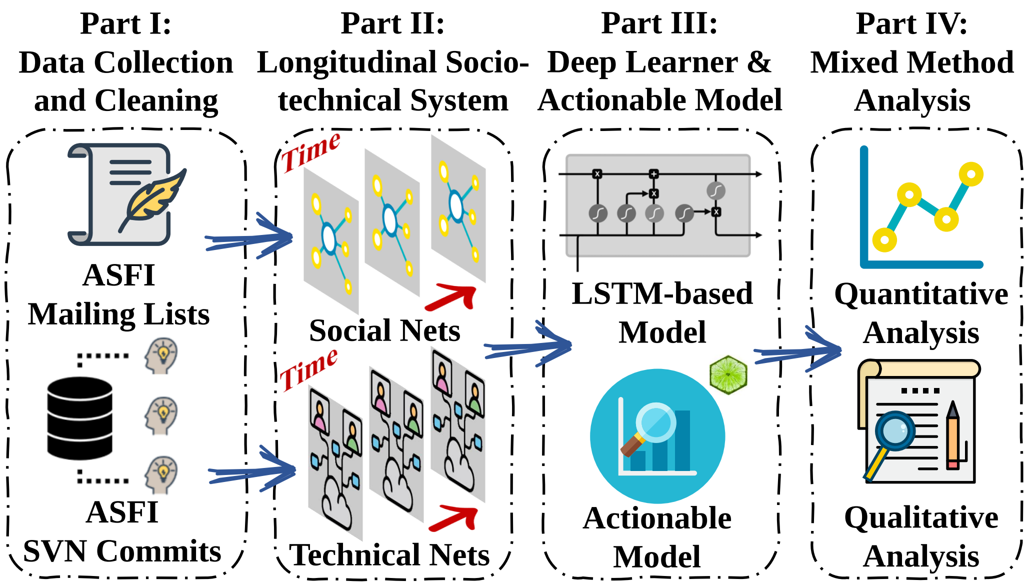


Figure 3.1. The workflow of our mixed-methods study.

- We propose the first OSS project sustainability forecast measure modeled from tens of socio-technical network and project features. Our model shows excellent predictive performance ($\geq 93\%$ accuracy as early as 8 months into incubation).
- We find that ASF incubator projects with fewer but more focused committers and more but distributed (participating in asynchronous discussions) communicators are more likely to gain momentum to self-sustainability.
- We describe a strategy for real-time monitoring of the sustainability forecast for any project, derived from an interpretable version of our DNN model.

This paper is a first step toward showing that end-results of OSS projects' sustainability trajectories can be effectively forecast and possibly corrected upward, if needed. Our motivation goes beyond ASFI as many more nascent projects fail outside of ASFI, so self-monitoring and self-adjustment may be more pertinent to them.

3.2 Background and Theories

We present background on the Apache Software Foundation (ASF) and OSS success, and then we introduce related theories through which we generate our Research Questions.

3.2.1 Apache Software Foundation Incubator

Community over code is the tenet of the Apache Software Foundation (ASF) community [10]. Their belief is that if they take good care of the community, good software will emerge from it.

However, conflicts are ubiquitous in OSS projects [91], and not even ASF can escape them. To minimize conflicts, ASF requires projects to make all communication publicly available on the mailing lists, summarized popularly as *if it did not happen in the mailing lists, it did not happen* [92]. The communication records benefit developers as they reflect on previous decisions and trace their precursors, therefore improving efficiency and productivity.

The ASF community adopts a democratic way in many of their affairs. For example, contributors are invited to vote $+1$ (yes), 0 (okay), or -1 (no) to project-wide changes. However, ASF committers can live in different time zones, and their response to a project decision can delay largely. Regarding that, ASF community adopts the *Lazy Consensus* [93]. Moreover, the ASF community also believes in *Earned Rights*, that newcomers should be treated the same way if they have proven their technical skills.

The goal of the ASF Incubator (ASFI) is to help projects become self-sustained and eventually join ASF. Like many OSS developers, ASFI committers work at will and there are no formal obligations on them. Thus, ASFI projects are required to show they are able to recruit new committers, and fill existing technical debt [94]. However, attracting new committers is difficult as they can be affected by both social and technical barriers [95]. To address this issue, the ASFI community has established a set of specific rules that emphasize providing mentorship to newcomers [96].

During incubation, ASFI projects need to adopt ASF procedures to develop and cultivate the projects' community, and standardize their working style. To graduate from the incubator and finally become a part of the ASF, projects are required to demonstrate they can self-govern and be self-sustained [97]. The specific requirement of sustainability can vary from one project to another [2, 98]. A project's graduation is ultimately approved for its self-sustainability by ASF's Project Management Committee via several rounds of public voting[3].

3.2.2 OSS Projects Success and Sustainability

Recently, substantial work has focused on modeling the success of OSS projects [99, 100, 33]. Even though there is no universally agreed definition of OSS success [101], there are two main perspectives. The first one is from the development process viewpoint, which is often measured by technical metrics of software [102], e.g., code defect density [103], response time [104], and error resolution rate [105]. The second one is more from the social angle, including contributor growth [106], community participation [107], and communication patterns [108]. In effect, K. Crowston et al. [109] studied the operationalizing success measures under the context of FLOSS projects. N. Cerpa et al. [110] provide survey evidence that factors in the logistic regression models are not as predictive as expected. D. Surian et al. [111] identify discriminative rich patterns from socio-technical networks to predict project success in the context of *SourceForge* projects. J. Coelho et al. [112] conduct mixed-method analysis on GitHub projects, and they find that most of modern OSS projects fail due to project characteristics and team issues. M. Valiev et al. [113] conducted studies on sustained activity within the PyPI ecosystem, and find that relative position in the dependency network has a significant impact on project sustainability. None of these consider forecasting over time and thus are not useful for real-time monitoring, which is the major contribution of this work.

Although OSS success and sustainability measure similar aspects of projects [114], they are, in fact, not the same thing. There are two main differences between the two. First, OSS success is measured statically while sustainability is measured dynamically. Second, sustainability is a measure related more to the human and social aspect (e.g., the ability to take responsible collective action, and an open and inclusive atmosphere, etc.) than the technical aspect (defect density, technical advantage, etc.).

Therefore, the sustainability of an OSS project becomes even more important when it is a part of a larger ecosystem [115, 113]. Such OSS projects are inter-dependent to each other, the sustainability and stability of one project can introduce tremendous network

[3] `https://incubator.apache.org/guides/graduation.html`

effect to its ecosystem.[4] Therefore, the sustainability of OSS projects becomes even more significant as they can influence many other OSS projects in the ecosystem that rely on them.

3.2.3 Socio-technical Systems Theory

Socio-technical structure plays an important role in achieving collective success in OSS projects [116, 19, 108]. A Socio-Technical System (STS) comprises two entities [117]: the social system where members continuously create and share knowledge via various types of individual interactions, and the technical system where the members utilize the technical hardware to accomplish certain collective tasks. The theory of STS is often referenced when studying how the technical system is able to provide efficient and reliable individual interactions [118], and how the social subsystem becomes contingent in the interactions and further affects the performance of the technical subsystem [119].

OSS projects have been studied from the network view [116]. González-Barahona et al. [120] proposed using technical networks, where nodes are the modules in the CVS repository and edges indicate two modules share common committers, to study the organization of ASF projects.

Moreover, in socio-technical systems, governance can be applied through long-term or short-term interventions. Smith et al. [121] proposed two conceptual approaches: 'Governance on the outside' objectifies the socio-technical and is managerial in approach. 'Governance on the inside' is more reflexive about the role of governance in co-constituting the socio-technical. From that perspective, the ASFI community is a unique system that has both outside influence (regulations from ASF committee) and inside governance (motivated by the project managers).

3.2.4 Contingency Theory

Contingency theory is the notion that there is no one best way to govern an organization. Instead, each decision in an organization must depend on its internal structure, contingent upon the external context (e.g., stakeholder [122], risk [123], schedule [124], etc.). Joslin

[4]A developer abruptly deleting the widely used, 11-line of *left-pad* code, led to cascading disruption of other OSS projects in *npm* ecosystem.

et al. [125] find that project success associates with the methodologies (e.g., process, tools, methods, etc.) adopted by the project. And not a single organizational structure is equally effective in all cases. As the organizational context changes over time, to maintain consistency, the project must adapt to its context accordingly. Otherwise, conflicts and inefficiency occur [126], i.e., *one size does not fit all.*

To address the conflicts caused by incompatible fitting to the project's context, previous work suggests thinking holistically. Lehtonen et al. [127] consider the project environment as all measurable spatio-temporal factors when a project is initiated, processed, adjusted, and finally terminated, suggesting that the same factor can have an opposite influence on the projects under a different context. In the domain of software engineering, Joslin et al. [125] considers project governance to be part of the project context, concluding that project governance can impact the use and effectiveness of project methodologies.

In the context of OSS projects seen as socio-technical systems, contingency theory implies that observing and tracking multiple facets/features of the projects may lead to more effective models of system evolution.

3.3 Hypotheses and Research Questions

Our goal is to build effective models for forecasting ASFI project sustainability. Here, we generate our hypotheses and formulate research questions based on prior work and the pertinent theories.

3.3.1 Hypotheses

STS theory suggests that publicly observable participation and decentralized contributions to software projects foster sustainable collaborations. The ASFI ecosystem is in a form of a typical STS where the technical activities build a shared code artifact and the social ones mediate knowledge and organizational details to individuals. Since all activities are logged, various socio-technical metrics can be calculated.

Our main hypothesis is that the STS formalism and the full availability of the projects' longitudinal digital traces, will make it possible to build an accurate model of sustainability. According to contingency theory, no single organizational structure is equally effective under all circumstances. Thus, across ASFI projects, we expect to see that the same socio-technical factors may have different contributions to sustainability. Finally, we posit, per contingency theory, that the roles of some social-technical factors may vary over time. Moreover, we expect to see that similar ASFI projects can end with divergent outcomes (graduation or retirement) over time based on actions they have undertaken.

3.3.2 Research Questions

We formalize the above into our RQs, as follows. The first is a validation of contingency theory hypotheses, that there exist measurable differences between graduated projects and retired projects, along with multiple features. Namely,

RQ$_1$ Are there significant differences among STS measures, between graduated projects and retired projects?

Next, STS theory holds that project sustainability is associated with social and technical network features. Contingency theory implies there will be multiple such features in play. Thus, we expect that a quantitative temporal model may be fitted well to the available ASFI data, so long as a sufficient number of features and projects are available.

RQ$_2$ How well can we predict the sustainability based on temporal traces of ASFI projects? And, can we identify the determinants along with their weights and directions?

To make the model useful in practice it needs more than just accurate predictions of outcomes, it also needs to generate timely advice on whether the project should stay the course or implement specific corrective action to improve the graduation trajectory. We formulate this as:

RQ$_3$ Can we monitor project sustainability status in a continuous manner? When and how should projects react to the monitoring advice?

3.4 Data and Methods

We collected historical trace data of commits, emails, incubation length, sponsor information, and incubation outcome for 263 ASFI projects, which have available archives of both commits and emails from 03/29/2003 to 10/31/2019 [128]. Among them, 176 projects have already graduated, 46 have retired, and 41 are still in incubation. The latter, projects still in incubation, were not studied in this paper.

We collected the ASFI data from two sources: ASF mailing lists and SVN commits. The mailing list archives are open access and can be accessed through the archive web page, `http://mail-archives.apache.org/mod_mbox/`. They contain all emails and commits from the project's ASF entry date, and are current. We constructed URLs for individual project files in the ASF incubator as *Project URL*. The project URLs use the pattern: `project-name/(YYYYMM).mbox`. For example, for project *hama*, the full URL is `http://mail-archives.apache.org/mod_mbox/hama-dev/201904.mbox`. Each such file contains a month of mailing list messages from the project, for the date specified in the URL. Here *dev* stands for 'emails among developers'.

However, we find that many projects, especially these over ten years old, which used SVN, used a bot in the *dev* mailing list to record all commits, thus a message from *dev* is not always an email. Similar emails were sent to the 'commits' mailing list, which, thus, contains some emails. Therefore, we collected both *dev* and *commits* mailing lists files for the 222 graduated or retired ASF Incubator projects through the archive web page[5].

3.4.1 Data Pre-processing

ASF manages and records the communications among people by globally assigning an exclusive email name to each developer at the project-level. However, some developers still prefer to use their personal email/name instead of the assigned one, which in turn complicates the identification of distinct developers [129]. We performed de-aliasing for those developers with multiple aliases and/or email addresses, as follows. We first remove titles (e.g., jr.) and common words in the name (e.g., admin, lists, group) from usernames, then we match with both the original order and switched first/last name order whenever names contain exactly one comma to eliminate ambiguous styles. Then we match each developer with her/his multiple email addresses (if any).

Many projects, especially those over ten years old that used SVN, utilized a bot for extensive mailings (empirical evidence shows 26% of popular GitHub projects use bots) [130], thus forming outliers in the dataset. Some broadcast emails are automatically

[5]Data and scripts are available: `https://doi.org/10.5281/zenodo.4564072`

generated by the issue tracking tool (e.g., JIRA), and no developer would reply to them. We eliminated the broadcast messages that no one replied to, and we find many of them are generated by automated tools. We find some developers contributed many commits by directly changing/uploading massive non-source code files (e.g., data, configuration, and image files). Since those can form outliers in the dataset, commits to files with extension data: .json, .xml, .yml, .yaml, .jar; text/configurations: .config, .info, .ini, .txt, .md, and image: .jpg, .gif, .pdf, .png are eliminated.

As result, we identify 21,328 unique contributors (who either committed code or posted emails). Among them, 1,469 only committed code, and 18,205 only posted/replied to discussions without committing code. The remaining 1,654 contributors engaged in both activities. We identify 2,764,309 commits, modifying a total of 404,455 source code files. We collect 879,812 emails, from them we identify 19,859 developers who participated in discussions (by sending or receiving emails). Among them, 19,573 proactively engaged in discussion activities (i.e., sending emails), the remaining 286 developers collaborated in a passive way (only received emails).

3.4.2 STS and Socio-technical Networks

We use socio-technical networks to anchor our study of OSS STS. Network science approaches have been prominent in studying complex dynamics of OSS projects [131, 111], although the specific definition may vary with domain context [16].

In this paper, we define the projects' socio-technical structure using social (email-based) and technical (code-based) networks, induced from their emails to the mailing lists and commits to source files, as follows. Similar to the approach by Bird et al. [19], we form a social (email) network for each project, at each month, from the communications between developers: a directed edge from developer A to B exists if B has replied to A's post in a thread or if A has emailed B directly (which is contained in the "in-reply-to" field). The technical (code) collaboration networks are formed for each project, at each month, by including an undirected edge between developer A and developer B if both developer A and B has committed to the same coding source file(s) F that month (excluding the SVN branch names).

3.4.3 Features/Metrics of Interest

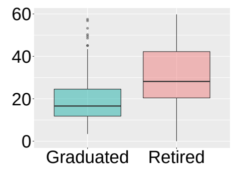
The socio-technical and project features/variables that we chose for this study have been identified based on our discussion and consideration of the underlying theories. All our data is longitudinal. All metrics are aggregated over monthly intervals, for each project, from the start to the end of its incubation [132]. We started with 29 variables, given their statistics as Supplementary Material.

Variable Selection We used Lasso regression [133] (L1 regularization) to identify a smaller set of 18 linearly independent variables, plus the outcome, described in the following. We used *R*'s library *glmnet* [134] for the Lasso regression, with $\lambda = 0.001$.

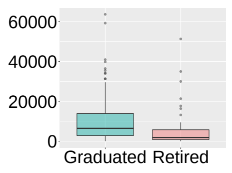
Outcome: Graduation Status. Graduation `grad_status` is a binary variable (0='Retired' or 1='Graduated') indicating the projects graduation status in the incubator, as discussed above.

Longitudinal Project Metrics: The number of Active Developers `num_act_devs` is the count of contributors who have been active by either making commits or participating in discussions. Number of commits `num_commits` is the count of source code commits made by all committers in the project. The process of excluding the commits that do not contain source code is described in the Data Section. The number of Emails `num_emails` is the number of emails (including both thread starter emails and reply-to emails). `num_files` is the total number of unique source code files created during the incubation. To measure the continuity of activities, we define `c_interruption` and `e` as the sum of the time intervals of the top 3 longest interruptions between successive commits and successive emails, respectively. The commit percentage `top_c_fract` and email percentage `top_e_fract` are the percentages of respective activities performed by the top 10% contributors.

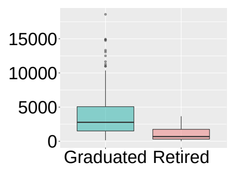
Longitudinal Socio-Technical Project Metrics: For each project network, for each month, we constructed the technical and social networks, and from them calculate the number of active nodes, `c_nodes`, and edges `c_edges` in the technical network; `e_nodes` and `e_edges` in the social network. The prefix `c_` in a variable's name indicates it is of the technical (code) network, while the prefix `e_` in a variable's name indicates it is of the social (email) network. Additionally, we calculated the mean degrees `c_mean_degree`



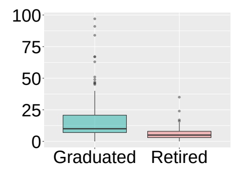
(a) Incubation Months ($p < .001$)



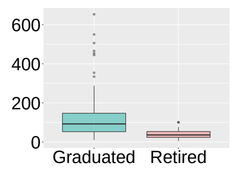
(b) Num. of Commits ($p < .004$)



(c) Num. of Emails ($p < .001$)



(d) TN Nodes ($p < .001$)



(e) SN Nodes ($p < .001$)

Figure 3.2. The descriptive variables between graduated projects (in green, left) and retired projects (in red, right). The corresponding p-value of the Student's t-test is in the brackets, suggesting significant statistical differences exist between them.

and `e_mean_degree` (sum of all nodes' degree divided by the number of nodes) in the technical network and social network, respectively. We calculate the clustering coefficients `c_c_coef`, `e_c_coef` as the number of connected triplets divided by the number of all triplets in the corresponding monthly network. The long-tail-edness `c_long_tail`, `e_long_tail` is calculated as the degree of the $75th$ percentile of nodes in the network, for the monthly networks, in the technical and social network, respectively. To get a sense of the range and variability in these variables, we show them aggregated over all months and projects in Table 3.1.

3.4.4 Models

We needed a modeling approach able to learn and forecast from longitudinal data, have excellent performance, and be interpretable.

3.4.4.1 LSTM-Based Learning Model

Long short-term memory [135] is a variant of Recurrent Neural Networks (RNNs), designed to learn and model sequential data and is less sensitive to the problem of gradient disappearance and gradient explosion when training on long sequences. To obtain sequence data for each project, we aggregated historical ASFI records into monthly data, for each month from the incubation start date to the project graduation/retirement from the incubator. We interpret the monthly LSTM output probability as the *graduation forecast*, i.e., the probability of the project eventually graduating.

We prepared the data as follows. We randomly divided the projects into training and test sets in an 8-to-2 ratio. Because we have variables that are of very different

Table 3.1. Statistics of the 176 graduated and 46 retired projects in the ASFI dataset. $c\_$ and $e\_$ correspond to technical networks and social networks, respectively.

Statistic	Mean	St. Dev.	5%	95%
`grad_status`	0.79	0.41	0	1
`num_files`	1,821.87	3,346.23	122.45	5436.6
`num_emails`	3,963.12	4,930.54	262.65	12463.6
`num_commits`	12,451.84	27,373.41	453.8	36359.7
`num_act_devs`	121.23	119.85	25	415.05
`c_interruption`	0.20	0.20	0.03	0.60
`e_interruption`	0.11	0.14	0.01	0.32
`top_c_fract`	0.65	0.19	0.38	0.94
`top_e_fract`	0.71	0.11	0.49	0.85
`c_nodes`	15.44	17.09	2	49.9
`c_edges`	120.15	276.19	1	531.5
`c_c_coef`	0.78	0.25	0	1
`c_long_tail`	10.65	11.93	0	33.85
`c_mean_degree`	8.14	7.45	1	23.75
`e_nodes`	113.38	115.07	22	408.15
`e_edges`	399.23	562.38	47.1	1315.9
`e_c_coef`	0.43	0.10	0.28	0.58
`e_long_tail`	10.33	7.15	3	24.95
`e_mean_degree`	6.07	1.91	3.88	9.62

magnitudes, and many of those are not normally distributed, thus we choose to use the *MinMaxScaler* function to standardize all prediction variables.

We implemented a 3-layer LSTM model: a 64 neurons LSTM layer, followed by a 0.3 rate drop-out layer, and a dense layer with the *softmax* function to yield the predicted outcome of the classification task (graduate/retire). If the probability of graduating is higher than 0.5 then we predict the project will graduate, otherwise, we predict it will retire. We validated the cutoff choice by examining the distribution of sustainability forecasts. During training, we used a binary cross-entropy as the loss function and *Adam* as the optimizer. Since the length of the temporal data of each project varies, instead of using zero-padding, which could possibly introduce variance to the model, we chose to use a slower but more reliable way by only feeding one training sample at a time. We use the accuracy, precision, recall, and F_1-measure to evaluate the performance of the LSTM model using the *classification_report* function from the *sklearn* package.

To get the *graduation forecast* for a project at month $m+1$, we cap the project history at month m, i.e., we only use the first m months in the model. We interpret the outcome yield of the LSTM model as the *graduation forecast*. Projects are not being calculated and used in the prediction when the current time exceeds their incubation lengths. We generated graduation forecasts for each month, and thus obtained the *graduation forecast trajectories*. Repeating the above process 10 times, selecting different training/test split each time, produced our error bounds.

3.4.4.2 LIME-Based Interpretable Model

Black-box deep learning models, like LSTMs, are less ideal for decision making than interpretable approximations of deep learning models. One such approach is the Local Interpretable Model-agnostic Explanations (LIME) [136]. Given a pre-trained model and an input instance, LIME reasons how the black-box model yielded the output, by probing the model along each of the features. LIME yields a magnitude and a sign (positive or negative) that characterize the contribution of each feature toward explaining the outcome.

Assumption LIME assumes that any complex model is linear (i.e., interpretable) at a local scale. Therefore, given an input instance, LIME first artificially generates large enough samples that are presumably very close to the given sample (by some distance measure). Then, by training on the predictions of those newly generated samples given by the complex model, LIME can locally approximate the complex model using linear models, thereby presenting the coefficients of variables of the input instance.

Procedure We first constructed a LIME explainer using the *RecurrentTabularExplainer* function from the Python LIME package (*version 0.2.0*). That package was designed for explaining RNN-type models with tabular data. For each project, we use *explain_instance* function, setting the parameter *num_features* to the product of the incubation length and the number of features. In this way, we can obtain the coefficients of all features over all time. The parameter *num_samples* is set to 5,000 (by default), which is empirically sufficient for convergent results. Next, LIME probes the pre-trained LSTM model 5,000 times by feeding it the newly generated samples. LIME uses a similarity/distance function to measure the importance of each new sample on the locality of the instance to be explained. Lastly, LIME fits a weighted linear model dataset, and the explanations all come from the final linear model. Since the LIME framework requires all samples to have the same shape, we divided our projects into several buckets where the projects have at least n months of temporal data in the n-th bucket.

Project-specific vs Overall Modeling We used the LIME results in two modeling ways: *project-specific* level and *overall* level. In the former, we used LIME to obtain the monthly coefficient of each variable and then aggregated them over all months to obtain a project-specific coefficient. In the latter, we aggregated project-specific coefficients over all ASFI projects to obtain the coefficients for each variable over all projects.

3.5 Results and Discussions

3.5.1 RQ1 Graduated vs. Retired Projects

To perform exploratory data analysis, we first contrast ASFI graduated and retired projects along the technical (code-based) and social (email-based) dimensions in our data.

Box-plot comparisons of the incubation months, number of commits, number of emails, number of nodes in the social networks, and number of nodes in the technical networks are shown in Figure 3.2. We observe notable differences as follows. The median incubation months of retired projects is significantly higher than of the graduated projects, suggesting that retirement is not an easy decision, and that perhaps necessary time is given to projects to change their trajectories and achieve graduation.

Graduated projects also tend to have more code commits and more email communications, implying that, in terms of criteria for graduation, the ASF community values both technical contribution (as commits) and social communication (as emails), and both of them may be of importance in building a sustainable community. Such results also motivate us to expand our research goals from descriptive data to inferential data with more complex network features.

Across both the social and technical networks, the network size varies for the graduated projects, indicating that projects of any size can be sustainable, and it also suggests that project size can be used as a control in modeling. The notable difference between graduated and retired projects, and the lack of variance in network sizes in the retired projects suggests recruitment difficulties in the latter, exposing them to significant risks as people leave.

Answer to RQ$_1$: We observe significant differences across multiple key measures between graduated projects and retired projects. Notably, retired projects tend to stay longer in the incubator than graduated ones, and the productivity and diversity of graduated projects are higher compared to retired projects.

3.5.2 RQ$_2$ Interpretable Forecasting

Here we present the results of training an LSTM model on our ASFI data, and an LSTM-derived LIME model on the same data, using the methods as described above. We use those models for forecasting by each month into the project the eventual graduation outcome. First, we show the performance curves of the LSTM model, over time, in Figure 3.3. The overall accuracy, and F1-value, and their standard errors (grey area), for the full model and the model without the socio-technical variables, suggest an excellent and stable predictive performance of the trained LSTM model, with significant contribution from the socio-technical networks. As early as month 8 the accuracy is 93%, and staying above after that.

The total incubation time varies significantly across ASFI projects. While for most projects the incubation time is between 8 months and 25 months, some spend more than 30 months in incubation, while others only 6 months, as shown in the inset plot in Figure 3.3. Thus, the model's performance decreases for projects with below 8 and above 25 incubation months, due to insufficient data above and below those values.

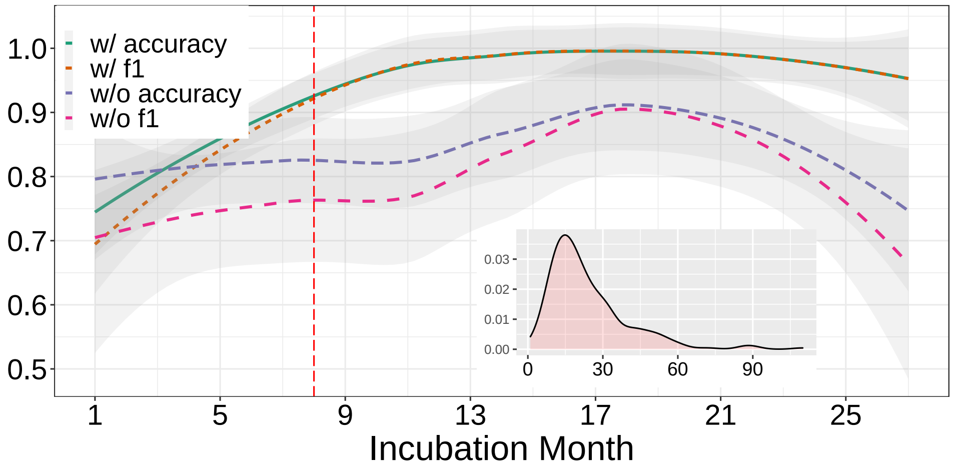


Figure 3.3. Performance metrics of the full LSTM model across incubation months (top 2 curves), showing the significant contribution of the socio-technical metrics. Curves are plotted using *loess*. Grey area shows the standard errors. The red vertical line shows 93% accuracy at 8 months of incubation. The inset shows project density over total incubation time.

Next, we apply LIME to understand and interpret the LSTM model and derive regression-like coefficients for the features in the socio-technical networks. To illustrate how to interpret the results at a project-specific level, we show an example of LIME's output for a graduated project, 'Empire-DB', in Figure 3.4. Note that the coefficients are aggregated over all incubation months. Looking at the median over all months provides model coefficient stability and avoids emphasizing very small effects which nevertheless dominate in some months. The magnitudes of the coefficients tend to be small because the model predicts probabilities within [0, 1], and there are tens of them.

There, we see that the technical network clustering coefficient `c_c_coef` is negatively associated with successful graduation. A high clustering coefficient in the technical network of people and source files indicates a high overlap of developers' activities on the same files. One possible reason for the negative effect introduced by `c_c_coef` is that, work may not be well distributed among the team members. Another reason might be that the artifact is not well-conceived. Yet a third reason can be that the number of developers on the project is small and they must all "tend to fires" wherever they might be. Interestingly, the fraction of commits (top_c_fract) and emails (top_e_fract) by the top 10% developers who tend to be the influencers in projects are positively associated with

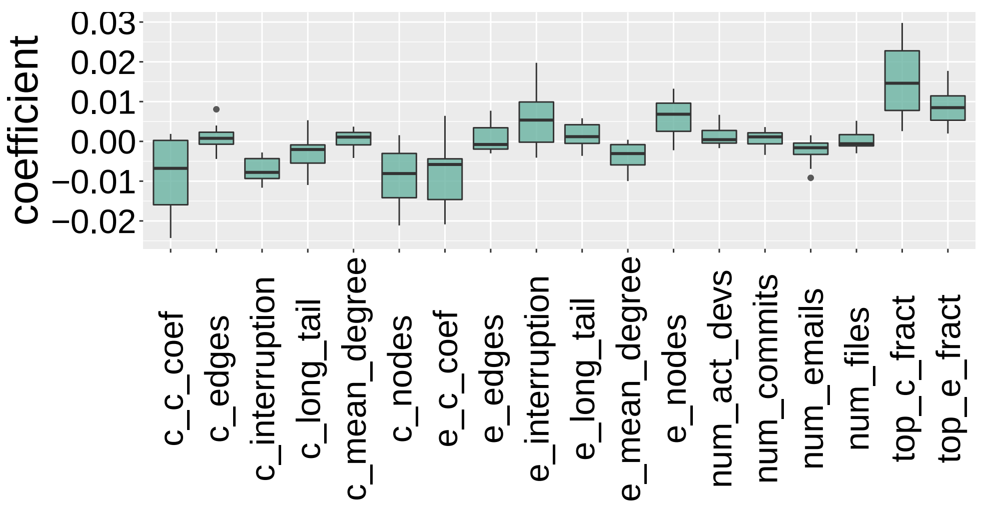


Figure 3.4. The coefficients of all variables from a graduated project ('Empire-DB'), aggregated over all incubation months, showing that LIME delivers stable estimation at the project-level.

graduation, implying that the efforts of the top 10% help sustain the project. Figure 3.4 also shows the feature coefficients of a selected project in both size and direction across all incubation months, which provides further insight, and confidence in the methods utility.

Next, by only counting the signs of the project-level coefficients across all projects, we can identify whether there is an overall consistent direction in which that feature is contributing to the prediction. E.g., if a feature is consistently negative to the outcome across all projects, i.e., that feature's coefficient is negative in most project models, then it has the same, overall negative effect.

In Figure 3.5, we show the count of aggregate signs of feature coefficients across all projects. There, *blue* indicates a positive effect and *red* a negative one. Visually, if most of the bar is a single color, then that variable has a consistent effect direction, i.e., same sign coefficient, among all projects. Overall, perhaps surprisingly, we find that for almost all projects, the number of nodes in the technical networks `c_nodes` has a negative effect on graduation for minor projects while the number of nodes in the social networks `e_nodes` is positively associated with graduation. This is consistent with prior research findings that communication is more determinant of success and onboarding than coding activities [137]. Other variable appear to have inconsistent effect across projects, e.g., `num_files` and `c_mean_degree`.

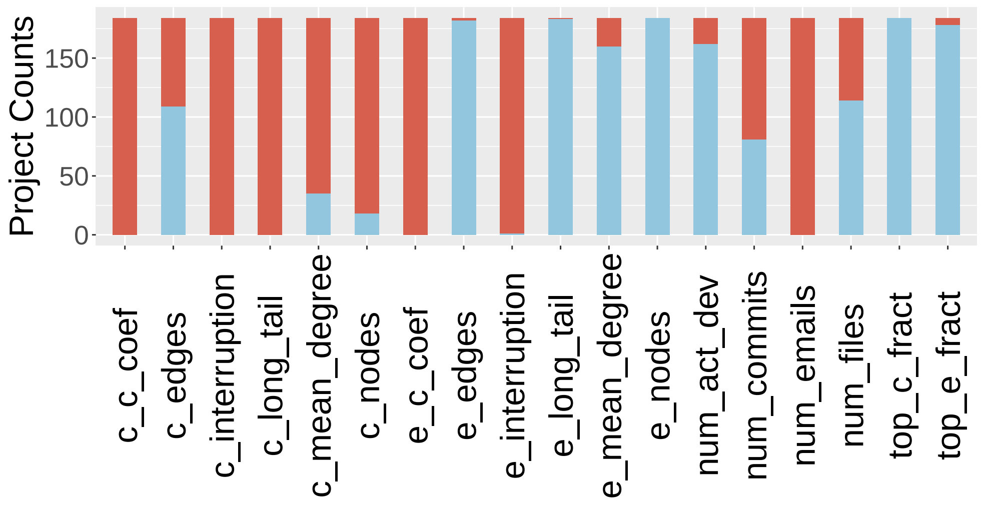


Figure 3.5. The overall-level coefficients of all variables of interest (blue is positive, while red is negative to graduation). It shows that some variables have same effect on almost all projects, while others do not.

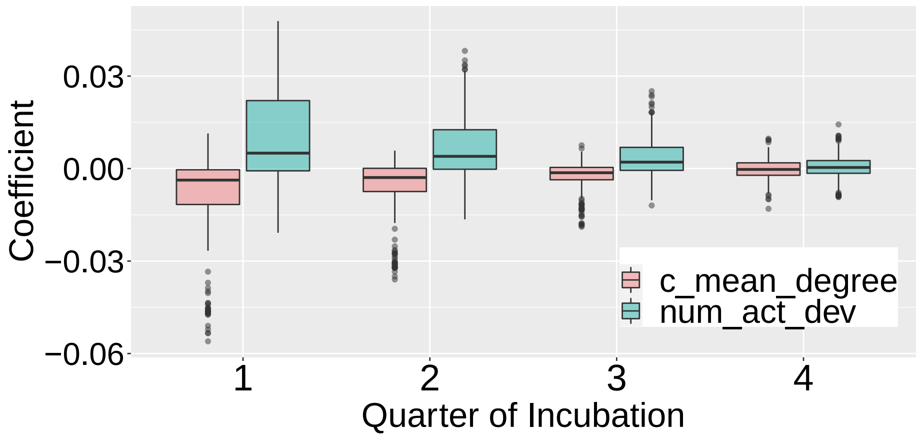


Figure 3.6. The overall-level coefficient of two selected variables: mean degree in technical network (in red) and the number of active developers (in green) in different incubating quarters.

Since LIME fits model coefficients for all months, we can also examine the dynamics of feature coefficients. Figure 3.6 shows that when broken down into 4 intervals, the effect of *num_act_devs* becomes less positive, and less important, over time, perhaps due to the project becoming more stable. Contrariwise, the negative effect of the mean degree in technical networks (*c_mean_degree*), diminishes in latter development, arguably, again, because of increased project stability over time.

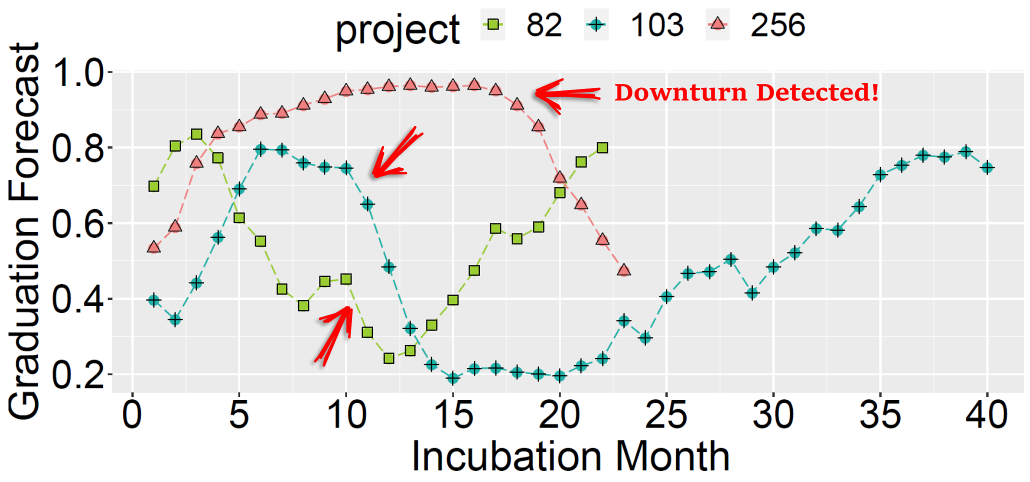


Figure 3.7. The graduation forecast of the marginal projects. *Commonsrdf* (ID: 82, in green) and *Etch* (ID: 103, in blue) are graduated projects that almost failed while retired project *Ariatosca* (ID: 256, in red) almost succeeded.

Answer to RQ$_2$: Effective models of project sustainability can be built from tens of socio-technical and project features. Stable and interpretable models can be derived yielding feature coefficients at both project-specific and overall levels. Notably, overall, projects with fewer but more centralized committers and those with more but distributed communicators are more likely to become self-sustainable in the ASF incubator.

3.5.3 Case Study: Change of Fate

To understand in depth why trajectories may change, with the help of our interpretable model we randomly selected three qualitatively different example projects (2 graduated and 1 retired), showing up- or down-turn points in their sustainability trajectories, as illustrated in Figure 3.7.

§1 In project *Commonsrdf* (ID: 82), the graduation forecast starts high but experiences a downturn in the first half of incubation, and then it rises again. Our model identifies the lack of both email and commit activities are associated with the downturn. There is also an overall decreasing trend in the number of active developers (from 27 to 11, then to 4 eventually). To investigate why their promising trajectory changed, and then changed again, we looked through their discussions on the mailing list.

In the month with the lowest graduation forecast, the project released a major version of their software, following which the committers showed less activity. We also noticed that the ASF incubator Project Management Committee (PMC) routinely sent an email to the project asking for the periodically report [6], but such reports were not requested for more than 2 months after the release. Lastly, we saw email arguments on the technical direction for the project's future development. Unsurprisingly the project was going to fail if no one observes such situation and takes action to intervene. However, an email from one of the major contributors, Dev_1, appears to have initiated the turning point. Below we quote it and the ensuing discussion.

Dev_1 "*Folks, I've seen very little traffic for the last few months... I am concerned that there is perhaps no longer a viable community around this podling...*", and seriously asked "*Do people still think this project has/can build the momentum to move forwards towards graduation?*"

Dev_2 "*... we lost one of the main pillars of this projects ... So our mentors are right. We're in a situation where the project has no momentum at all, and honestly I have no idea what's best to do...*"

Dev_3 "*... there was a small group of 5 core committers to begin with. As of right now, the number is* 22. *We've actually done pretty well...*"

Then, Dev_4 responded with an email titled 'Values and Terms' and suggested a technical directions, with detailed reasoning: "*... if you actually tried to use this (algorithm) would (i) hurt speed, and (ii) hurt the perception of speed...*" and clearly states that "*I'd be inclined to go another step further and add a generic parameter...*"

After this discussion, the community became more engaged and increased some activities, and the community felt more confident about the upcoming routine report. Eventually, the project graduated in the end (our forecast went up to 80%).

§2 In project *Etch* (ID: 103) there was a major depletion of senior developers after a milestone was reached in the middle of incubation. Then commits stopped for almost one year. Our graduation forecast reflects that: it dropped from 79% to only 19%.

After a long time being inactive, one developer sent a broadcast email titled 'Future

[6] `https://cwiki.apache.org/confluence/display/INCUBATOR/Reports`

of Etch'. Many developers participated in the discussion thread, with seeming agreement that their project is either to be retired or changes are needed. The project mentor brought up the lack of diversity as a possible cause for stagnation, since all developers came from the same company. Some developers concurred, and they feel "*...continual pressure to wrangle new committers...*", and consider that the ASFI values "*...extroverted tendencies of the committers rather than the merits of technology...*".

Eventually, the contributors reached an agreement that the project technology is and will be valuable in the future. Among them, one developer stated that they "*...do not want to see it retired...*" The developers then made a list of future objectives, and worked to make the community thriving again.

§3 Project *Ariatosca* almost succeeded, but eventually failed (the graduation forecast dropped from 96% to 47%). We find that the major reason is that all senior contributors left the project due to their busy day work. At the very end of the project, there were new(er) developers who wanted to contribute to the project. However, since newcomers could only contribute by creating Pull Requests (PRs), and PRs required a senior committer to accept and merge the code changes, they could not submit their code. Attempts at setting meetings with the original developers failed due to busy schedules, and the project was eventually retired.

3.5.4 RQ$_3$ Actionable Recommendation

We start with a caveat. Precise actionable models require interventions and randomized experiments. Our models are not based on such experiments, and any actionability we derive from them must, therefore, be less powerful than those. At best, our experiments can be considered *natural experiments*, a subclass of quasi-experiments where the class assignment is not controlled by the experimenter. Thus, any interventions we suggest here must be validated experimentally in order to avoid large uncertainties in outcomes. With that caveat in mind, we sought to answer, to the best of our experimental methods, the following intervention question: "What action should a project take and when?", in order to increase its graduation forecast in our model.

Here we propose a pragmatic, laissez-faire-unless-needed prospective strategy: to continuously monitor the graduation forecast for significant downturns, and if detected, suggest interventions that may improve the forecast. We deconstruct the intervention question above into two parts: 1) What is a significant downturn? and 2) how to interpret the variables and coefficients in our fitted model into actions. For the following, recall that our interpretable sustainability model gives a graduation forecast from the historical project trace data and the socio-technical project structure, available until that time. It also yields the coefficient of each significant model feature along with its direction for every month.

§1 Identifying significant downturns. We want to identify downturns that dominate any naturally occurring noise or jitter in the forecast. We looked across all projects for how long it takes for a forecast to bounce-up from any downturn. We found that 149 out of the 222 projects (about 67.1%) experienced an upturn or downturn event of a 0.3 magnitude in their forecast probability curve. Of those projects that have not experienced downturns, the average incubation time was 16 months, versus 28 months for projects that have had downturns, implying an association between downturns and length of stay in the incubator. We note that the upturn events tend to be associated with ASF mentors' intervention. This is consistent with one of the responsibilities of ASF mentors: to watch over the projects they are mentoring and keep them on track. However, downturn events do not correlate similarly with extrinsic events; we think they tend to be more related to projects' internal events, e.g., ebbing social interest, core committers leaving, etc. Figure 3.8 shows that the median bounce-up time is about 2.5, and, respectively, 3.5 months for drops of 0% - 5%, and, respectively, $> 5\%$ in the graduation forecast. We also noted that graduated projects seem to bounce-up faster than retired projects. So, we use the median of the latter as the baseline, and define a drop in the forecast of greater than 5% over a period of one or two months to indicate a significant downturn event. This ad hoc approach, while an approximation, is a natural signal processing way to account for inherent uncertainty in the signal.

§2 Actions that improve the forecast. Our model yields real- numbered coefficients for each significant feature in each month, which we aggregate for stability over all months, and obtain the medians as in Fig. 4. Increasing the values of features with positive coefficients and/or decreasing the values of the negative coefficient features results in an increase of the graduation forecast. Thus, once a month or two with a significant downturn is detected, the project developers can look at the most positive and most negative medianed significant features from the fitted model and consider increasing, respectively decreasing, them. This is an approximation and is valid to the extent that the median is representative of the values over all months, which is more the case in the earlier months than the latter ones, see Figure 3.6. The earlier months are the ones we care more about in practice, as we care about being most helpful to nascent projects.

Some of the socio-technical and project features may be difficult to interpret in practice. To aid with this step, we suggest a project should compile an *action table* to summarize possible actions that may positively (or negatively in the reverse way) change the value of model features. (Multiple actions may affect the same feature.) In Table 3.2 we provide one such action table: a mapping between all of our model features and a non-exhaustive set of actions that we identified as likely to move each variable in the positive direction. (The negative actions are not shown, but are complements of those.) E.g., the `e_c_coef`, which counts the number of triangles in the social network, can be increased by increasing emails to everyone and not just the prominent developers or thread starters; conversely, communicating hierarchically in a tree-like fashion would decrease e_c_coef as it will eliminate triangles.

§3 Ecosystem and project-specific fine-tuning. Our strategy can be further fine-tuned in several ways. First, there are common patterns in the graduation forecasts over all projects. Figure 3.9 shows that for graduated projects (in green) there is an apparent upward trend in the forecast in the first 6 months, suggesting that the early-stage development deserves more attention from project managers. From month 6 to month 12 we do not observe a significant change, and the decreasing variance also tends to support such an argument. However, we find increasing variance in months 12 through 18. One possible reason is that many graduated projects achieve their milestone in that period, and slow further commits and discussions, thus lowering the graduation forecast. These

Table 3.2. Positive Actions for Each Feature.

	Positive Action (+)
num_act_dev	Contribute frequently; Advertise, Recruit.
num_emails	Reach out; Ask questions; Encourage communication.
num_commits	Commit frequently; commit smaller; use CI.
num_files	Split files; refactor code; encourage modularity.
c_interruption	Go on vacation often; contribute in bursts.
e_interruption	Email seldom; discourage discussion.
top_e_fract	Encourage core emailers to respond more.
top_c_fract	Core contributors commit exclusively.
c_nodes	Establish technical mentorship; encourage commits.
c_edges	Commit to same files as others; document code well.
c_c_coef	Encourage collaborations, pair programming.
c_mean_degree	Encourage commits by minor contributors.
c_long_tail	Mentor collaborations with newcomers.
e_nodes	Mentor low communicators.
e_edges	Reply to questions; ask questions.
e_c_coef	Encourage non-hierarchical communications.
e_mean_degree	Communicate with minor emailers.
e_long_tail	Foster communication-heavy culture.

considerations can be taken into account during forecast monitoring, with more frequent monitoring chosen or increased attention paid at times around milestones and releases. The ASF committees can also ask for more frequent project reports, during the first 6 months and 12 months of incubation, as project reports were seen to be an incentive to productivity in our case study.

We recognize that some socio-technical elements are more difficult to change compared to others. This is, project-specific, in that some projects can easier modify some features than can other projects. E.g., if the interpretable model suggests increasing `c_edges` and

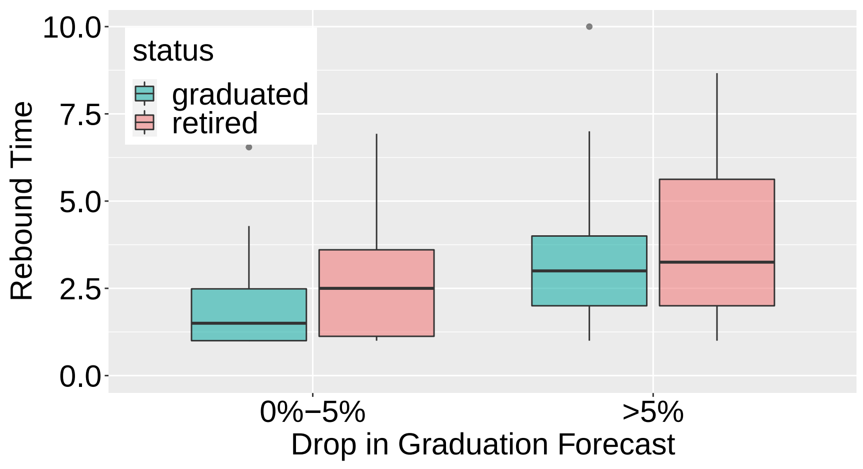


Figure 3.8. Bounce-up after downturns in graduation forecasts for graduated (green) and retired projects (red).

decreasing `c_interruption` this may be easier done in smaller projects than larger ones due to the difficulty of influencing many people at once. Thus these action tables should ideally be project-specific, designed and updated as the project evolves.

Mechanistically, here are some possible reasons for why features, e.g., number of commits/files, may have different effects across projects. First, having much more than the usual amount of code changes and file touches in a given period of time may be indicative of refactoring or even change of direction for a project. This can be the result of a project pivoting which may lead to longer stay in the incubator and possible retirement. Second, increases in the number of files may result in insufficient human resources to handle them (e.g., bugs, documentation, mentoring, and training), resulting in code quality issues and technical debt, especially for smaller projects. This can lead to loss of interest in the small community and eventual disengagement. Third, for overleveraged projects, increase in commits and/or file touches may mean not having time for proper communication, which can slow progress and result in community collapse. In effect, the coefficients of the same factor can vary under different project contexts.

Lastly, a word of caution. In terms of expectations that including more features to intervene on, may lead to faster bounce-up, we note that empirical evidence shows socio-technical features have ways of getting correlated pairwise over time in the same system. Thus, even if not correlated, the effect of increasing multiple features simultaneously may not be additive.

Answer to RQ3: Our strategy can be used as a monitoring tool that feeds suggestions into developers' decision process. Project-specific features can be selected from the suggestions for intervention when experiencing downturns. The algorithm can be made bespoke by introducing more frequent monitoring around releases/milestones.

3.5.4.1 Actionable Strategy Example

Here we apply our strategy on a project from Figure 3.7: *Commonsrdf*, and show specific recommendations following a detected downturn.

While monitoring project *Commonsrdf* [7] we would have observed a significant downturn at months 4 and 5 (¿ 5% drop), see Figure 3.7. For that project, our interpretable model yields as the 3 features with the highest positive medians overall: `top_c_fract`, `top_e_fract`, and `e_nodes`; the three with the most negative median are: `c_c_coef`, `c_interruption`, and `c_nodes`. Consulting Table 3.2, it calls for the project to increase the commit and email contributions by core developers and encourage more committers to communicate. It also calls for the project to decrease collaborations, decrease commit interruptions, and decrease the number of developers that commit. A continuous integration may also be recommended to this project, to keep people on track with smaller, more frequent commits.

What actually happened in the project is that that initial period of downturn was missed; as we saw in our case study for this project, the project manager sent an email titled 'Anybody there?' in month 8, when productivity was already significantly reduced. Using our strategy, the downturn could have been identified and possibly avoided 3 months sooner.

3.6 Threats to Validity and Conclusion

Threats. First, our commit and email data is from only hundreds of projects ASF incubator projects. Thus, generalizing the implications beyond ASF, or even beyond the

[7] `http://mail-archives.apache.org/mod_mbox/commonsrdf-dev/`

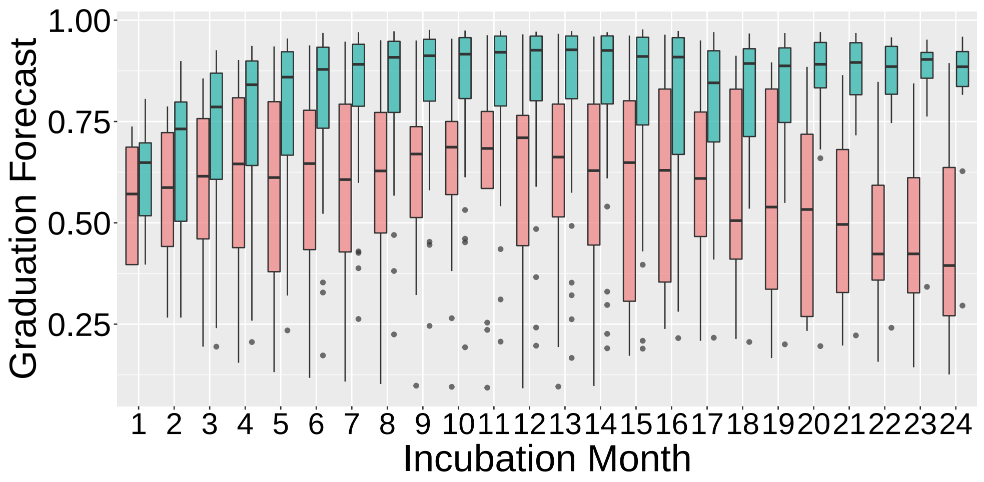


Figure 3.9. Graduation forecasts for all graduated (green) and retired projects (red) over the first 24 incubation months.

ASF Incubator projects carries potential risks. Expanding the dataset beyond ASF, e.g., with additional projects from other open-sourced incubator projects can lower this risk. Second, we do not consider communications other than through the ASF mailing lists. However, ASF's policies and regulations insist on the use of mailing lists, which lowers this risk. Lastly, interpreting deep learning models is still an art, and LIME models are approximations. They may be particularly sensitive to correlated features. We lower such risk by eliminating correlated variables before training. Taking the actionable suggestions given in this paper may result in changes of more than one variable, e.g., increase the active developers may also increase the number of commits.

Conclusion. Understanding why many nascent projects have failed may help others improve their individual practice, organizational management, and institutional structure. Here we showed that quantitative network science approaches combined with state-of-the-art AI methods can effectively model ASF incubator project sustainability, from a novel longitudinal dataset of socio-technical contributions in ASFI projects, more narrow in scope than general OSS projects but with extrinsic graduation/sustainability labels. We also demonstrated the combined power of mixed methods: both quantitative and qualitative studies. Through case studies, we identified specific reasons for success and failure of projects, complementing our models. Finally, we developed a strategy for translational use of the models in practice. Our methods make it straight forward to track a project's trajectory as it progresses toward sustainability, and even offer advice for correcting trajectories upwards. Future work is needed to offer validation of this or similar strategies experimentally.

Chapter 4

Open Source Software Sustainability: Combining Institutional Analysis and Socio-Technical Networks

4.1 Introduction

Open Source Software (OSS) is a multi-billion dollar industry. A majority of modern businesses, including all major tech companies, rely on OSS without even knowing it. OSS contributions are an important manifestation of computer-supported collaborative work, for the high degree of technical literacy typical of OSS contributors. Even though this popularity attracts many software developers to open source, more than 80% of OSS projects are abandoned [82].

The failure of collaborative work in OSS has received attention from two perspectives. In software engineering, the focus has been on understanding success and sustainability from the socio-technical perspective: the OSS developers' day-to-day activities and the artifacts they create. In the management domain, on the other hand, emphasis has been on institutional designs (e.g., policies, rules, and norms) that structure governance and OSS project administration. In particular, systems that generate public goods address these and other endemic social challenges by creating governance institutions for attracting, maintaining, incentivizing, and coordinating contributions. Ostrom [138] defines institutions as "... prescriptions that humans use to organize all forms of repetitive and structured interactions ...". Institutions guide interactions between participants in an OSS project, and can be informal such as established norms of behavior, or more formalized as written or codified rules. These norms and formalized rules, along with the mechanisms for rule creation, maintenance, monitoring, and enforcement, are the means through which collective action in OSS development occur [82], and they can be tiered or nested, as in the context of OSS projects embedded within an overarching OSS nonprofit organization.

Both methods have separately been shown to be utilitarianly describing the state of a process, however, combining the two perspectives has been barely explored. In this paper, we undertake a convergent approach, considering from one side OSS projects' socio-technical structure and the other aspects of their institutional design. Our goal is to use these two perspectives synergistically, to identify when they strengthen and complement each other, and to also refine our understanding of OSS sustainability through the two methodological approaches. Central to our approaches is the idea that trajectories of individual OSS projects can be understood in the convergent framework through the context provided by similar projects that already are being readily sustained or have been abandoned.

We leverage a previously published dataset [128] of traces representing OSS developer's day-to-day activities as part of the Apache Software Foundation Incubator (ASFI) project. These developers are a part of projects that have decided to undergo the process of incubation, toward becoming part of the ASF, and benefiting from the services it provides to member projects. The dataset includes historical traces and a sustainability label (graduation or retirement) for each project. Graduation is an indication of successful incubation and the readiness of a nascent project to join ASF proper, otherwise the project is retired. In other words and importantly, in this paper, we use the ASFI project outcomes of graduation or retirement as a measure of sustainability of the project. We assume that graduated projects are sustained longer than retired ones, although that might not always be the case[1]. But key hurdles that OSS projects have to demonstrate to

[1]For example, it could be that some ASFI retired projects simply could not adapt to the policies and requirements set in the ASFI program but yet continue on, 'in the wild' or perhaps aligned with a

graduate is that they can (1) produce new releases, and (2) show the ability to attract new developers. Both of these factors arguably are key to the sustainability of OSS projects.

We utilize this dataset to study the extent to which graduated and retired projects differ from each other, from the point of view of both the socio-technical structure and the institutional governance. On the socio-technical side, we construct the monthly longitudinal social and technical networks for each project, and calculate several measures describing the features of the networks. On the institutional governance side, we implement a classifier trained on manual annotations of institutional statements in the publicly accessible email communications among ASF participants. Then we compare the findings of our socio-technical and institutional metrics for project-level and individual-level activities. Next, we perform exploratory data analyses, deep-dive case studies, and eventually, we look at how socio-technical measures associate with the prevalence of institutional statements, and evolutionary trajectories during OSS project incubation to sustainability. In summary, we find that:

- We can effectively extract governance content from email discussions in the form of institutional statements, and they fall into 12 distinguishable topics.
- Projects with different graduation (i.e., sustainability) outcomes differ in how much governance discussion occurs within their communities, and also in their socio-technical structure.
- Self-sustained projects (i.e., graduated) have a more socially active community, achieving it within their first 3 months of incubation, and they demonstrate more active contributions to documentation and more active communication of policy guidance via institutional statements.
- A project's socio-technical structure is temporally associated with the institutional communications that occur, depending on the role of the agent (mentor, committer, contributor) communicating institutional statements.

different OSS foundation.

To provide the most relevant context, recently, Yin et. al. [22] showed that socio-technical networks can be used to effectively predict whether a project will graduate or retire from the ASF incubator. That work did not include any institutional or governance analysis. Here, we focus on closing the gap by studying the relationship between the organizational structure (i.e., the socio-technical system) and institutional governance in peer-contributed OSS projects. Our study is the first attempt to provide a common framework for simultaneous, socio-technical structure and institutional, analysis of OSS projects, in order to describe and understand a process affected by both, that is, project gaining self-sustaining and self-governing community and eventually graduating from the ASF incubator. We are hopeful that refining this convergent approach, of structural and institutional analyses, will open new ways to consider and study emergent properties like project sustainability.

4.2 Theoretical Framework

Here we introduce the theories behind the two different viewpoints, Institutional Analysis and Development (IAD) and Social-Technical Systems (STS), as well as Contingency Theory serving as the glue between institutional governance and the organizational structure of OSS projects.

4.2.1 Institutional Theory and Commons Governance

OSS projects are a form of digital commons, or more precisely, Commons-Based Peer Production (CBPP) [82]. Legal scholar Yochai Benkler [139] introduced the phrase CBPP to describe situations where people work collectively over the Internet, and where organizational structure is less hierarchical. While CBPP situations are found in a variety of settings (e.g., collaborative writing, open source hardware) Benkler argues that OSS is the 'quintessential instance' of CBPP.

There is a relatively long history of the study of governance in commons settings, arguably led by Nobel laureate Elinor Ostrom and her groundbreaking book Governing the Commons [140]. Ostrom's Institutional Analysis and Development (IAD) framework was developed to study the governance institutions that communities develop to self-manage natural resources. Much of this research focuses on the governance and sustainability of natural resource settings, e.g., water [141], marine [142], and forest [24] settings.

A key challenge in natural resource commons settings is that individuals who cannot easily be excluded from extracting resources from the pool of available natural resources often have little incentive to contribute toward the production or maintenance of that resource – what are commonly referred to as 'free-riders' [143]. In forest, fishery, and water settings, the free-rider problem in open access settings can lead to a problem termed by Hardin as the 'Tragedy of the Commons' [144]. Ostrom famously pushed back against Hardin's analysis and over a course of a lifetime of work, highlighted that communities can avoid tragedy through hard work in developing self-governing institutions.

OSS commons are fundamentally different from natural resources in that digital resources can be readily replicated and are not subject to degradation due to over-harvesting. Therefore, if over-appropriation is not a problem, is there a potential tragedy of the commons in an OSS context? Invariably the answer is yes, and it lies at the heart of the idea of OSS sustainability. The tragedy occurs when there are free-riders and insufficient human resources available to continue to further develop and maintain the software and, as a result, the software project fails to achieve the functionality and use that was perhaps envisioned when it began, and becomes abandoned [145]. Ostrom and Hess [146] aptly describe this tragedy as 'collective inaction.'

Ostrom's Nobel Prize-winning body of work was studying how humans collectively act and craft self-governing institutional arrangements to effectively avoid the tragedy in natural resource settings. Central in this effort was the introduction and evolution of the Institutional Analysis and Development (IAD) framework [138]. Later, IAD was applied to the study of digital or knowledge commons [147, 146] and explicitly to the study of self-governance in OSS, where Schweik and English undertook the first study of technical, community, and institutional designs of a large number of OSS projects [82].

With that being said, prior work has found that self-governing OSS projects develop highly organized social and technical structures [148]. Those having foundation support, like the ASF, may additionally be in the process of organizing the developers' structured interactions under a second tier of governance prescriptions as required by the ASF Incubator. We refer to an individual institutional prescription as an *Institutional Statement* (IS), which can include rules and norms, and which we define as a shared linguistic constraint or opportunity that prescribes, permits, or advises actions or outcomes for actors (both individual and corporate) [149, 150]. Institutions, understood operationally as collections of institutional statements, create situations for structured interaction for collective action. In other words, configurations of ISs affect the way collective action is organized. In the context of ASF and OSS projects, incubator ISs can affect OSS project social and technical structure.

With IS and other approaches to institutional analysis, it becomes possible to articulate the relationships between governance, organizational, and technical variables. For example, previous studies on OSS often report code modularity as a key technical design attribute [151, 152]. Hissam et al. [153] write: 'A well-modularized system ... allows contributors to carve off chunks on which they can work.' Open and transparent verbal discussion between OSS team members and other ASF officials (e.g., mentors) about OSS project or ASF institutional design, captured in the form of institutional statements, could then predict effort by project contributors to restructure their project's technical infrastructure to be more modular and inviting to new contributors. Using the approaches of institutional analysis, we extract institutional content from open access email exchanges between OSS project contributors to understand the role of communication governance information in OSS project sustainability.

4.2.2 Socio-Technical System Theory

A Socio-Technical System (STS) comprises two entities [117]: the social system where members continuously create and share knowledge via various types of individual interactions, and the technical system where the members utilize the technical hardware to accomplish certain collective tasks. STS theory can be considered to combine the views from both engineers and social scientists, an intermediary entity of sorts, that transfers the institutional influence to individuals [154]. The theory of STS is often referenced when studying how a technical system is able to provide efficient and reliable individual interactions [118], and how the social subsystem becomes contingent in the interactions and further affects the performance of the technical subsystem [119]. Moreover, the socio-technical system theory plays an important role in analyzing collective behavior in OSS projects [19]. OSS projects have also been studied from a network point of view [8, 116]. González-Barahona et al. [120] proposed using technical networks, where nodes are the modules in the CVS repository and edges indicate two modules share common committers, to study the organization of ASF projects. In socio-technical systems, organizations can intervene through long-term or short-term means. Smith et al. [121] propose two conceptual approaches, 'outside' and 'inside': 'outside' approaches represent the socio-technical and are managerial in approach. 'Inside' approaches are more reflexive about the role of management in co-constituting the socio-technical.

From that perspective, the Apache Software Foundation (ASF) community is a unique system that has both outside influence regulations from ASF board and members and inside governance managed or self-governed by individual Project Management Committees (PMC).

4.2.3 Contingency Theory, or There Are No Panaceas in Self-Governance

Contingency theory is the notion that there is no one best way to govern an organization. Instead, each decision in an organization must depend on its internal structure, contingent upon the external context (e.g., stakeholder [122], risk [123], schedule [124], etc.). Joslin et al. [125] find that project success is associated with the methodologies (e.g., processes, tools, methods, etc.) adopted by the project. Here, in particular, we treat the institutional statements as an abstraction of the methodologies in OSS development. As the organizational context changes over time, to maintain consistency, the project must adapt to its context accordingly. Otherwise, conflicts and inefficiency occur [126], i.e., not a single organizational structure is equally effective in all cases. Similar arguments have been made in the field of institutional analysis, arguing that there are no panaceas or standard blueprints for guiding the institutional design of a collective action problem [155].

To address the conflicts caused by incompatibilities with the project's context, previous work suggests thinking holistically. Lehtonen et al. [127] consider the project environment as all measurable spatio-temporal factors when a project is initiated, processed, adjusted, and finally terminated. They suggest that the same factor can have an opposite influence on the projects under a different context. Joslin et al. [125] consider project governance to be part of the project context, concluding that project governance can impact the use and effectiveness of project methodologies.

As per contingency theory, during ASFI projects' incubation, developers and mentors have to make in-time decisions on their organizational structure, contingent on what is happening in the institutional rules and governance, and vice versa.

4.3 Research Questions

Reflecting on the previous discussion, the primary goal of this paper is to demonstrate that the evolution of a project from a nascent state to a sustainable state can be studied effectively by combining the two different methodologies of socio-technical network analysis and institutional analysis.

We reported in prior sections that a variety of scholars have utilized a socio-technical systems approach to analyze collective behavior in OSS projects. We also described how institutional analysis is useful in understanding collective action in OSS settings. To enable to dual-view on sustainability, we first describe and evaluate our automated approach to identifying institutional statements in project emails.

RQ$_1$: Are there institutional statements contained in ASF Incubator project email discussions? Can we effectively identify them?

With the next two research questions, we assess the utility of our convergent approach to the Institutional Analysis (IAD) and STS frameworks. In the case of the ASF incubation program, there are two eventual outcomes: either a project graduates from the ASF incubator and becomes a full-fledged ASF-associated project, or it retires without achieving that goal. In this context, we operationalize a sustainable state as one where an OSS project graduates from the ASF incubator program, rather than retires. We ask:

RQ$_2$: Is OSS project evolution toward sustainability readily observable through the dual lenses of institutional and socio-technical analysis? And how do such temporal patterns differ?

Per institutional analysis theory, strategies, norms, and rules can affect the social and technical organizations of projects. Governance and organization, per social theories, must work hand-in-hand to make viable socio-technical systems. Ill-designed institutional arrangements would introduce inefficiencies into the system, and such inefficiencies may amplify deviant behaviors and irregular structures in the system. Such influential links from institutional design to the organizational structure can be, in fact, bi-directional. In effect, in a sustainable system, an ill-formed organizational structure may instigate new rules to adjust and improve such structure, further improving efficiencies in the systems.

Thus, we hypothesize that the feedback, if any, between project governance and project organization should be observable, specifically in that intensified governance discussion should precede and/or follow changes to the project organizational structure. As a reminder, we consider institutional statements as indicators of intensified discussions of OSS project self-governance or new incubator requirements on that self-governance. We also consider socio-technical network parameters as indicators of organizational structure. Thus, we ask:

RQ$_3$: Are periods of increased Institutional Statements frequency followed by changes in the project organizational structure, and vice-versa?

In the following section, we introduce the methodologies approaching the above three research questions.

4.4 Data and Methods

To study the difference between projects that graduate ASFI (i.e., become sustainable) and those that do not, in this paper we use a collection of large-scale data sets comprising Institutional Statements and Socio-Technical variables extracted from all graduated and retired projects from the Apache Software Foundation Incubator, ASFI. In ASFI, graduation is an indication that a nascent project is sufficiently sustainable to join ASF proper[2], otherwise the project is retired. Our combing through the Apache lists, inspecting the data, and speaking to project and community members have shown that almost all failures to graduate are sustainability failures. On rare occasions, some projects have retired for reasons other than sustainability, e.g., some are not a good fit for the Apache model[3], despite evidence that projects are generally sufficiently aware of the ASF model before entering incubation according to their project proposal[4].

For the socio-technical networks, we collected historical trace data of commits, emails, and incubation outcomes for 253 ASFI projects, which have available archives of both commits and emails from 03/29/2003 to 02/01/2021[5]. Among those, 204 projects have already graduated, and 49 have retired. ASF incubator projects that are still in incubation are not studied in this paper.

We collected the ASF incubator project data from the ASF mailing list archives[6], which are open access and can be retrieved through the archive web page lists, `http://mail-archives.apache.org/mod_mbox/`. They contain all emails and commits from the project's ASF incubator entry date, and are current. The project URLs follow the pattern: `proj_name - list_name/(YYYYMM).mbox`. For example, the full URL for the *dev* mailing list of the *Apache Accumulo* project, in Dec 2014, is `http://mail-archives.apache.org/mod_mbox/accumulo-dev/201412.mbox`. Each such *.mbox* file contains a month of mailing list messages from the project, for the date specified in the URL. Here *dev* stands for 'emails among developers'. Notably, there are some sites that are not following the pattern, e.g., 'ASF-wide lists' are not project-owned mailing lists, and the list 'incubator.apache.org' contains data of more than one project.

To extract Institutional Statements, we combined our email data set with a prior data set on ASF policy documents. In a given organization, institutional statements are characterized by a finite set of semantic roles (e.g. ASF Board, Mentors, contributors, etc. in ASF), and their interactions (e.g. management committees requesting reports from

[2]ASF's guide to project graduation: `https://incubator.apache.org/guides/graduation.html`
[3]ASF's reason behind projects' retirement: `https://incubator.apache.org/projects/#retired`
[4]ASF incubator projects' proposal `https://cwiki.apache.org/confluence/display/INCUBATOR/Proposals`
[5]Our code and data is available at Zenodo: `https://doi.org/10.5281/zenodo.5908030`
[6]During the submission of this study, ASF had moved their email archives to Pony Mail system.

projects, developers voting to induct committers in ASF), in specific contexts. To account for their representation in our training corpus, we included institutional statements from not only ASF project-level email exchanges among participants, but also ASF policy documents. The supplementary set of Institutional Statements included 328 policies, which were compiled from ASF policy documents (e.g., Apache Cookbook, PPMC Guide, Incubator Policy, etc), in an economic analysis of the ASF Incubator's policies [156].

4.4.1 Pre-processing

We collected all 1,330,003 emails across the ASF Incubator projects, from 03/29/2003 to 02/01/2021 (under mailing lists of 'commit', 'dev', 'user', etc.). We find that 128,257 (about 9.6%) emails are automatically generated and broadcast by continuous integration tools (i.e., bots). Because the amount of such emails is substantial, but they carry less meaningful social or institutional information, and list members rarely reply to them, we use regular expression rules to identify and eliminate them from the corpus, leaving us 1,201,746 emails.

And, for the technical contribution side, many projects, especially those over ten years old that used SVN, utilized a bot for extensive mailings, thus forming outliers in the dataset. Thus, we eliminate commit messages from automated bots (e.g., 'buildbot'), 253,758 out of 3,654,196 (about 14.4%) commit messages, and email messages from issues/bug tracking bots (e.g., 'GitBox'). Moreover, we find some developers contributed commits by directly changing/uploading massive non-source code files (e.g., data, configuration, and image files). Since committing non-coding files can form outliers in the data set, we choose to apply the *GitHub Linguist*[7] to identify 731 collective programming language and markup file extensions, and exclude any other non-coding commits (e.g., creating/deleting folders, upload images, etc.).

4.4.2 Constructing Socio-technical Networks

Network science approaches have been prominent in studying complex systems, e.g., OSS projects [131, 111]. Since networks can contain rich information for both the elements (i.e.,

[7]GitHub Linguist `https://github.com/github/linguist`

nodes) and their interactions (i.e., edges), in this study, we use socio-technical networks to anchor the abstraction of socio-technical systems. We define the projects' socio-technical structure using social (email-based) and technical (code-based) networks, extracted from their emails to the mailing lists and commits to source files. Similar to the approach by Bird et al. [19], we form a social network (weighted directed graph) for each project in each incubation month, from the communications between developers: a directed edge from developer *A* to *B* forms if *B* has replied to *A*'s post in a thread or if *A* has emailed *B* directly. The weight of the edge represents the communication frequency between a pair of developers. The technical bipartite networks (weighted bipartite graph) are formed in a similar way. For each project in each month, we include an un-directed edge between a developer *A* and a source file *F* if developer *A* has committed to the source file *F* that month (excluding the SVN branch names). The weight of the edge represents the committing frequency between the developer and the source file. In summary, social networks are weighted directed graphs. We form edges between two developer nodes, if one developer replied to or referenced the other's email. Technical networks are undirected bipartite graphs, with developers forming one set of nodes, coding files forming the other, and a link being drawn when a developer contributed to a coding file. We use the NETWORKX package from *Python* for the network-related implementation.

4.4.3 Extracting Institutional Statements

We combined the email exchange data set with the ASF policy document data to fine-tune a BERT-based [157] classifier, for automatic detection of ISs (see Sect. 2.1 for the definition of IS).

To start, we hand-annotated a small subset of our data for ISs as follows. After selecting a random subset of 313 email threads from incubator project lists, two hand-coders labeled the sentences in them as 'IS' or 'Not IS', on the basis of whether they fit the definition of Institutional Statements. They resolved disagreements through discussion and recorded these conclusions, achieving a peak out-of-sample agreement between 0.75 to 0.80. A sentence was coded as an IS only if it was a complete sentence; fragments such as parenthetical mentions of rules or resources were not annotated as positive. This

Table 4.1. Selected Examples of Institutional Statements Found in ASFI Project Email Discussions.

Project	Date	Institutional Statements
Airflow	21 Dec 2016	... running in our Lab there is virtually no restriction what we could do, however I will hand select people who have access to this environment. I will also hold ultimate power to remove access from anyone ...
ODF	07 Dec 2011	Please vote on releasing this package as $<$ *Package* $>$. The vote is open for the next 72 hours and passes if a majority of at least three +1 ODF Toolkit PMC votes are cast ...
Airflow	24 Feb 2017	... Next steps: 1) will start the voting process at the IPMC mailinglist. ... So, we might end up with changes to stable. ... 2) Only after the positive voting on the IPMC and finalisation I will rebrand the RC to Release.

resulted in 6,805 labeled sentences (i.e., 'IS' or 'Not IS'); 273 were labeled as IS.

We treated all 328 policies from the ASF documents as institutional statements, since policy documents provide arguably more formal institutional sample text compared to the norm in the email discussions. Thus, we had 601 Institutional Statements in total across these two coded datasets.

Institutional statements refer to prescriptions and shared constraints in the form of norms, rules, and strategies that are meant to mobilize and organize actors towards collective actions. The examples of institutional statements provided in Table 4.1 provide some instances of developer exchanges that encompass norms and strategies with institutional implications. The first example from the Airflow project, dated 12/21/2016, involves a situation where certain developers find the computational infrastructure provided by ASF insufficient for testing and development requirements, and discuss setting up alternate arrangements to meet the bottleneck. Faced with resource limitations, one developer offers an externally hosted cloud environment through his private resources. The selected excerpt is a quote from the individual establishing the terms for using the alternate resources he may offer to the project members, including access permission and usage restrictions. ASF projects conduct voting from time to time to gather community consensus on matters of significance. The following example from ASFI project ODF, dated 12/07/2011 describes the stepwise process expected to be followed by members project-wide to conduct a vote that decides on the approval of the release of the current candidate under development. The final example from Airflow, 02/24/2017 also pertains to a similar process, where a developer discusses the voting process and the implications, especially in terms of subsequent steps that need to be fulfilled to ensure product release.

BERT-based Sequential Classifier. In natural speech, such as emails, ISs can appear as whole sentences, parts of sentences, or span multiple sentences. They are also relatively sparse, with their institutional quality dependent on their inherent interpretation as well as context. Framing IS extraction as a sequential sentence classification task in the context of self-contained email segments, instead of labeling individual sentences helps take into account contextual cues.

We used the sequential sentence classifier developed by Cohan et al. [157], which leverages Bidirectional Encoder Representations from Transformers (BERT) sequence classifier [158] to classify sentences in documents. BERT can be employed to generate the representation for a sentence, through joint encoding over its neighboring sentences and then leveraging the corresponding sentence separator '¡SEP¿' token's tuned embedding for downstream applications, such as sentence labeling, extractive summarizing, etc. Thus, our classifier comprises BERT for attention-based joint encoding across sentences followed by a feedforward classifier to predict sentence labels based on these separator '¡SEP¿' vectors.

To test the performance of the classifier on email IS extraction, we held-out 40 email threads (12.5%, randomly split) out of our 313 hand-annotated email threads. The training was performed on the combined set of the remaining 273 coded email threads and the ASF policy documents. The coded training and, respectively, testing email data contained 231 and, respectively, 42 institutional statements. For both training and testing, email threads were processed to generate classifier inputs as follows. To include neighboring context while meeting length limits of the BERT-based text classifier, for each email document, sentences were first chunked into segments using a sliding window of up to 256

BERT sub-word (word piece) tokens. This resulted in segments containing 6 contiguous sentences each, on average, comprising as many full sentences as could be accommodated in the specified subword limit. The rolling window had a step of 1 full sentence. We generated 3322 and 384 email segments for training and testing, respectively. For the policy documents, each policy with its sentences was treated as a segment, leading to 328 additional segments in the training data. There are several reasons to support the inclusion of ASF policies to augment positive training examples. (1) In terms of semantic information, they are about institutional themes and actions. This was expected to help the language model learn what sets apart Institutional themes from regular development activities and artifacts. (2) ASF policies are critical in common pool resource management and institutional operations as they describe roles, responsibilities and regulate actions, and are often invoked in email discussions[8]. (3) The institutional statements of the formal policies are the source texts that in-email references to IS are drawing from when they discuss ASF's rules in email. From this perspective, they are a vital source text for detecting these statements as they occur in email settings. Hence, while apparently sourced from formal bylaws beyond emails, ASF policies are indeed institutional statements relevant and recurring in developer conversations and are hence included in the training data.

We fine-tuned our classifier end-to-end against the corresponding labels for sentences in the segment. The training stage was conducted with a batch size of 16 and a learning rate of $2 \cdot 10^{-5}$, for 6 epochs. All other hyperparameters were left as defaults. To account for the class imbalance, we randomly oversampled training data segments that had at least one IS sentence to match the number of segments that had no IS sentences (1:1). In both the training and predicting phase, we did not incorporate any temporal information, other than the sequentiality captured by the segments. That is, when extracting the institutional statements, the model does not require the exact time of the discussion.

During testing or prediction, due to variable length of context preceding or following each sentence in any particular segment, we treat a sentence in an email as a 'positive' classification, if it has been detected as an IS in at least one segment. The performance of

[8]`https://lists.apache.org/thread/zykybdvnk9cwx03pnrfl2br9nkcb7q3f`

the model has been reported in terms of the F1-score, precision, and recall with respect to the positive ('IS') label detected for sentences in the test email set in Sect. 5.1.

4.4.4 Topics Identification in Institutional Statements

The purpose of text modeling is to describe the text given a specific corpus, and provide numerically measurable relationships among texts, e.g., topics identification, measuring similarity, etc. We use a Latent Dirichlet Allocation (LDA) model to get semantically meaningful topics to better understand the extracted institutional statements. LDA is an unsupervised clustering approach [159], which when given a set of documents, iteratively discovers relevant topics present in them, based on the word distributions and relative prevalence in each document. We used LDA to identify prominent topic clusters occurring among all institutional statements extracted from our email archives through our trained classifier (see Sec 4.3). No prior training from our coded email set against pre-identified topic labels was used to train the LDA model. We use the coherence score provided by the *gensim* package [160] to optimize the performance of the LDA model with respect to the number of topics; a higher coherence score represents a better clustering performance. We select the LDA model with the highest coherence score from which to draw the clusters. However, since the LDA model does not automatically generate a label for each cluster, we need to assign a label intuitively based on our domain knowledge of the ASF incubation process. Naming each topic cluster certainly carries some risks on interpretation, however, we believe that providing all top keywords for each cluster reduces such risk.

4.4.5 Variables of Interest

We draw institutional and socio-technical project features and variables on the basis of each framework's predictions for our research questions. Our socio-technical variables are pulled from a recent study on forecasting the sustainability of OSS projects [22], showing high predictive power of socio-technical variables. All metrics are aggregated over monthly intervals, for each project, from the start to the end of its incubation.

Longitudinal Socio-Technical Metrics: For each project network, for each month, we constructed the social and technical networks, and from them calculate various organi-

Table 4.2. Summary statistics for the monthly socio-technical variables and the counts of institutional statements from project mentors, committers, and contributors after removal of the top 2% of outliers. The numbers in parentheses denote the values after the removal of inactive months (i.e., absent of emails/commits). Prefix s_ denotes features in the social network while t_ represents the technical network.

Statistic	Mean	St. Dev.	25%	75%
s_num_nodes	13.04 (16.96)	14.56 (15.04)	4 (7)	17 (22)
s_graph_density	0.30 (0.30)	0.27 (0.22)	0.12 (0.14)	0.40 (0.40)
s_avg_clustering_coef	0.22 (0.29)	0.23 (0.21)	0 (0.11)	0.39 (0.43)
s_weighted_mean_degree	11.83 (15.56)	12.03 (12.81)	4 (7.43)	16 (19.71)
t_graph_density	0.37 (0.68)	0.41 (0.32)	0 (0.36)	1 (1)
t_num_dev_nodes	1.18 (2.21)	1.59 (1.60)	0 (1)	2 (3)
t_num_file_nodes	60.99 (114.83)	153.94 (197.25)	0 (6)	38 (126)
t_num_file_per_dev	28.79 (53.57)	80.46 (104.23)	0 (4)	20 (54.5)
num_IS_mentor	15.46 (15.99)	24.46 (25.01)	0 (1)	20 (20)
num_IS_committer	9.34 (12.89)	19.36 (22.36)	0 (0)	10 (16)
num_IS_contributor	13.18 (16.36)	21.72 (24.42)	0 (2)	18 (21)

zational structure measures. In our tables and results, the prefix t_ in a variable's name indicates it is of the technical (code) network, while the prefix s_ in a variable's name indicates it is of the social (email) network. For the monthly social networks, we calculate the weighted mean degree s_weighted_mean_degree (sum of all nodes' weighted degree divided by the number of nodes), average clustering coefficient s_avg_clustering_coef (the average ratio of closed triangles over open triangles), graph density s_graph_density. In the technical bipartite networks, for each month, we calculate the number of unique developer nodes t_num_dev_nodes, the number of unique file nodes t_num_file_nodes, the number of files per developer t_num_file_per_dev, and the graph density t_graph_density.

Institutional Statements Frequency Metrics: For each project, for each month, we added up the ISs in all emails of that month sent by each of the following three separate and identifiable groups of people: ASF mentors (num_IS_mentor), registered ASF committers (`num_IS_committer`), and contributors (`num_IS_contributor`). We summarize their statistics in Table 4.4.5. As noted earlier, there is a final group of emails not accounted here, sent by bots. Similar to calendar entries, they may be useful, but are not the object of our study here.

4.4.6 Granger Causality

Time series data allows for the identification of relationships between temporal variables that go beyond association. One approach, *Granger causality*, is a statistical test for identifying quasi-causality between pairs of temporal variables [161]. Given two such variables, X_t and Y_t, the Granger causality test calculates the p-value of Y_t being generated by a statistical model including only Y's prior values, Y_{t-1}, Y_{t-2}, etc., versus it being generated by a model that in addition to Y's prior values, also includes X's prior values X_{t-1}, X_{t-2}. Thus, Granger causality simply compares a base model involving only Y to a more complex model involving Y and X, and calculates if the latter is a better fit to the data. In the context of Granger causality, prior values are called *lagged values*, with X_{t-1} having a lag of 1, X_{t-2} having a lag of 2, etc. If the Granger causality test returns a small enough p-value (e.g., < 0.01), it is interpreted as the rejection of the null hypothesis, thus establishing that X *Granger causes* Y.

The Granger causality test makes an assumption that the time-series on which it is applied are stationary, meaning they do not have a trend or seasonal effects. It is necessary to test for stationarity before running the Granger causality. We use the augmented Dickey-Fuller test [162], as implemented in ADF.TEST from the *R* package TSERIES [163], to test stationarity. Both institutional and socio-technical variables were found to be stationary. We note that a distinction is typically made between scientific causality based on controlled experiments, and Granger causality, with the latter only satisfying one (precursor property) of multiple different properties of causality. Because of that, when Granger causality is used, the word 'causality' is always preceded by 'Granger'. We also note that this test does not identify the sign, if any (i.e., positive or negative) of the Granger causality. It simply says if one exists. We use the PGRANGERTEST function to test Granger causality.

4.5 Results

In this section, we answer the proposed research questions by adopting a dual view, from the institutional analysis and socio-technical network perspectives. We first establish the utility of our IS identification methodology.

4.5.1 RQ_1: Are there institutional statements contained in ASF Incubator project discussions? If any, can we effectively identify the content of ISs?

Detecting Institutional Statements. First, we focus on the ability of our BERT-based classifier to identify institutional statements in emails. When tested on the 857 held-out sentences from the 40 email threads in our test set, see Sect 4.3, our classifier achieved a precision score of 0.667, recall score of 0.681, and F1 score of 0.674 on classifying Institutional Statements, demonstrating it is able to extract ISs from developer email exchanges in spite of there being only 5.1% ISs.

For model validation against overfitting, we sought to perform stratified cross-validation (CV) on our training data. We note that our data was not ideal for a CV study: we had (1) limited data size (2) uneven distribution of ISs across the email threads and (3) class imbalance between IS and non-IS sentences. E.g., due to the limited data size, emails with high IS density could find their way in the train but not the test split, and dramatically increase the variance in cross-validation results. To ameliorate that, for more uniform stratification we chunked up each of the 273 threads in our training data into 442 sub-emails of 20 contiguous sentences each (the email threads had a mean length of 22 sentences). We fine-tuned our classifier end-to-end against the corresponding labels for sentences in the sub-emails. The subsequent input segment generation and training of the pipeline were otherwise kept unchanged. We obtained a mean F1 score on positive labeling of sentences with ISs of 0.603, with some high IS variability between folds still persisting.

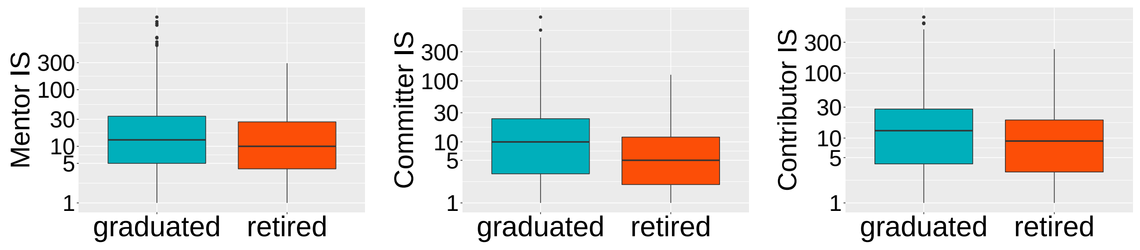
We consider these performance results satisfactory given that we had a small and highly imbalanced data set (273 ISs out of 6,805 sentences). There are strong indications that increasing the positive examples in the training data set will further increase our classifier's performance.[9] Of course, it is challenging to ascertain if classifier performance varies across projects due to limited data size and expected differences in underlying distribution of institutional themes and policy mandates across projects.

We ran our classifier on the full corpus of 1,201,746 emails (after bot email removal) across all ASF incubator projects. It identified 313,140 ISs in the emails, for an average of 0.261 sentence-level ISs per email. Table 4.4.5 shows descriptive statistics for both the socio-technical variables and the number of institutional statements from project mentors, committers, and contributors, calculated in monthly intervals, per project.

We find that the classifier's errors are also informative. In one set of false positives, participants described plans for an event occurring outside of Apache and the relevant incubator project, not the kind of process or behavioral constraint typical of ISs. It was probably detected as an IS due to its semantic similarity to rules and guidelines which make up other positive examples. Conversely, the sentence '*Send it to ¡EMAIL¿ and see what the reaction is*' was missed as an IS, despite appearing in the context of contributor agreements. This miss is likely due to the fact that many such recommendations are made in the emails that would not be considered institutional, because they indicate a particular individual as an individual, rather than in their institutional role.

<u>Institutional Statements Over Roles and Sustainability Status</u>. We turn to some exploratory analysis, to demonstrate the utility of our chosen features when reasoning about differences between graduated and retired projects. Comparing graduated and retired projects, we find a significant difference in the number of ISs. For example, in Figure 4.1(a), the number of IS sent by mentors in graduated projects is statistically higher than retired projects (the Mann-Whitney U test is used for testing the difference in means). This, along with the fact that graduated projects tend to be more active socially overall compared to retired projects (i.e., more email exchanges), suggests the mentors of retired projects are concerned about the projects' community progressing, thus, most of the email content is about rules and guidance. On the other hand, it is also plausible that mentors engage more socially and less institutionally with graduated

[9]When we fine-tuned the classifier with only the 273 training email threads (i.e., without Institutional statements from the ASF policy documents), the F1 for positive label was found to be about 20% lower.



(a) Num. Mentors IS ($p < .001$) (b) Num. Committers IS ($p < .001$) (c) Num. Contributors IS ($p < .001$)

Figure 4.1. Comparing graduated (in blue) vs retired (in red) projects along the number of Institutional Statements (IS) (color online). The Mann-Whitney U test p-val is sufficiently small (in brackets), suggesting significant differences in means between groups.

Table 4.3. Topics Identified in Institutional Statements.

ID	Heuristic Topic	Top Sample Words
1	Progress Report	review, require, meeting, board, submit, report
2	Collective Decision	vote, start, proposal, thread, close, day, bind
3	Project Release	release, issue, think, fix, branch, policy
4	Community	project, email, send, community, behalf, incubation, talk
5	Report Review	board, report, time, meeting, prepare, reminder, review
6	Mailing List Issues	list, mailing, discussion, question, issue, comment, request
7	Documentation	update, wiki, page, website, documentation, link, doc
8	Software Testing	release, source, build, test, note, artifact, check
9	licensing Policy	license, file, software, version, copyright, compliance
10	Routine Work	project, committer, help, work, way, code
11	Mentorship	podling, report, form, mentor, know, sign, month, wish
12	Software Distribution	work, repository, information, file, distribute, commit

projects, which may benefit those projects more. The numbers of ISs sent by committers and contributors show similar patterns. We investigate them longitudinally in the next section.

Topics Identification in Institutional Statements. We use the Latent Dirichlet Allocation (LDA) model to study the token-level topics in institutional statements. By optimizing the LDA coherence score, we get the optimal number of topics of 12. The result further enables us to study which words are important to each topic. We present the clusters of top words for each topic in Table 4.3.

As this table reveals, words are well extracted from the institutional statements and are distinguished from each other. For example, in the first topic (i.e., 'Progress Report'), there is a cluster of words – 'review', 'board' (which relates to ASF board), 'submit', and 'report' – all of which are associated with the important incubator rule that requires projects to report regular progress reports. While in topic 7, words like 'update', 'wiki', 'page', 'website', and 'documentation' emerge, all related to requirements projects need to address related to their website or documentation requirements. The results advance the institutional theory under the software engineering domain, arguably that the IS is associated with OSS sustainability, suggest diving deeper into the connections between the social-technical system and institutional analysis.

RQ$_1$ Summary: We demonstrated that institutional analysis methodologies can capture differences between graduated projects and retired projects. We also showed that we can effectively identify meaningful institutional statements, and common topics, from ASF incubator projects' emails.

4.5.2 RQ$_2$: Is OSS project evolution toward sustainability observable through the dual lenses of institutional and socio-technical analysis? And how do such temporal patterns differ?

In this section, our goal is to contrast graduated and retired projects over time in both IS space and socio-technical space. Projects exit the ASF incubator at different times. In effect, there will be a larger variance during the end of the incubation month. Therefore, we restrict ourselves to the first 24 months for all projects (more than 60% projects stayed within 24 months in the incubator).

Topic Evolution Over Time. After identifying the words that contribute to various identified topics, by aggregating over all projects, we get the volume, which is measured

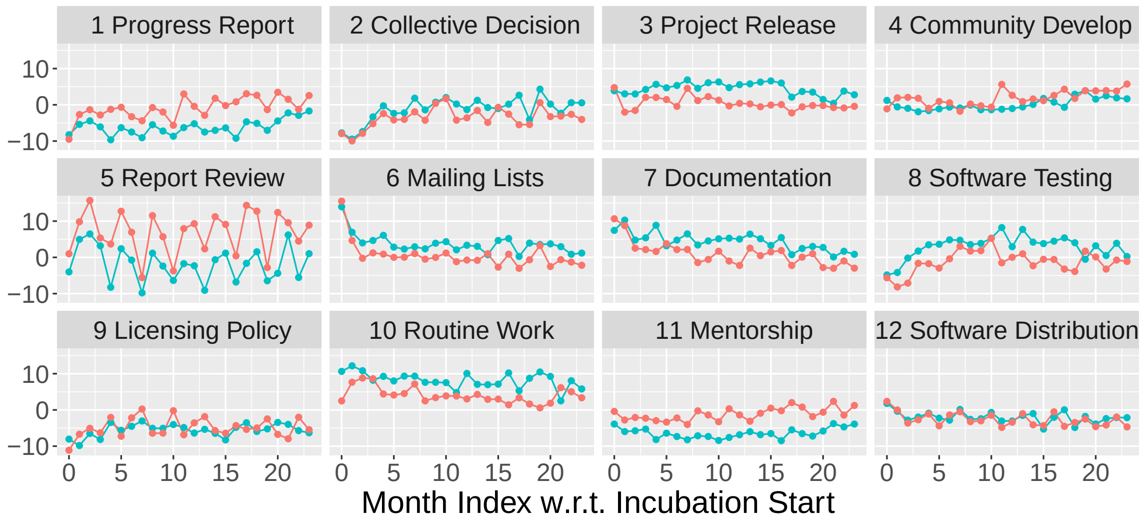


Figure 4.2. Topics Evolution for graduated projects (in blue) compared to retired projects (in red). The x-axis indicates the i-th month from their incubation start and the y-axis represents the relative volume of the topics. Mann-Whitney U test found 10 out of 12 topics are significantly different in their means between graduated and retired projects (p-val ¡ 0.01). Not significant were topic 9 (licensing policy) and topic 12 (software distribution).

by the number of tokens contributing to that topic, of each topic in each month. Moreover, since there exist trends in the number of IS, we subtract the mean volume for each month, separately for the graduated and retired projects. We present them in Figure 4.2, where the x-axis is the number of months after their incubation start, and the y-axis indicates the relative volume compared to the mean.

The results of Mann-Whitney U test show 10 out of 12 topics are significantly different in their means between graduated and retired projects (p-val ¡ 0.01). Not significant were topic 9 (licensing policy) and topic 12 (software distribution). Additionally, the augmented Dickey-Fuller test suggests that over time, 9 out of 12 topics are not stationary (i.e., temporal trends exist, with p-val ¡ .01), except for topic 2 (collective decision), topic 6 (mailing lists), and topic 12 (software distribution). The testing results prompt us to analyze the difference in project-level dynamics between graduated and retired projects.

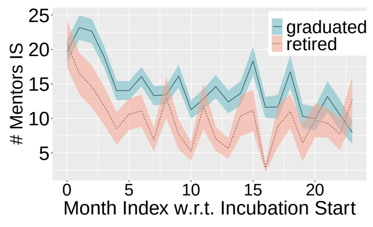
We observe an increasing trend of Topic 1 'Progress Report' with a small seasonal effect, suggesting the projects are learning the 'Apache Way' and more actively discussing their regular project reporting over time. And such seasonal effect is found to be more significant in Topic 5 (‘Report Review’). Project releases, documentation, and software testing, are all connected to the number of people participating regularly. Retired projects are on average smaller than the graduated ones, which is the likely explanation for the differences. E.g., in Figure 4.3(f), we show that graduated projects, on average, have more source files than retired projects. Moreover, we find that Topic 9, ‘license policy’, has an increasing trend in the earlier stages of incubation (e.g., months 1-7) which makes sense in that the shift from one OSS license to the license required by ASF is an important discussion that projects would want to address earlier on.

On the contrary, the longitudinal pattern of IS language related to software testing is relatively rare at the beginning of project incubation. It suggests that in earlier stages of incubation, developers are more likely focused on the transition to the incubator and perhaps less on new code development and testing. On the other hand, such transitions were implemented in a fast manner, with testing discussions increasing rapidly in incubation months 3, 4, and 5.

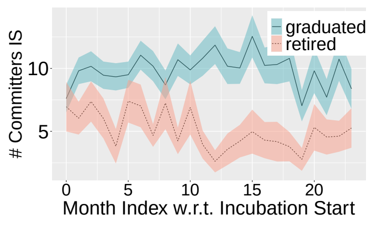
By comparing graduated and retired projects, we find that, Topic 10, ‘Routine work’, to be the dominant topic for both types of projects, almost through all projects’ incubation (i.e., remain high volume compared to other topics). We also find that graduated projects tend to be more active on Topic 7 ‘Documentation’ and Topic 3 ‘Project Release’. Interestingly, on the other hand, mentorship-related ISs (Topic 11) are found to be more active in retired projects rather than in graduated projects. One possible reason is that retired projects did seek help from their mentors when their projects were experiencing downturns, and further issuing institution-wise statements.

Metric Evolution. We continue by exploring the evolution of our metrics over time. Looking at the mentors’ ISs, shown in Figure 4.3(a), we can see that even at the beginning of their incubation, mentors email a greater number of ISs to projects that eventually graduate compared to ones that eventually retire.

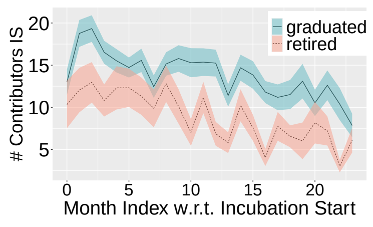
Next, we see that the number of ISs in mentor emails decline for both graduated projects and retired projects before month 5, suggesting that ASFI mentor activity may decrease after incubating projects work through the first steps of the incubation process.



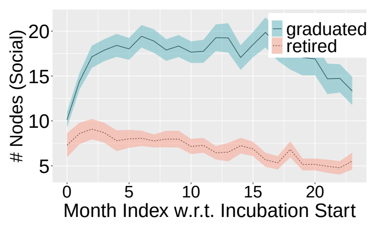
(a) Num. IS from Mentors



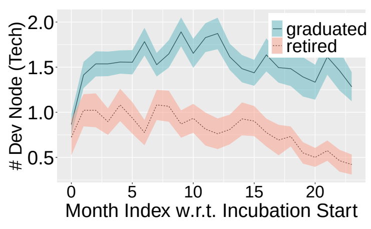
(b) Num. IS from Committers



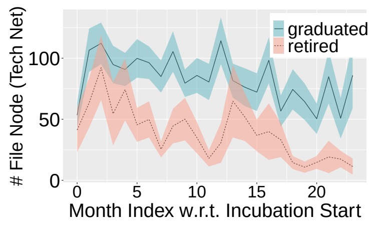
(c) Num. IS from Contributors



(d) Dev Nodes in Social Network



(e) Dev Nodes in Tech Network



(f) File Nodes in Tech Network

Figure 4.3. The averaged monthly IS and ST variables between graduated projects and retired projects. On the top are the IS measures; On the bottom are ST measures. Shades indicate one st. error away from the mean. Month index 0 indicates the incubation starting month (color online).

Then, we visually identify an increasing trend of IS from mentors around month 6 for graduated while 5 for retired projects. One possible reason is the fact that mentors start helping projects when they are experiencing difficulties or downturns. It is consistent with ASF mentorship that during the early stage of the incubation, developers are required to make institutional-related decisions, e.g., voting for reports, discussing the ASF required licensing, and the community-related issues, and it is in these kinds of areas where mentors come to help.

On the Socio-Technical networks side, shown in Figure 4.3(d), for the first 6 months, we can see the graduated projects have a clear increasing trend in the number of nodes in social networks, while it seems to be constant in retired projects. We can see a slight decrease around month 10 to month 12 for both types of projects, suggesting 10 months might be a good timing for mentors to intervene/motivate their projects, if they are experiencing some difficulties.

RQ$_2$ Summary: We identify socio-technical and institutional signatures of OSS project evolution, and evidence that it differs between graduated and retired projects, and that these patterns can even be distinguished by institutional heuristic topics. On the institutional side, both graduated and retired projects have more stable institutional topics during their first 3 months. On the Socio-Technical network side, graduated projects keep attracting community over their first 6 months, while retired projects are unstable during their first 3 months.

4.5.3 Case Study: Association Between Institutional Governance and Organizational Structure

To communicate concretely how the institutional and socio-technical dimensions interact within the ASFI ecosystem, we showcase four diverse instances of their mutual interrelationship.

Case A. In July 2011, the *HCatalog* project announced a vote for its first Release Candidate (RC), the first officially distributed version of its code [10]. Because a project's RC's reflect on the whole ASF, they require approval from the foundation after project contributors have given their approval. In preparation for the first vote, developers double-checked the installation process and reported missing files and features. This drove contributions to the code and documentation, e.g., release notes were added after being reported missing. The contributors then cast their votes. With four people's votes, the product was approved and a proposal was forwarded to Apache Incubator leadership for approval.

Case B. In December 2010, an independent developer emailed the *Jena* project community to share their idea for a new feature, and was asking how to proceed toward contributing it [11]. Their query includes policy questions, such as whether they must obtain an Individual Contributor License Agreement (ICLA). A developer responds that the policy does not require an ICLA for the type of smaller contribution that the volunteer is proposing. The developer then guides the volunteer through established project processes for contributing to the code, including what mailing lists to use and how to submit their

[10] `https://lists.apache.org/thread/p88lpxn3gtprc11xw20jbnrmp3f8fmpw`
[11] `https://lists.apache.org/thread/8dgbvjkwosbvxg6oj6ptyssn0m4rdto1`

feature as a patch.

Case C. In December 2016, a developer in the *Airflow* project community raised concerns over the integration testing infrastructure offered by Apache, citing unnecessary obstacles it imposes on volunteer contributors[12]. The developer offers their resources as an alternative, with the caveats that they will administer it and control access. This triggers a discussion on the technical merits of the developer's concerns, and a policy discussion as to whether ASF permits the use of unofficial alternative infrastructure options. Several developers conclude that a transition is technically advisable and institutionally sound, and the community transitions to the alternative integration testing framework.

Case D. In September 2015, the *Kalumet* project received a proposal that it be retired from ASFI after its code had been languishing for several months. Contributors agreed upon retirement almost unanimously. One contributor, identifying features of the project that could be of use to other ASF and ASFI projects, suggests distributing key parts of its functionality to other active projects. The retirement vote is ultimately followed by developer effort distributing *Kalumet*'s assets.

These cases illustrate how institution-side policy discussion and sociotechnical-side project contributions interact, with developments on the artifact motivating policy discussions, and policy constraints steering developer effort. With longitudinal data on both institutional and socio-technical variables, we now transition to a quantitative investigation of these relationships.

4.5.4 RQ$_3$: Are periods of increased Institutional Statements frequency followed by changes in the project organizational structure, and vice-versa?

In the previous RQs, we conducted exploratory and qualitative studies of the IS extraction technology, and of IS and socio-technical variable changes over time. In this section, we investigate the temporal relationship between our measures of institutional governance and organizational structure, as OSS projects progress on their incubation trajectories. As predicted by contingency theory, our hypothesis is that during project evolution, devel-

[12]`https://lists.apache.org/thread/twsg2h72d2025jc6nc1ltzgbmwoxh7cx`

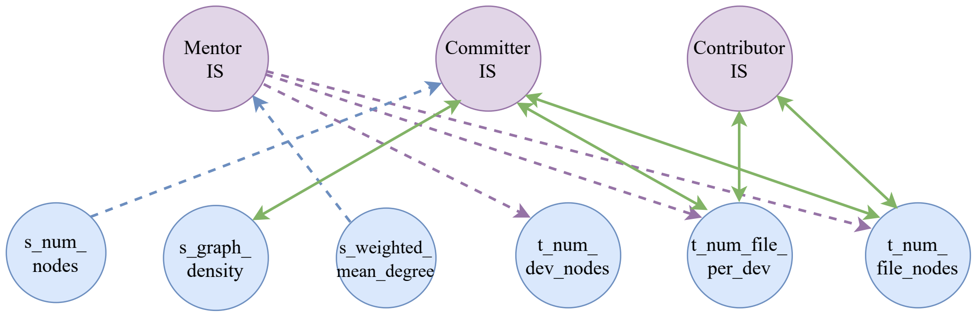
opers and mentors must make time for decisions related to their organizational structure, contingent on ASF-required institutional arrangements and governance. That is, incubating projects change their organizational structure based on the institutional norms and rules being discussed, as required of them as a potential new member of the ASF community. And vice versa, organizational changes can incite follow-up discussions about institutional processes. To test for $\mathbf{RQ}_3$, here we use the pair-wise Granger causality test with lagged order of 2. We run the test for all pairs between the institutional statements and socio-technical variables, resulting in 36 separate tests for the graduated projects set and 36 for the retired ones. We adjust our p-values for multiple hypothesis testing to control false discovery rate, using the Benjamini-Hochberg procedure [164]. We only consider significant with p-val < 0.001.

The results are summarized in Figure 4.4, where a directed edge from node X to node Y indicates that X Granger-causes Y, i.e., change in X is the precursor to the change in Y. Also, as discussed in Section 4.6, the Granger approach we used is not a complete test of causality, but does yield an effect and its directionality, although without effect size or sign.

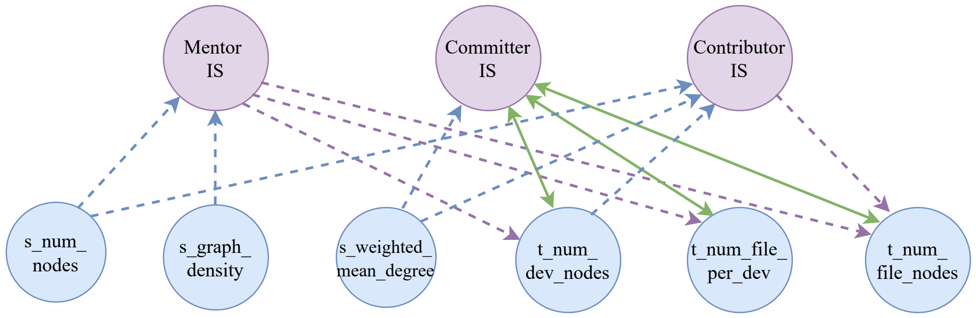
We observe a large number, 31 (out of 72 total), of Granger-causal relationship between the measures of institutional governance and the organizational structure. Of those 31 Granger-causal relationships, 15 are from the graduated set and 16 from the retired set, and 8 of the relationships are shared between the sets. We conclude that there is a significant Granger-causality between changes in institutional governance discussions and the organizational structure of the projects. We note 8 bidirectional relationships[13], the remaining 15 are unidirectional.

We look at graduated projects first. Interestingly, Figure 4.4, top, shows that the number of ISs from mentors, committers, and contributors has effects on the technical network, and vice-versa for the latter two. Namely, IS from all roles (mentors, committers, and contributors) Granger-cause changes in the technical networks, i.e., on developer file productivity (`t_num_files _per_dev`), and total number of coding files changed

[13]Bidirectional causality indicates feedback of some sort. E.g., supply causes demand, and demand in turn, causes supply.



(a) Granger Test for Graduated Projects ($p < .001$)



(b) Granger Test for Retired Projects ($p < .001$)

Figure 4.4. The Granger Causality between Institutional Statements and Socio-Technical networks. The blue/purple directed links indicate Granger causality from ST/IS measures, respectively. A green bi-directional link indicates that there is two-way significant temporal relationship (p-val < .001). Graduated projects seem to have fewer links from ST variables to IS variables, suggesting a more unidirectional flow from institutional to sociotechnical changes in successful projects (color online).

(`t_num_file_nodes`) variables. Mentor IS, additionally, Granger-cause changes to number of developers (`t_num_dev_nodes`). This is consistent with ASFI expectations that a mentor's emails provide advice and engage people, and conversely, that a drop in engagement may elicit mentors' engagement. Mentors usually do not code, which is presumably why they Granger-cause but do not appear in feedback relationships with any of the technical network variables.

Notably absent, however, are links from mentor and contributor ISs into social network variables. Only committer ISs (bidirectionally) Granger-cause changes in the social network density, which, perhaps, simply indicates that ISs from committers induce substantial traffics in the social network, which in turn gets committers to discuss policy and rules issues. We have observed situations where mentors are likely to interrupt the projects when the projects become less active (either socially or technically)[14]. On the other hand, it could also be that a mentor is reacting to some particular broader discussion among developers, e.g., one on a monthly report.

Together, the above tells a story of the importance to the technical networks of changes in any IS variable. Surprisingly, mentor IS changes are not as consequential to the social network, seemingly at odds with the ASF community-first goals. Thus, there may be room to enhance community engagement with mentors and vice-versa.

RQ$_3$ Summary: In both graduated and retired projects, there are no inputs from the IS into the social network variables, even though there are IS inputs into all technical network variables. Retired projects exhibit less bidirectionality between ST and IS variables. Finally, and interestingly, among retired projects, there are causal inputs into contributor ISs from both the social and technical variables. This is not the case for the graduated projects.

4.6 Discussion

In this study, we use individual institutional prescriptions, Institutional Statements (IS), and the Socio-Technical (ST) network features to reason about OSS project sustainability. OSS projects are a form of digital public goods which, like other public goods (e.g., water, forest, marine, etc.), can be subject to degradation due to over-harvesting, e.g., in the form of free-riders who take advantage of OSS but do not contribute to the required resources for development and maintenance of the software. Ostrom's work illuminated the fact that many communities avoid the dreaded 'Tragedy of the Commons', and other collective action problems, through the hard work of designing and implementing self-governing institutions. In that context, the ASF is a nonprofit foundation that, through its incubation program, encourages nascent OSS projects to follow some ASF-guided operational-level rules or policies around their self-governance. The OSS projects that join the ASF incubator trade some of the freedom of unlimited institutional choice in

[14]An example of mentor interrupting project *warble*: `https://lists.apache.org/thread/x6h8pzhmfwtyy354ml1xm9sylq4y5r7l`

exchange for incubator resources that increase their chances of enduring the collective action problems that characterize OSS development [145], and becoming sustainable in the long run.

We found that in the ASF Incubator, the amount of institutional statements and levels of socio-technical variables are associated with projects graduation outcome, suggesting that the measures of institutional governance and organizational structure can signal information on sustainability. In particular, in $\mathbf{RQ}_1$, the Mann-Whitney U test shows that the graduated projects have significantly more ISs from all three types of participants: committers, contributors, and project mentors than retired projects. This, presumably, is indicative of more active or intentional self-governance. In theoretical and empirical work on commons governance, it is well documented that getting self-governing institutions 'right' is hard work and takes time and effort [138]. This is consistent with a narrative that participants in graduated projects debate and work harder on their project's operational-level institutional design.

Recent work has shown that ASFI graduate and retired projects have sufficiently different socio-technical structures [22], so that graduation can be predicted early on in development at 85+% accuracy. The results in $\mathbf{RQ}_2$, show that, for the first 3 months of incubation, developer nodes in the social networks of graduated projects increase at a higher rate (means increase from 10.1 to 17.1, and from 7.3 to 9.1 for graduated and retired projects, respectively), suggesting graduated projects were able to keep developers contributing more actively or recruit more new members. On the other hand, for the first 3 months, we also found that the amount of Institutional Statements by mentors increases in graduated projects, and decreases in retired projects (from 19.7 to 22.7 vs 22.6 to 14.6, for graduated and retired projects, respectively), suggesting that the initial help from project's mentors is of importance.

To further study the effects of ISs, we performed a deep-dive into IS topics. We found the topics of institutional-relevance in the graduated projects differ from those of the retired projects, specifically, we find that the topic of documentation (topic 7) in graduated projects is more prevalent than in retired projects. On the other hand, we found that the topics of mentorship (topic 11) of retired projects are significantly higher than retired projects, signaling that the retired projects might be struggling during the incubation. Combined with the fact that there are more developer nodes in both the social and technical networks, together the findings suggest that graduated projects have more capacity and energy to attend to non-coding issues, like documentation, than retired projects do. However, even among graduated projects there is still diversity in the institutional statements. Thus, as predicted by contingency theory, as well as Ostrom's theory of institutional diversity [155], a one-size-fits-all solution to a successful trajectory toward sustainability is not likely. Instead, future work should focus on gathering larger corpora of data, to be able to resolve individual or small-group differences in sustainable projects.

Our framework allowed us to combine the IS and STS structures and study them together over time. With it, in $\mathbf{RQ}_3$, we found two-way, causal correlations between socio-technical variables and ISs over time, arguably indicating that OSS project socio-technical structure and their governance structure evolve together, as a coupled system. In addition, our methods point to a way to study possible interventions in underperforming projects. Specifically, the finding that in retired projects there are bi-directional links from committer's ISs to all three features of technical networks (i.e., $t\_num\_dev\_nodes$, $t\_num\_file\_per\_dev$, $t\_num\_file\_nodes$), suggest that increase in committer's IS are interleaved with changes in features of the socio-technical networks.

As for the design implications, in addition to the current categories of mailing lists in ASF incubator (e.g., 'commit', 'dev', 'user', etc.), there can be a benefit to creating a separate mailing list, for institutionally-related discussions to help committers (and also for mentors and contributors) participate faster in those discussions in a timely manner. This could be made more useful using technology for self-monitoring, with which project participants could monitor a project's digital traces and discussions in order to more quickly react to episodic events. Some such tools have already been created for socio-technical networks in ASFI projects [165], and could be extended to include ISs as well. Such tools can help identify entry points and targets for interventions, whereby underperforming projects could be leaned on, internally or externally, via rules or advice to adjust their trajectories.

Contributions to Institutional Analysis and Socio-technical System Theory. Making a full circle, our findings also point to ways in which the theories we started from can be refined or extended. We find, in Sect. 5.4, evidence that the features of OSS projects' socio-technical systems co-change together with the amount of Institutional Statements in them, and that the co-change relationships are sparse. This evidence of co-change implies that the OSS projects' structure and their governance form a (loosely) coupled system. From a controllability point of view, a dynamically coupled system refines Smith et al.'s mechanistic binary notion of 'inside' and 'outside' interventions [121].

Our findings also suggest that for OSS projects, adopting additional rules and norms (e.g., by joining ASFI) can be worth the loss of some freedoms, as the Institutional Statements (Sect. 5.2, 5.3, 5.4) seem to serve to organize the project's actions and discussions, as predicted by Siddiki et al. [150] and Crawford and Ostrom [149]. Thus, our findings tie in with, and potentially extend the Institutional Analysis Design (IAD) view, suggesting that the feedback between the socio-technical system structure and institutional governance analysis is sufficiently direct and significant, and should be considered unitary in further studies.

More practically, our institutional statement predictor, although still a work in progress, can effectively predict atomic elements of self-governance. As such, it can be used as a tool to provide quantitative data for applying institutional analysis and design (IAD) more generally, e.g., to OSS projects that are outside of ASF, or self-governed systems with public documents and discussion forums.

4.7 Threats to Validity

First, our data is from only hundreds of projects ASF incubator projects. Thus, generalizing the implications beyond ASF, or even beyond the ASF Incubator projects carries potential risks, for example, OSS projects in other incubator programs may not have mentors. Expanding the dataset beyond the ASF incubator, e.g., with additional projects from other OSS incubator programs could lower this risk. Second, we do not consider communication channels other than the ASF mailing lists, e.g., in-person meetings, website documentation, private emails, etc. However, ASF mandates the use of the public mailing lists for most project discussions, a policy that ensures a particularly low risk of missing institutional or socio-technical information [15]. Annotations of the Institutional Statements (IS) can be biased by individual annotators, while we gave the annotators sufficient training and reference documentation which lowers the risk. We expect the performance of the classifier as we increase the size of the training set and better incorporate contextual information, and we plan to distinguish types of ISs for future work. In OSS projects, developers may use their different emails or aliases, which in turn complicates the identification of distinct developers, while assigning and insisting on using a unique apache.org domain email address reduces such risks [16]. Finally, as noted in Sect. 4, there are likely cases where OSS projects that have retired from the ASF Incubator program still go on to become sustained over time. In these instances, some OSS projects entering the ASFI may simply not be a good fit for the ASF culture and institutional requirements or policies and ultimately retire as a result. In this paper, we explicitly use graduation as a measure of sustainability given that this is an ultimate goal of the ASFI – to create projects that can indeed be sustainable. But we want to recognize the point that few retired projects still could become sustainable by following a different path than association with ASF.

4.8 Conclusion

Understanding why OSS projects cannot meet the expectations of nonprofit foundations may help others improve their individual practice, organizational management, and institutional structure. More importantly, understanding the relationship between institutional design and socio-technical aspects in OSS can bring insights into the potential sustainability of such projects. Here we showed that quantitative network science features can capture the organizational structure of how developers collaborate and commu-

[15]The Apache Way: `http://theapacheway.com/on-list/`
[16]ASF committer emails: `https://infra.apache.org/committer-email.html`

nicate through the artifacts they create. Combining the two perspectives, socio-technical measures, and institutional analysis, we leverage the unique affordances of the Apache Software Foundation's OSS Incubator project to extend the modeling of OSS project sustainability, leveraging a novel longitudinal dataset, a vast text and log corpus, and extrinsic labels for the success and failure of project sustainability.

Chapter 5

On the Self-Governance and Episodic Changes in Apache Incubator Projects: An Empirical Study

5.1 Introduction

Sustainable Open Source Software (OSS) projects are characterized by volunteering work, continuous recruitment, and effective governance. However, even OSS projects that are widely used by many, including large companies and national governments, may not attract the attention and resources they need, resulting in unsustainable development and potentially severe consequences downstream [166, 167]. E.g., on December 9th, 2021, the Apache *Log4j* project was reported to have a severe security vulnerability, likely due to being severely under-resourced, affecting countless individuals and organizations. One day later, US government officials assigned the highest severity score for the *Log4j* incident [1].

While not all OSS projects that are unsustainable will have such drastic consequences, many of them that depart from a sustainable trajectory suffer the 'tragedy of the commons' or even get abandoned. This is due to a complex set of circumstances making it challenging to pinpoint and mitigate. On the one hand, software engineering researchers have favored a socio-technical perspective of OSS projects, using email communication and code commits to build socio-technical representations [33, 115, 168]. On the other

[1]Log4j incident post:`https://nvd.nist.gov/vuln/detail/CVE-2021-44228`

hand, management science researchers have studied episodic changes in organizations and sustainability for governing the commons, including forests, marine, and fisheries, through the lens of institutional written or unwritten norms, rules, and regulations [169, 170]. By and large, these two perspectives on sustainability have not been fruitfully combined, and very little is known about how effective governance and software development work together in practice toward sustaining OSS projects. But we do have an example that clearly demonstrates how changes in self-governance can change the trajectory of project sustainability.

<u>Motivating Example</u> In Dec 2018, the Apache Software Foundation incubating project OPENWHISK was experiencing a hard time: the number of active developers dropped from 35 to 17 in only a few months. Had such a downturn continued, the project could have gotten abandoned and finally retired from the ASF incubator. Fortunately, a developer started an email thread, noting that there had been insufficient engagement in running the project: '*I've done the release manager role for package release so far, but to me it seems that our release process is being impeded by a lack of engagement from eligible voters on the IPMC mailing list.*' This email thread acted to incentivize others to get positively engaged '*I will reflect this in the quarterly report ... The IPMC is aware of the issue and is currently doing something about that.*' In the months that followed, the project was discussing how to get back on the sustainable trajectory, and finally decided to continue its development by enforcing regulations. '*I would like to see us push out a consolidated next release in the near future (by end of January?). I'd also like to see us attempt to establish a regular cadence of such consolidated releases (perhaps quarterly?)*'. During this change event, the number of developers grew from 17 to 28; so did the commits and engagement on the mailing list. Later, the project graduated from the incubator in 2019.

We see in the above that discussions related to norms, rules, and regulations can trigger changes in the software development process, leading to corrections in the OSS project trajectories. We also suspect that OPENWHISK is not the only such example. Prior work has shown that many ASF incubator projects experienced large changes in their socio-technical structure, over short periods of time [22]. Projects with effective governance are more likely to come out unscathed from such large changes, while others do not, implying that governance may be a catalyst to sustainability [28].

Inspired by the above example and prior work, our hypothesis is that concerted institutional discussions can lead to large changes in the underlying socio-technical structure. Symmetrically, changes in the projects' socio-technical structure may require modified rules or institutional governance to be compatible with the new structure. To validate our hypothesis, in this work we chose a set of 262 Apache Software Foundation (ASF) incubator projects. The ASF incubator is a well-known pioneer and champion for open source. It hosts hundreds of OSS projects, striving to nurture sustainable communities in each project through ASF-wide mechanisms, including a set of institutional policies and governance. When a project exits the incubator, each project is evaluated and labeled as graduated (sustainable) or retired (unsustainable) by ASF committees. Such an extrinsic labeling is essential to understanding sustainability. Just as importantly, the openness and completeness of the ASF mailing lists (ASF's tenet is 'If it didn't happen on the mailing list, it didn't happen'), makes the ASF incubator a key resource for studying OSS sustainability from both the socio-technical, software engineering and the institutional governance perspectives.

Starting from a data set of social, technical, and policy digital traces from 262 sustainability-labeled ASF incubator projects, and guided by related social and organizational theories, here we sought to study the more specific hypothesis:

> Sustainable projects can process and translate self-governance rules and policies into socio-technical changes, and vice versa, more effectively than unsustainable projects.

To that end, we employ a large-scale empirical study to characterize sustainable and unsustainable projects by matching episodic changes in their socio-technical structure to evidence of institutional discussions. We operationalize this by matching episodic time-series events, i.e., Change Intervals (CI) in the socio-technical structure, to sentence-level institutional discussions, i.e., Institutional Statements (IS), as well as the temporal relationships between them. We develop a framework for simultaneous, socio-technical and institutional, analysis of OSS projects, with a view to describing and understanding a process affected by both, namely, projects gaining self-sustainability and self-government and eventually graduating from the ASF incubator. Our findings are as follows:

- We can effectively identify episodic Change Intervals (CI) in the socio-technical structure, and they tend to be temporally co-located with Institutional Statements (IS);
- CIs have effects at both individual-level and project-level, and such effects vary across both agents and projects with different levels of sustainability;
- During episodic changes, sustainable (i.e., ASF incubator graduated) projects can convert institutional rules into practice more efficiently than other projects.

To the best of our knowledge, this work is among the first to attempt to study the structural changes in OSS projects and their self-governance under a unified analytical framework. We are hopeful that refining this convergent approach, of socio-technical and institutional analyses, will lead to new ways of thinking about and analyzing emergent properties in modern software engineering such as OSS sustainability.

5.2 Background and Theoretical Framework

This section introduces the background and social theories pertinent to OSS governance and sustainability.

5.2.1 Theory of Governing the Commons

A major portion of Ostrom's Nobel Prize-winning work [140] investigated how individuals collaborate and create self-governing institutions in natural resource settings [171], e.g., water [141], marine [142], and forest [24]. However, in practice, individuals who cannot be easily excluded from the use of shared natural resources often have little incentive to contribute to the production or maintenance of these resources [143, 172]. This refers to the 'Tragedy of the Commons' [173], and these individuals are often referred to as free-riders in natural resource commons settings [143, 172].

In the OSS context, OSS code is clearly an inexhaustible resource to users: it can be copied over and over. However, there are exhaustible (limited) resources in OSS commons, e.g., developer's efforts. But there is also maintenance that is regularly needed, e.g., on defects, technical debt, etc. The combination of limited developer effort available and the need to keep technical debt low produces situations similar to the tragedy of the commons, since developers find building new features more rewarding than performing code maintenance or fixing bugs in OSS projects. In that sense, OSS free-riders, would be those favoring feature development over fixing bugs.

Over the course of a lifetime, Ostrom demonstrated through hard work in the development of self-governing institutions that communities can avert such tragedy [140]. This was accomplished primarily through the introduction and evolution of the Institutional Analysis and Development (IAD) framework [138]. We and others before us have realized Ostrom's formalism is appropriate for analyzing OSS commons [147, 146, 82], where exogenous factors are: the socio-technical context as community attributes, ASF's and project-specific regulations as the rules-in-use, and the biophysical conditions correspond to software artifacts being developed. The action arena consists of OSS contributors and action situations. Of course, sometimes the concepts fit very well, and other times, as in our answer about free-riders above, the concept matching is more distant, and that's where our current and future work lies: in extending IAD and Ostrom's rules to OSS ecosystems. However, even if over-appropriation may not be a problem for OSS, the tragedy of the commons can still happen in the OSS context, and OSS sustainability lies at the core of the solution. Such tragedy arises when there are free-riders who do not provide sufficient work on development and maintenance while taking the spot, therefore, the project cannot achieve the functionality and use intended, and thereafter becomes abandoned [145].

5.2.2 Organizational Change Theory

One can gain a more comprehensive understanding of the nuances of organizational change through the interaction between different perspectives, because every theoretical perspective provides a partial account of a complex phenomenon [174]. Here we present three main pillars of organizational change theory.

Episodic Change Organizational changes are viewed as episodic changes when they occur infrequently, discontinuously, and intentionally [175]. Organizations tend to undergo episodic change during periods of divergence when they move away from their equilibrium condition [174]. Developing divergence results from a growing misalignment between an inertial deep structure and perceived environmental demands.

Agents During organizational change, influencers who are committed to change are viewed as change agents. With their charisma and fortitude, such agents motivate and lead their teams by engaging them in the change process. These kinds of leadership pedigrees can be found in two types [176]: The first type of agent uses power as a means of rewarding and sanctioning their staff. The second type of agent has the trust of their staff, and in these cases charismatic agents can successfully influence others to follow their commands.

Resistance In effect, episodic changes in an organization can cause employees' resistance in the workplace [177]. Piderit et. al. [178] claim that people tend to stay unchanged in their workspace, as their primary responsibility might be the welfare of their families. Therefore any organizational change that is going to impact that reality is going to encounter some kind of resistance if the employees are not involved in the change process. As such, successful organizational adaptation is increasingly reliant on generating employee support and enthusiasm for proposed changes, rather than merely overcoming resistance [178].

5.2.3 Socio-Technical System Theory

OSS projects, and the socio-technical side of software engineering in general, have dominated the analysis for a long time through organizational and socio-technical perspectives [148, 8, 179, 180]. Social-technical systems (STS) consist of two main components: the social part, where users continuously create and share knowledge by engaging in various kinds of interactions with one another [181], and the technical part, where they rely on technical hardware to accomplish collective tasks [117, 182, 17]. STS is typically referred to when examining how a technical system is able to provide efficient and reliable interaction between individuals. In addition, it examines how the social subsystem is affected by interactions and therefore influences the performance of the technical system [118, 119]. One might also describe STS as an intermediary entity that transfers institutional influence to individuals, combining the views of engineers and social scientists [154]. Xuan et al. [183] propose a method to measure the interleaving effects between email communications and code commits in OSS projects, and they find that bursts in communications before and after code commits are essential for effective software development. From the STS perspective, the ASF community is a unique system that has both outside influence regulations from the ASF board and members and inside structure managed or self-governed committees.

5.3 Research Questions

In the previous section, we reported that a variety of scholars have utilized a socio-technical approach to analyze complex collective behaviors in OSS projects. We also described how institutional analysis is useful in understanding institutional governance under the context of digital commons (i.e., OSS projects). Moreover, prior work has shown that episodic changes to the socio-technical network can be indicators of changes in project sustainability [22]. As a first step to linking such changes to their antecedents in institutional discussions, we focus on the methodology for identifying large changes in socio-technical features, over time. We ask:

RQ$_1$: Are there episodic changes in the socio-technical structure of projects during their incubation? Likewise, can we identify discussions related to policy and governance?

As predicted by organizational change theory, episodic changes are associated with agents and will be reflected in some form of resistance (e.g., negative developers' engagement, responsiveness, and sentiment). We ask:

RQ$_2$: Is there significant resistance evident in policy/governance-related discussions associated with episodic change? How do they differ between sustainable projects and others?

Per institutional analysis theory, strategies, norms, and rules can affect the socio-technical structure of projects. In addition, institutional governance and organizational structure must work hand-in-hand to make viable socio-technical systems. Ill-designed institutional arrangements can introduce inefficiencies into the system, which may amplify non-standard behavior and structure. In a sustainable system, an ill-formed organizational structure may induce new rules to adjust and improve such structure, improving efficiencies in the systems. Therefore, such influential links from institutional design to the organizational structure can be, in fact, bi-directional. Most such changes will motivate some and demotivate other developers, which will manifest as variable sentiment in their communication.

Thus, we hypothesize that the feedback loop, if any, between institutional governance and organizational structure can be quantitatively measured and associated with overall sentiment. We ask:

RQ$_3$: What are the associations between episodic change direction (i.e., up-turns and down-turns) in socio-technical structure and the sentiment in IS-related discussions?

In the following section, we introduce the methodologies approaching the above three research questions.

5.4 Data and Methods

In this work, we leverage a previously published data set [128] consisting of hundreds of Apache Software Foundation Incubator projects. ASF Incubator (ASFI) aims to help projects become self-sustaining and eventually join ASF [6]. The incubation outcome is two-fold: One is graduation indicating that the project has a self-sustainable community to move it forward, otherwise the project is retired.

On the data end, in addition to the previously published data set, we gather complementary time-stamped trace data of commits and emails using PERCEVAL [184]. To reduce the noise caused by outliers in email data, we removed bots' automated emails by applying regular expressions to email titles and content. Similarly, for commit data, we use *GitHub Linguist* and identify 731 collective programming language and markup file extensions to remove non-coding commits (e.g., committed to files with extension .json,

.jpg, .png, etc.). Our final data contains 262 projects, among them, 205 are graduated projects, and 57 projects are retired. In total, we collect 1,548,807 email records from 42,191 unique emails contributors, and 359,297 commit records from 5,931 unique ASF committers [2].

5.4.1 Constructing Socio-technical Networks

Studies of complex systems, such as OSS projects, have largely relied on network science approaches [131, 111, 185]. As socio-technical networks can contain both information about the components (i.e., the nodes) and the interactions between the components (i.e., the edges), we use them here as abstraction anchors. In this work, the socio-technical networks consist of two types of networks [15]: social networks, which are extracted from their email communications, and technical networks, based on commits to source files. At each month in incubation, for each project, we form social networks (weighted directed graphs) from the communications between developers as follows: developer A has a directed edge to developer B only if B has replied to A's post on the mailing lists during that month. The edge weight represents the frequency of communication between two developers. The technical weighted bipartite graph is formed in a similar way. We include an undirected edge between developer A and a source file F if developer A has committed to file F during that month. Each edge is weighted according to the frequency at which it is committed to the source file. We use the *Python* NETWORKX package for the network implementation.

5.4.2 Identifying Institutional Statements (IS)

Through the lens of open-access email discussions among ASF committers, ASF mentors, and other types of contributors, we can then capture their institutional designs in the form of ISs.

Definition of Institutional Statement (IS). We refer to a sentence-level institutional discussion as an *Institutional Statements* (IS). For example, on 24 Feb 2017, an ASF incubator project Airflow sent out an email containing institutional statements "*Next*

[2]Our code is available at Zenodo: https://doi.org/10.5281/zenodo.6526833

steps: 1) will start the voting process at the IPMC mailing list. ... So, we might end up with changes to stable. ... 2) Only after the positive voting on the IPMC and finalisation I will rebrand the RC to Release." In short, norms, rules, and strategies are outlined as prescriptions and constraints that mobilize and organize actors for collective action in a form of institutional statements. To extract IS from the email corpus, we leverage previous work on institutional analysis [28]. As there is no ground truth for institutional statements to train the IS classifier, they first hand-annotated a small subset of the data for IS as follows. Using a random subset of 313 email threads from incubator project lists, two coders classified each sentence in them as either 'IS' or 'Not IS' according to whether it came from an institutional statement or not, which results in 6,805 labeled sentences (i.e., 'IS' or 'Not IS'), and there were 273 of them labeled as IS [3]. They combined the email exchange data set to fine-tune a BERT-based classifier [157], for automatic detection of ISs. In the end, given the fact that ISs are rare (there are only about 5% emails contain ISs), and the task naturally is challenging, the classifier achieved a precision score of 0.667, recall score of 0.681, and F1 score of 0.674 on classifying institutional statements, showing the classifier is able to extract ISs from developer email exchanges.

5.4.3 Identifying Change Intervals (CI)

Organizational changes are categorized as episodic changes when they occur infrequently, discontinuously, and intentionally. When studying the dynamics of socio-technical systems, prominent changes in socio-technical network variables that intuitively mark critical events are particularly relevant.

Definition of Change Intervals (CI). We refer to the time periods (in months) during which these episodic changes occur as *Change Intervals* (CI). We use the Cumulative Sum (CuSum) algorithm to detect these change intervals with the package DETECTA. CuSum algorithm is a widely used method for monitoring abrupt changes in time-series data [186]. A typical form of CuSum algorithm is to calculate the cumulative sums in positive and negative directions along an axis of the data and mark an alarm point when reaching some threshold c. DETECTA can extract the increasing/decreasing change interval containing

[3]Coding manual: https://doi.org/10.5281/zenodo.7042616

Table 5.1. Definition of variables.

Type	Variable	Definition
Social-technical	`s_num_nodes`	The number of unique active developers in the social network.
	`s_num_component`	The total number of disconnected components in the social network.
	`s_graph_density`	The number of existing edges divided by all possible edges
	`s_avg_clustering_coef`	The ratio of closed triplets divided by all triplets.
	`s_weighted_mean_degree`	The mean degree of the social network.
	`t_num_dev_nodes`	The number of unique developers in the technical network.
	`t_num_file_nodes`	The number of unique source code files in the technical network.
	`t_num_dev_per_file`	The number of developers per file node.
	`t_num_file_per_dev`	The number of files per developer node.
	`t_graph_density`	The number of existing edges divided by all possible edges
Institutional:Agents	`Mentor`	Person who mentors projects and helps them grow a sustainable community.
	`Committer`	Person who commits and reviews code changes.
	`Contributor`	Person who contributes through non-coding activities.
Institutional:Resistance	`Responsiveness`	The average delay time (in days) for agents to reply to IS-related emails.
	`Engagement`	The average number of emails that an agent engages in.
	`Negativity`	The number of negative emails over all emails for each agent.

an alarm point. It also uses a drift parameter, denoted by d, to penalize a long, flat drift. Our parameter selection procedure on c and d respects the diverse properties of different projects. Specifically, for a socio-technical variable, each project gets its unique pair of c and d, calculated based on the project's fluctuation level. In the procedure, we first calculated the pairwise differences ($|x_i - x_j|$, where x_i is the data point at time $t = i$) in the data for each project. Then, within each project, we took the ratio between the mean μ^* of the largest 20% of the pairwise differences and the overall mean μ, denoted by $p = \frac{\mu^*}{\mu}$. This ratio p from all projects formed a distribution P which provides a comprehensive view of the extent to which large changes in these projects outstrip project averages. From distribution P, we select a value for the base parameter p_0, which is used to generate unique c and d for each project, with the equations: $c = p_0\mu$ and $d = 0.1c$. To restrict the *Type I* error rate, we conservatively set $p_0 = P_{0.75}$, where $P_{0.75}$ is the 75^{th} percentile of P. We demonstrate an exemplary CI in Figure 5.1.

5.4.4 Variables of Interest

§ Socio-technical Recent work shows that the network modeling is exhibiting high predictive power for OSS success and sustainability [8, 187, 188]. Our socio-technical network variables are pulled from a recent study on forecasting the sustainability of OSS projects [22], All metrics are aggregated on a monthly basis for each project. In total, we have ten socio-technical network variables. The first five are in the social network: (1) number of nodes `s_num_nodes` indicates the unique active developers in social networks; (2) average clustering coefficient `s_avg_clustering_coef` describes the linkage of a node to its neighbors, measured by the closed triplets divided by all triplets; (3) number of components `s_num_component` is the total number of disconnected components in the social networks; (4) weighted mean degree `s_weighted_mean_degree` represents the mean degree of the social networks; (5) graph density `s_graph_density` measures the density of the network, calculated as the number of existing edges divided by the number of all possible edges; And the other five variables in the technical network are: (6) number of developer nodes `t_num_dev_nodes` is the number of unique developers in the technical networks; (7) number of file nodes: `t_num_file_nodes` is the number of unique coding files; (8) number of developers per file node `t_num_dev_per_file` measures the degree of collaborative behaviors; (9) number of files per developer node `t_num_file_per_dev` describes the degree of multitasking behaviors; (10) graph density `t_graph_density` represents the density of the network, calculated as the number of existing edges divided by the number of all possible edges in the technical networks.

§ Institutional In addition to institutional discussions, we define the variables indicating agents and resistance during episodic changes. First, the change agents are categorized into three classes below: (a) Mentors, who give mentorship to projects and help them grow and build their community toward sustainability. Mentors may intervene in the projects if the projects are not progressing well. (b) Committers, who are the major component of the workers for building the artifact. In a project, an individual becomes a committer until they make their first actual code commit to the code base. (c) Contributors, who present the largest population in the community. Contributors are the individuals

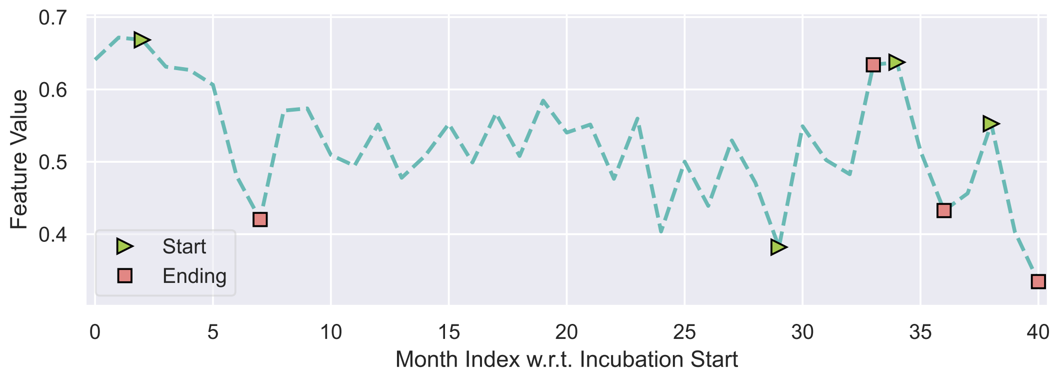


Figure 5.1. An illustration of a change interval (CI).

who are neither mentors nor the committer, i.e., they do not contribute code changes nor mentor the projects. However, contributors are essential to OSS sustainability. They can be helpful in the sense of non-code contributions, e.g., writing documentation, testing, and providing feedback. As predicted by the institutional change theory, during episodic changes, certain resistance may occur in the organization. Resistance measures are calculated on a monthly basis. (a) Responsiveness. Responsiveness is the first level of resistance by slowing down their work pace. It is measured by the average delay time (in days) when agents reply to previous IS-related emails. (b) Engagement. Engagement is the second level of resistance by not participating in certain discussions. It is calculated by the average number of emails the agent engaged in. (c) Negativity. Negativity is the third level of resistance and it carries the opposite information to the discussion. It is measured by calculating the number of negative emails over all emails. Since the negative content is much zero-inflated, we only look at the top 20% negative periods in our data, and use the Mann-Whitney U test to test the shift in means. We summarize the above variables in Table 5.1

5.4.5 Sentiment Detection

To detect the sentiment in institutional discussions, we use the state-of-the-art NLP model from package PYSENTIMIENTO to extract opinions from texts [189], which first came out in 2021. The base model is BERTweet, a RoBERTa model [190] trained on tweets, which is designed to handle sentiment and emotion analysis tasks in social discussions [191].

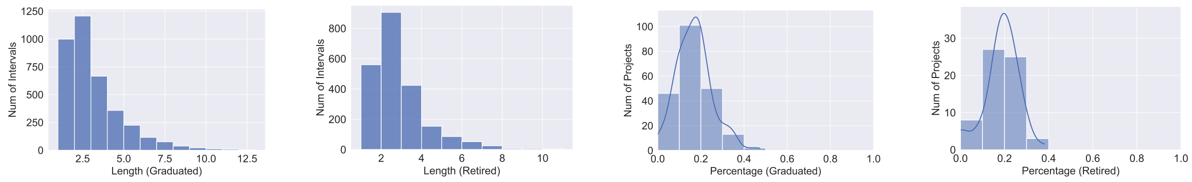


Figure 5.2. The distribution of the change interval duration (in months) and percentage w.r.t. projects' incubation time averaged across all socio-technical variables.

In our setting, the task is to extract sentiment from the discussions about institutional statements, we find that such a task is suitable to use BERTweet since the institutional statements are similar to open discussions in tweets. Previous work, Senti4SD [192], aims to address sentences like 'kill this process' under code-mixed software engineering context. Despite the fact that Senti4SD is trained within software engineering context. We believe that the BERT-based PYSENTIMIENTO model is, arguably, more suitable for our task, as discussions about institutional are more akin to social discussions rather than technical discussions with code snippets. To reduce the noise in the sentiment data, we only keep informative replies that are classified as either positive or negative.

5.5 Results

In this section, we study the change in OSS projects by adopting a unified framework, from the institutional analysis and socio-technical system perspectives.

5.5.1 RQ$_1$: Are there episodic changes in socio-technical structure during project incubation? Likewise, can we identify institutional discussions?

As described in Sect. IV, we used a change interval detection model to help us identify events in the socio-technical structure of ASF incubator projects. Here, we present descriptive statistics about the change intervals we detected.

In Figure 2, we show the total number of change intervals across all socio-technical structures per project. We aggregate the projects by their sustainability status accordingly (i.e., graduated or retired). We find that graduated projects tend to have a higher value in the total change intervals, suggesting that the development of graduated projects is more

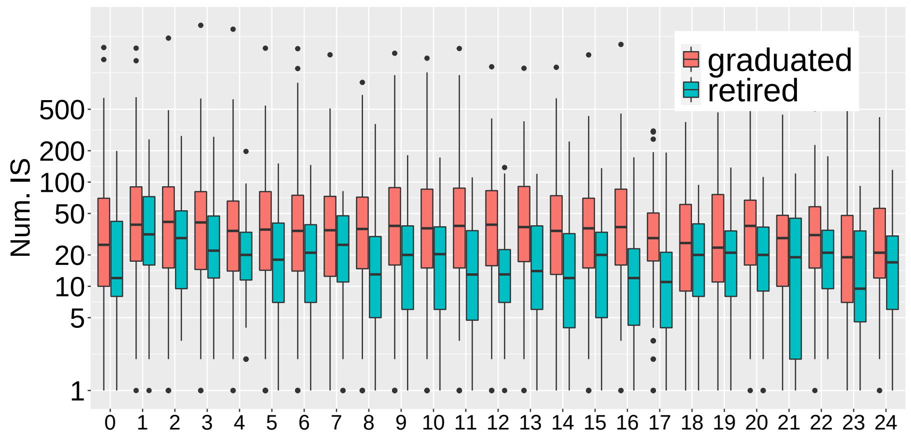


Figure 5.3. The monthly num. of institutional statements by sustainability status.

fluctuating than the retired projects. The total length of the change intervals is averaged by each socio-technical variable per project. We find that, for most of the change intervals, they tend to occupy 20% of the project incubation time, while the change intervals in retired projects have around 20% to 30% for their incubation time.

On the other hand, we are also interested in the stats from our IS classifier. As we can see in Figure 5.3, the number of monthly IS of graduated projects is more than retired ones'. Moreover, the number of IS from retired projects has a decreasing trend from month 2 to month 6 while for graduated projects it is more stable from month 1 to 12.

We have shown descriptive statistics about the detected change intervals, in the following sections we will show how to use the CI and IS more practically.

RQ$_1$ Summary: We showed that most change intervals tend to occupy about 20% of the projects' incubation time. We demonstrated that graduated (i.e., sustainable) projects have more institutional statements and shorter change intervals than retired (i.e., unsustainable) projects.

5.5.2 RQ$_2$: Are there significant resistances in IS-related discussions associated with episodic change? How do such temporal patterns differ across graduated and retired projects?

As predicted by institutional change theory, during an episodic change in the project's organizational structure, the OSS volunteers may incur an extra workload. In addition, different agents have varying levels of importance during episodic change, and in reverse, they are influenced by episodic change differently, i.e. resistance to change. In this section, to study the association between institutional statements and episodic change intervals, we dive deeper into three aspects at the project level: responsiveness, engagement, and negativity with respect to the change interval. Together, they represent three different levels of observable resistance in projects. Here, we only measure such change regarding IS-related discussions, that is, only the email exchanges containing IS are included. The three resistance measures are defined in Sect. 2.4.2.

We show the response delay over three periods, i.e., one month before the episodic change (pre-CI), during the episodic change (within-CI), and one month after the episodic change (post-CI) in Figure 5.4. All values are normalized as per month and aggregated by groups. The result of the Mann-Whitney U test suggests that there exists a significant increase in the means of response delay (in days) from pre-CI to within-CI, and from within-CI to post-CI periods, for both contributors and mentors, with p-value ¡.001. We find that the shift in committers' responsiveness is much lower and insignificant from pre-CI to within-CI period than contributors and mentors, suggesting the episodic change has a more significant effect on contributors and mentors, rather than committers, in terms of responsiveness. A possible reason for this phenomenon is the fact that committers are required to maintain a consistent commitment to development even during episodic changes.

Next, we look at the second level of resistance, engagement. As a reminder, the engagement is quantified by the number of IS-related emails for each type of agent engaged per month. The engagement indicates the level of participation in institutional issues (e.g., regulation, rules, and norms) within the projects. Unlike the responsiveness index, agents

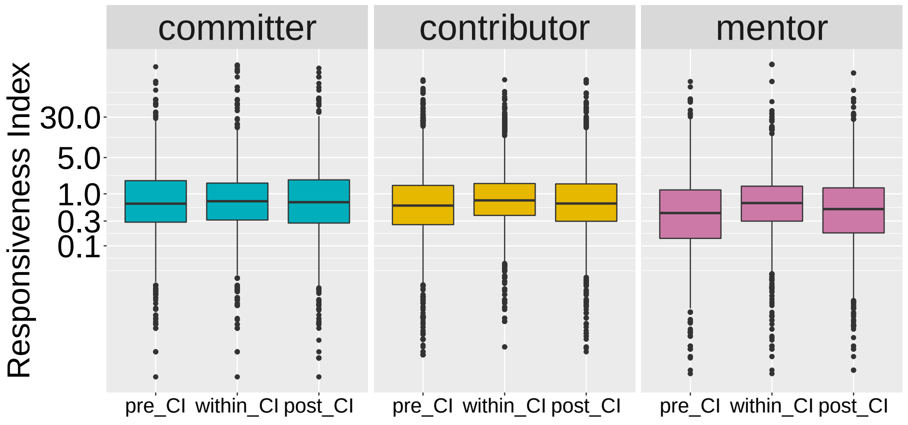


Figure 5.4. Responsiveness over pre-, within-, post-CI periods.

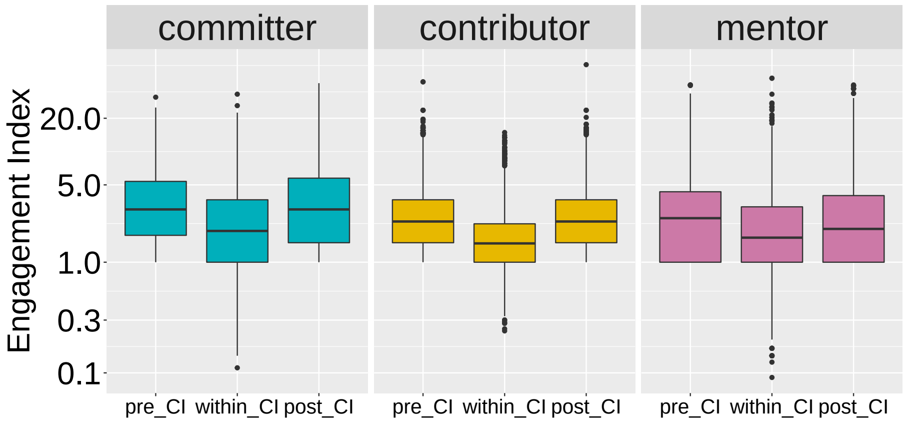


Figure 5.5. Engagement over pre-, within-, and post-CI periods.

can refuse to engage in certain IS-related issues during an episodic change period, and such reactions, if significant, should be observable via statistical analysis. As shown in Figure 5.5, the Mann-Whitney U test suggests that the engagement of all three types of agents in IS-related discussions is significantly reduced during the episodic change with p-values ¡.001. Combined with the results of responsiveness in Figure 5.4, we consider one possible reason for this to be that committers are not slowing down because they are more focused on specific types of issues, allowing them to remain an almost constant response rate in those. On the other hand, mentors are the delegates of regulations in projects but not the stack-holder, therefore, they only need to attend to certain IS-related issues if projects are progressing well.

For the last attribute, we look at the negativity. The negativity index is calculated for

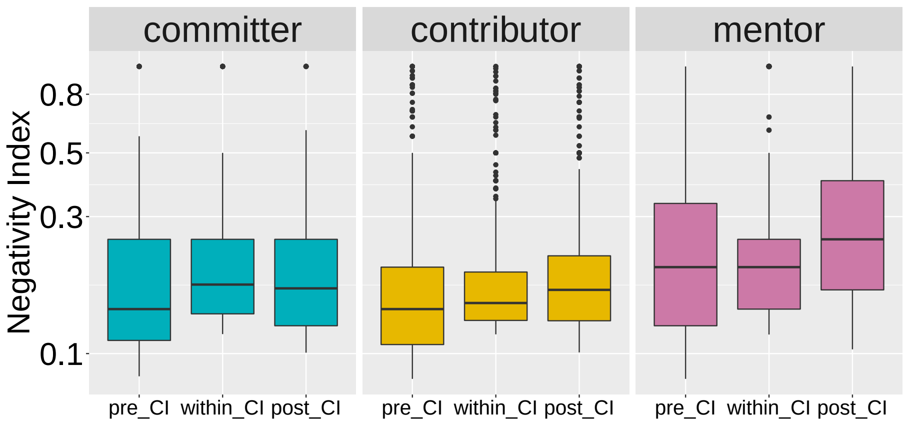


Figure 5.6. Negativity over pre-, within-, and post-CI periods.

all types of agents by the number of negative emails over all emails on a monthly basis. We measure how much agents oppose IS-related issues before, during, and after episodic changes. As shown in Figure 5.6, we measure such changes in the pre-, within-, and post-change interval periods. We find that three agents exhibit three different patterns in the negativity measure. Committers have a significant negativity increase between pre-CI and within-CI period, as suggested by the Mann-Whitney U test with *p*-value ¡.001. Moreover, we find that the mentors tend to be, in general, more negative than both committers and contributors regarding IS-related issues. In addition, contributors have steadily increasing shifts in negativity and much more outliers, while mentors have a significant increase in negativity only from within-CI to the post-CI period. There are the following possible reasons for this phenomenon: (1) Committers are the stakeholders, and they are the ones that will follow the regulations and norms, imposing episodic changes will shift them to be more negative on discussing IS-related issues. (2) Contributors are more of a customer of OSS rather than a producer, they are the less conservative people in the projects, and they are more likely to complain about regulations and rules they do not want to have, therefore, forming outliers. (3) Mentors' criticism is followed by the episodic changes, while during the episodic changes they tend to be more supportive of the team and help the community go through the changes.

In addition to time intervals around episodic changes, we also study the influence of episodic changes vary across projects with different sustainability levels, i.e., graduated

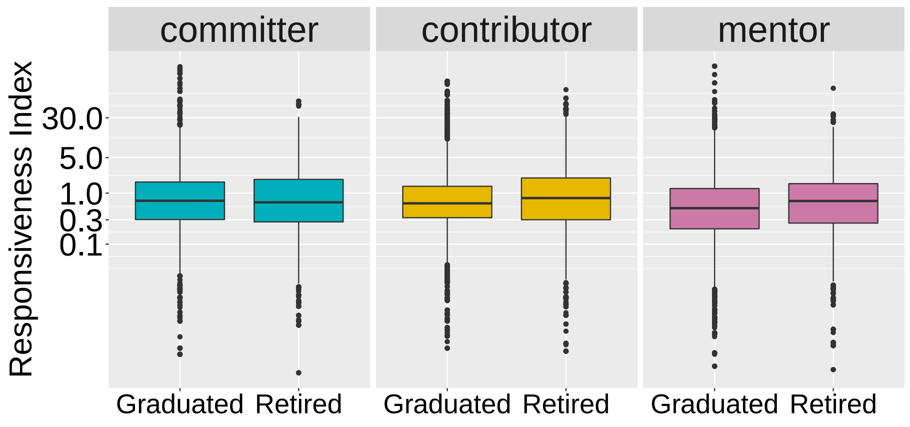


Figure 5.7. Responsiveness over graduated and retired projects.

and retired projects. As shown in Figure 5.7, for responsiveness, we find that there are significant differences in means between graduated projects and retired projects in contributors and mentors, but not committers, with *p*-value ¡.001. The possible reason for this phenomenon is the fact that ASF committers in both graduated and retired projects are almost equally responsive to issues. In addition, graduated projects have a more diverse community that can respond to issues more promptly, while retired projects may have a limited size of community members who work from the same time zone.

Then, we show that there exists a significant difference in committers' and contributors' engagement between graduated and retired projects (*p*-value ¡.001), as shown in Figure 5.8. The committers, as expected, have a higher level of engagement (the means in graduated and retired projects are 4.47 and 3.58, respectively), suggesting that developers in graduated projects are more interested in discussions regarding community building on institutional governance.

As shown in Figure 5.9, we find that both committers and contributors in retired projects are significantly more negative to IS-related discussions as compared to graduated projects. It suggests that in the retired projects, there is no capacity for people to move their projects forward regarding building up regulations and rules, i.e., committers in retired projects are 'burn-out' when they keep up the same response rate in graduated projects.

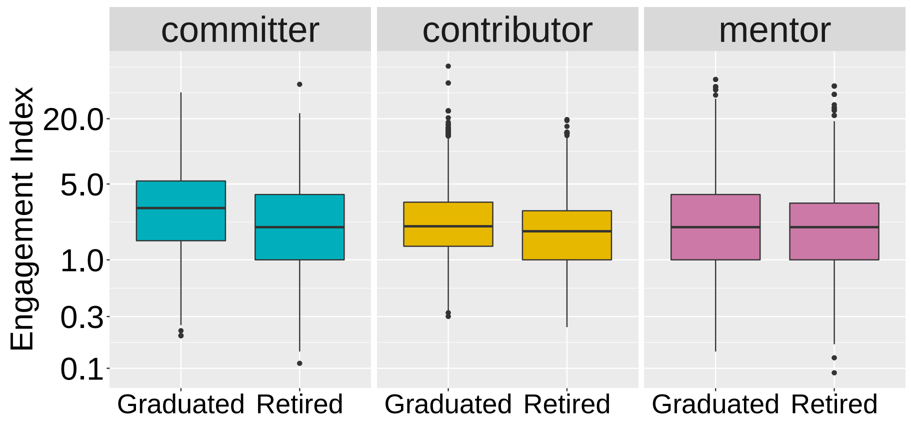


Figure 5.8. Engagement over graduated and retired projects.

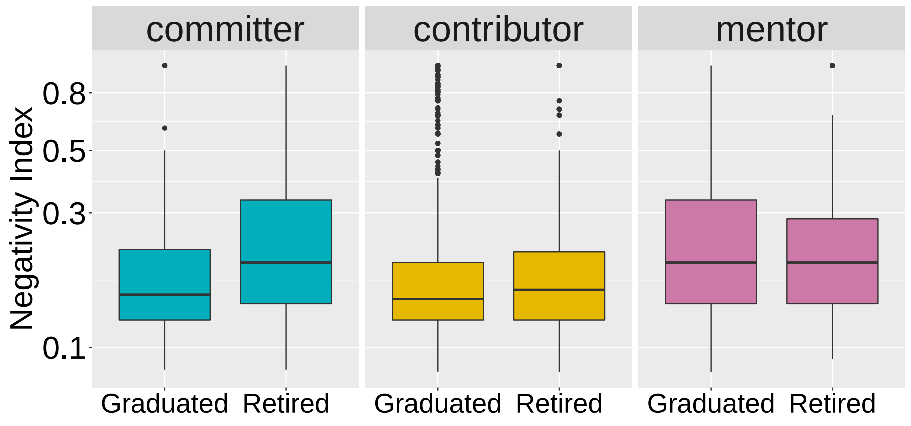


Figure 5.9. Negativity over graduated and retired projects.

RQ$_2$ Summary: We find that, around change intervals, there exist certain resistances, and they vary across different types of agents. E.g., contributors and mentors become more negative while committers are less negative after episodic changes while. In addition, in retired (unsustainable) projects, contributors and mentors exhibit lower response rates; committers and contributors are much less engaged and more negative.

5.5.3 Case study: Association between episodic change and institutional statements

To communicate concretely how the institutional and socio-technical dimensions interact within ASF ecosystem, we showcase two diverse instances of their mutual interrelationship.

§ Case A: From IS to CI. *Calcite* (former *Optiq*) is a project migrated to Apache Incubator from another platform. From the initial months since *Calcite* was accepted into Apache Incubator, we detected a series of IS preceding a sharp increasing CI in the average social network clustering coefficient (`s_avg_clustering_coef`), which reflects a growth in the connectivity between people in the project's social network. After closely examining the IS, we found that this IS→CI interrelationship revealed meaningful events.

In June 2014, the second month since their migration, the developers started arranging online meetings and calling for connections as an effort to adapt to the new community and its workflow. Below are examples of such IS from two of the developers, Dev_1 and Dev_2:

Dev_1 *"I'd like to hear from people across the community, users as well as developers, people who have contributed a two-line patch six months ago as well as people who check-in daily. ... I propose that we have an online meeting to introduce ourselves, using Google hangouts. I'd especially like to meet people who would like to get involved by writing documentation, blog posts, and testing."*

Dev_1 *"We can discuss at the online meeting. I volunteer to write a draft report on Monday."*

Dev_2 *"Before we start opening issues there, I would like to discuss here if you want to import the GitHub issues into JIRA."*

Shortly after, in July 2014, another IS shows that Dev_1 proposed for their first and subsequent releases on Apache:

Dev_1 *"Optiq has been releasing regularly, but it is important that Optiq soon makes an initial release under the Apache process."*

Following these IS, from July to October 2014, the project's social network connectivity skyrocketed. An explanation of this CI is that the project was under its migration process to the Apache platform. Instead of gradually building up a community, a migrated project like *Calcite* transferred its established community to the new platform within a relatively small amount of time. Moreover, the initiation of regular releasing also triggered interactions and enhanced this surge.

§ Case B: From CI to IS. From March to May 2016, the *Quickstep* project experienced sharp growth in its number of developers (`t_num_dev_nodes`) as indicated by our change detection results. Immediately after this CI, in June 2016, we detected a cluster of IS. This CI→IS interrelationship helped us locate a notable phase of the project when developers started to raise concerns in response to their rapidly growing community.

A developer first expressed their concern about the lack of on-list discussion in opposition to the frequent development activities with the following IS: *"There seems to be quite a lot of work happening on project, but I can't figure out where the design discussions and decisions are being made. ... Where are design discussions happening? Does the team have a weekly or daily meeting? I ask because if discussions are happening off-list, and in particular if decisions are being made off-list, the project is not attractive to outsiders..."*.

In a separate discussion thread, the developer also suggested that a clear criterion and process regarding committer election was needed and favorable to attracting and retaining committers with the following IS: *"You, as a PMC, should decide what are the criteria & process for making someone a committer... Electing committers is a consequence of a successful strategy for growing community, and helps further that growth."*

5.5.4 RQ3: What are the associations between episodic change direction and the sentiment to IS-related discussions?

In previous RQs, we conducted exploratory and qualitative studies of project discussion and dynamics around episodic changes. In this section, we triangulate those with quantitative studies, to understand the pre-cursor to episodic changes in terms of the sentiment in IS-related discussions, and vice versa. Such an approach enables us to understand how episodic changes in organizational structure features can introduce sentiment shifts in discussions on rules and regulations, and vice versa.

We present the occurrences of a 4-way combination of CI and IS in the table for both graduated and retired projects, as shown in Table 5.2. Each row represents a feature in the socio-technical system, while each column stands for a specific case in the association from Change Interval (CI) to Institutional Statements (IS), and vice versa. For example,

Table 5.2. The temporal association between episodic Change Interval (CI) and sentiment in the Institution Statement (IS). The value shows the occurrences of respective cases (ratio in parentheses). The underlined cells indicate the variables that have the most frequent pattern.

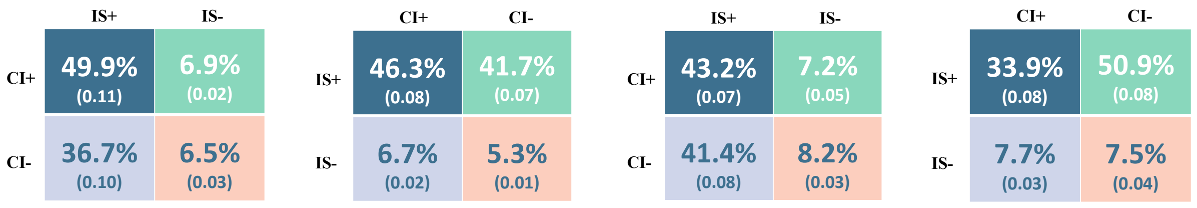
Status	Feature	$CI^+ \rightarrow IS^+$	$CI^+ \rightarrow IS^-$	$CI^- \rightarrow IS^+$	$CI^- \rightarrow IS^-$	$IS^+ \rightarrow CI^+$	$IS^+ \rightarrow CI^-$	$IS^- \rightarrow CI^+$	$IS^- \rightarrow CI^-$
Graduated	`s_avg_clustering_coef`	101 (0.48)	13 (0.06)	78 (0.37)	20 (0.09)	67 (0.39)	85 (0.49)	8 (0.05)	12 (0.07)
	`s_graph_density`	41 (0.25)	11 (0.07)	91 (0.55)	23 (0.14)	35 (0.32)	57 (0.52)	9 (0.08)	8 (0.07)
	`s_num_component`	50 (0.49)	5 (0.05)	43 (0.42)	4 (0.04)	50 (0.49)	45 (0.44)	3 (0.03)	5 (0.05)
	`s_num_nodes`	167 (0.64)	26 (0.1)	57 (0.22)	11 (0.04)	127 (0.6)	63 (0.3)	12 (0.06)	11 (0.05)
	`s_weighted_mean_degree`	94 (0.51)	13 (0.07)	67 (0.36)	12 (0.06)	80 (0.49)	68 (0.42)	7 (0.04)	8 (0.05)
	`t_graph_density`	83 (0.45)	7 (0.04)	84 (0.46)	9 (0.05)	59 (0.45)	51 (0.39)	11 (0.08)	9 (0.07)
	`t_num_dev_nodes`	138 (0.62)	20 (0.09)	55 (0.25)	8 (0.04)	92 (0.56)	58 (0.35)	9 (0.05)	6 (0.04)
	`t_num_dev_per_file`	104 (0.61)	14 (0.08)	47 (0.27)	6 (0.04)	49 (0.46)	43 (0.4)	11 (0.1)	4 (0.04)
	`t_num_file_nodes`	73 (0.51)	10 (0.07)	48 (0.34)	12 (0.08)	60 (0.47)	52 (0.41)	10 (0.08)	5 (0.04)
	`t_num_file_per_dev`	62 (0.44)	8 (0.06)	62 (0.44)	10 (0.07)	49 (0.4)	54 (0.45)	11 (0.09)	7 (0.06)
Retired	`s_avg_clustering_coef`	38 (0.54)	8 (0.11)	19 (0.27)	5 (0.07)	30 (0.39)	33 (0.43)	4 (0.05)	9 (0.12)
	`s_graph_density`	17 (0.35)	7 (0.15)	20 (0.42)	4 (0.08)	25 (0.45)	21 (0.38)	8 (0.15)	1 (0.02)
	`s_num_component`	18 (0.56)	2 (0.06)	10 (0.31)	2 (0.06)	15 (0.39)	16 (0.42)	4 (0.11)	3 (0.08)
	`s_num_nodes`	35 (0.39)	12 (0.13)	32 (0.36)	10 (0.11)	32 (0.37)	40 (0.46)	4 (0.05)	11 (0.13)
	`s_weighted_mean_degree`	41 (0.39)	8 (0.08)	43 (0.41)	12 (0.12)	28 (0.35)	43 (0.53)	5 (0.06)	5 (0.06)
	`t_graph_density`	34 (0.42)	2 (0.03)	38 (0.48)	6 (0.07)	25 (0.3)	43 (0.51)	7 (0.08)	9 (0.11)
	`t_num_dev_nodes`	49 (0.44)	7 (0.06)	51 (0.46)	4 (0.04)	26 (0.29)	51 (0.57)	6 (0.07)	7 (0.08)
	`t_num_dev_per_file`	35 (0.43)	2 (0.02)	35 (0.43)	9 (0.11)	12 (0.17)	47 (0.65)	4 (0.06)	9 (0.12)
	`t_num_file_nodes`	35 (0.39)	4 (0.04)	46 (0.52)	4 (0.04)	29 (0.34)	47 (0.55)	8 (0.09)	1 (0.01)
	`t_num_file_per_dev`	33 (0.39)	2 (0.02)	41 (0.48)	9 (0.11)	26 (0.33)	45 (0.58)	5 (0.06)	2 (0.03)

the column entitled CI^+IS^+ indicates that there is an increasing trend in a feature that serves as a precursor to positive IS discussions in the following month. CI can be either increasing (+) or decreasing (-), and the sentiment is either positive (+) or negative (-). For measuring the sentiment of IS discussions in respective months, we aggregate the sentiment across all IS discussions and get the majority sentiment.

For graduated projects, as shown in Table 5.2, we find that the number of nodes (`s_num_nodes`) in social networks accounts for most types of occurrences of CI and IS (168), suggesting that graduated projects are experiencing episodic changes with respect to the total number of active developers in social networks. We continue to use the tables in practice. The tables can also help us understand what is the precursor to positive/negative discussion in IS-related discussions, i.e., agreement/disagreement to rules and regulations. E.g., to find out the episodic change of which socio-technical feature is more likely to be followed by positive discussions, we can focus on the first 4 columns, e.g., $CI^{+}IS^{+}$, $CI^{+}IS^{-}$, $CI^{-}IS^{+}$, and $CI^{-}IS^{-}$, and then calculate the ratio of the cases having IS+ as the outcome, shown as the values in parentheses. In the case of graduated projects, it shows that a positive sentiment followed by an increasing episodic change in `s_num_nodes` has the highest ratio of 64%, suggesting that we can make people more positive about regulations and rules by recruiting more new-comers on the mailing list. One reason for this result is the fact that the newcomers are not the stakeholders, and they are more open to rules and norms.

As another use case of this table, we can ask the following question: What are the effects that sentiment has on the change interval of the number of unique committers (`t_num_dev_nodes`)? From Table 5.2, for retired projects, we find that the ratios of having positive sentiment and negative sentiment followed by an increasing trend of `t_num_dev_nodes` are 29% to 7%, respectively. It suggests that, in retired projects, odds of positive sentiment on IS-related discussions to attract new committers are more than four times than negative sentiment.

To account for the bias in the distribution of episodic change type and sentiment in IS-related discussions, we normalize all values, feature-wise, by taking the ratio of the occurrences over all events, e.g., normalized $CI^{+}IS^{+} = CI^{+}IS^{+} / (CI^{+}IS^{+} + CI^{+}IS^{-} + CI^{-}IS^{+} + CI^{-}IS^{-})$. And then we aggregate all features into four 2×2 matrices, two for graduated projects and two for retired projects, as shown in Figure 5.10. To compare patterns between graduated projects and retired projects, we first look at the first column of the left two matrices (i.e., $CI \rightarrow IS$). We find that graduated projects are more likely to have positive discussions after an increasing episodic change than retired projects (49.9% to 6.9% than 43.2% to 7.2%), suggesting that retired projects do not establish a feedback cycle for project progress, or maybe even worse, that they do not realize the progress they made. Then we attend to the second matrix for graduated and the last matrix for retired projects (i.e., all from $IS \rightarrow CI$), and we find that, for both graduated and retired projects, positive institutional discussions often serve as the precursor event to episodic changes (88.0% for graduated projects and 84.8% for retired projects).



	IS+	IS-
CI+	49.9% (0.11)	6.9% (0.02)
CI-	36.7% (0.10)	6.5% (0.03)

	CI+	CI-
IS+	46.3% (0.08)	41.7% (0.07)
IS-	6.7% (0.02)	5.3% (0.01)

	IS+	IS-
CI+	43.2% (0.07)	7.2% (0.05)
CI-	41.4% (0.08)	8.2% (0.03)

	CI+	CI-
IS+	33.9% (0.08)	50.9% (0.08)
IS-	7.7% (0.03)	7.5% (0.04)

Figure 5.10. The aggregated ratio of two-way transmission effects from CI to IS, and vice versa. From left to right, they stand for graduated (CI→IS), graduated (IS→CI), retired (CI→IS), retired (IS→CI). We show the standard deviation in parentheses.

RQ$_3$ Summary: In graduated projects, the control features for outcome sentiment to IS lay mostly in social networks, while for retired projects, they are more evenly distributed in both social and technical networks. For both graduated and retired projects, positive IS-related discussions are more likely to be followed by a episodic change. While the graduated projects are more likely to have positive discussions after an increase in CI than retired projects.

5.6 Takeaways for Practitioners

In this section, we distill our findings into some practical takeaways and suggestions.

Like in other self-governed institutions, norms, rules, and regulations in OSS projects generate, moderate and direct actions. Concerted governance efforts can result in episodic changes in the socio-technical structure, achieving feedback between effective self-governance and sustainability. Generally, we found more and longer episodic change intervals in graduated projects than in retired ASF projects, potentially explained by the observation that institutional discussions often trigger episodic changes in the project's socio-technical structure. However, institutional discussions are mostly lacking during such changes, so our first takeaway is that just like participation in the technical aspects, developers should be encouraged and even brought into project institutional discussions. We found that episodic changes are associated with temporary developer disengagement from the project, and that the negativity of mentors increases significantly after episodic changes. New episodic changes can be distracting to a team and may bring more management efforts to the project. Setting manageable expectations for the team ahead of time can limit feelings of frustration arising out of lengthy discussions. Thus, being more positive than negative may help keep change intervals shorter. On the other hand, we found that projects that graduate are much less negative toward changes compared to projects that retire. Fostering positive discussions may help the project adapt to changes and become more sustainable. As another takeaway, perhaps projects can benefit from timing episodic changes, to the extent possible, to occur during periods of low cross-team interactions/-collaboration. That can potentially ease the cost of upcoming episodic changes.

5.7 Threats and Conclusions

§ Threats Generalizing our findings beyond ASF, or even beyond the ASF Incubator projects carries potential risks, for example, not every OSS foundation has a mentor program like ASF's. The risk could be reduced by expanding the dataset beyond the ASF incubator, e.g., to include additional projects from other OSS incubators. ASF mailing lists are the only channel we consider, therefore, developers may communicate through in-person meetings, webpage documentation, and private emails. ASF mandates the use of public mailing lists for most project discussions, which causes an especially low risk of omitting institutional or socio-technical information. Annotations of institutional statements could be biased by individual annotators. However, given that the annotators were adequately trained with given reference materials, which lowers the possibility of bias. ASF developers may use different aliases or emails making it difficult to identify distinct developers, while ASF's regulations on using *apache.org* official email addresses and our de-aliasing process reduce such risks [4].

§ Conclusions Practitioners may be able to improve their individual practices, organizational management, and institutional structure by understanding why open-source projects cannot meet the expectations of nonprofit foundations. Additionally, it is important to consider how institutional design and socio-technical aspects relate to OSS to understand its potential sustainability. Through the artifacts they create, we demonstrated that socio-technical network features can capture the episodic change in the organizational structure of developers' collaboration and communication. Through the unified

[4] ASF committer emails: `https://infra.apache.org/committer-email.html`

view of socio-technical network features and institutional analysis, we leverage the unique attributes of Apache Software Foundation's Incubator projects to extend the modeling of OSS project sustainability, by analyzing a longitudinal dataset consisting of vast text and log corpora, as well as extrinsic labels for sustainable and unsustainable.

Chapter 6

Exploring Apache Incubator Project Trajectories with APEX

6.1 Introduction

In spite of the large amounts of resource put in them, many OSS projects end up on trajectories that are ultimately not sustainable. In recent work we showed that OSS project sustainability can be effectively predicted early on in project development from longitudinal project and process metrics supplemented by socio-technical network metrics (developer communications and code contributions), specific to the Apache Software Foundation (ASF) [22]. ASF, as one of the most popular OSS communities, provides specific guidelines and establishes regulations to help OSS projects eventually become self-sustainable. Nascent projects with ASF aspirations are housed in the Apache Software Foundation Incubator (ASFI) for a period of time, after which they are graduated into ASF if they are found to be sustainable, otherwise they get retired.

Promisingly, our work [22] implied that monitoring and reflecting on their sustainability forecast can enable projects to act proactively, and potentially correct downturns in the forecasts. To enable such monitoring in practice, here we present a dashboard tool, *APEX*, intended for nascent projects in the Apache Software Foundation Incubator to monitor their sustainability trajectories over time, thus allowing for timely course corrections and for potentially improving the likelihood of project graduation into ASF. Our motivation goes beyond ASFI as many nascent OSS projects fall outside the ASF domain, and its well developed community support structure. Self-monitoring and self-correction may be even more pertinent to those. While intended for ASF projects specifically, APEX is designed in a generic way and thus can easily accommodate data from repositories other than ASF.

Related Work ASF provides a monitoring tool, Clutch[1], to help OSS developers self-reflect and take actions when their projects are experiencing issues. The Clutch tool uses colors to signal the status of project metrics, e.g., missing documentation, lack of new committers, etc. Although it works well for its intended use, Clutch's analysis is of limited use as a real-time monitoring tool since (1) it is not project-specific, i.e., all projects follow the same standards for all features regardless of their project size or context; (2) it does not consider historical records; and (3) it does not make actionable suggestions. Our APEX tool complements the existing Clutch tool by providing additional analytics power for understanding the longitudinal socio-technical aspects of projects. It also can yield potential actionable insights.

There are other projects that focus on analytics for OSS sustainability outside of ASF domain. E.g., the Augur and GrimoireLab within the CHAOSS (Community Health Analytics Open Source Software) project[2], provide a toolbox for project sustainability self-monitoring. However, unlike APEX, they don't provide a synthesis of metrics into a longitudinal sustainability forecasts, or allow deep dives into email and commits.

Next, we first introduce APEX, and then describe use cases that demonstrate its utility in a) monitoring for ASF project downturn events, b) identifying longer term engagements between developers, and c) within-ASF project comparisons. The APEX app is available at `https://ossustain.github.io/APEX/`. Code and data are available at `https://github.com/ossustain/APEX/`.

6.2 Data and Implementation

Data Source The APEX pipeline depends on four types of data: basic project information, periodic project reports, email communications, and code commits. We use our

[1]Clutch Analysis: `http://incubator.apache.org/clutch/`

[2]CHAOSS community `https://chaoss.community/`

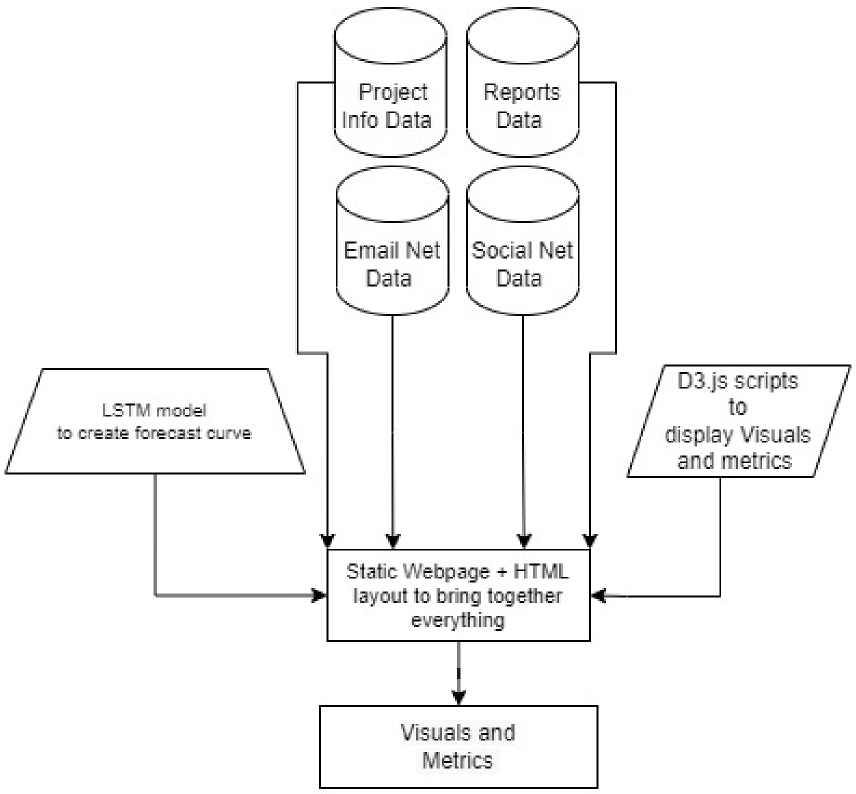

Project Info Data
Reports Data
Email Net Data
Social Net Data
LSTM model to create forecast curve
D3.js scripts to display Visuals and metrics
Static Webpage + HTML layout to bring together everything
Visuals and Metrics

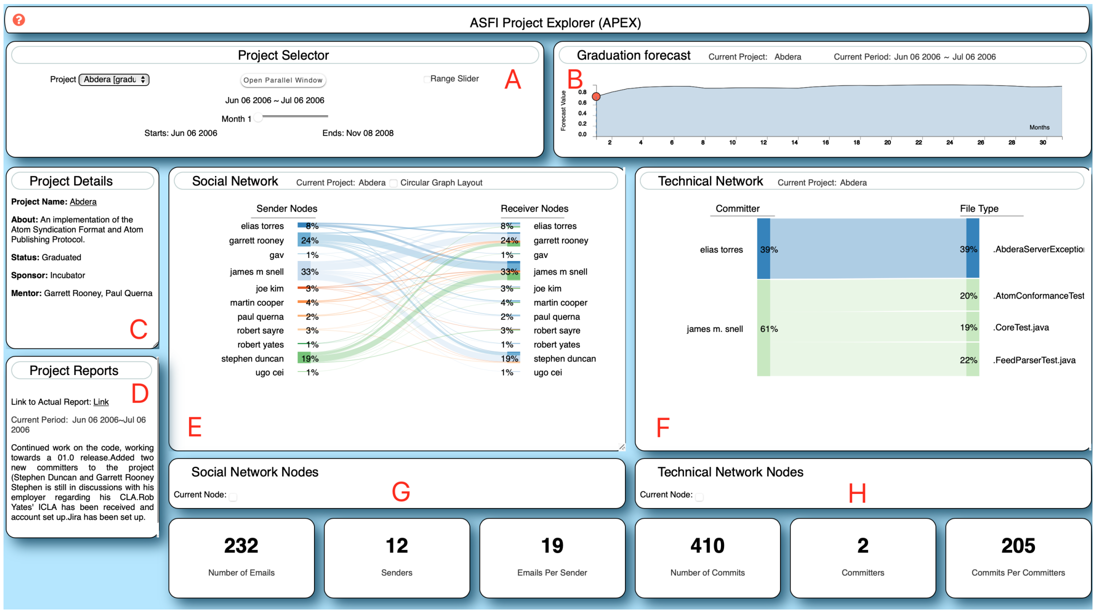
Figure 6.1. The APEX pipeline

previously published dataset from ASFI [128] to obtain the emails and commits. The dataset comprised 211 graduated and 62 retired incubator projects (in total of 273), with 1,201,746 emails, and 3,654,196 commits, with each project spending on average 22.32 months in incubation.

We scraped project information (name, mentors, dates, status) from the project's ASF Incubator homepage[3], using Python's BeautifulSoup Package. The Apache mailing list archive,contains full historical information for all projects including project participants, mentors who assisted with the projects, project reports, all emails, and all commits. To obtain developer emails we frequently had to backfill partial email addresses by writing scripts to search for the partial email throughout the email text. We used the same approach to identify unique committers that may have used aliases.

Social and Technical Networks From the ASF incubator data we derive two kinds of longitudinal networks for each project, for each month: a social and a technical. The

[3]ASF incubator: `http://incubator.apache.org/projects/`



Project Name: Abdera

About: An implementation of the Atom Syndication Format and Atom Publishing Protocol.

Status: Graduated

Sponsor: Incubator

Mentor: Garrett Rooney, Paul Querna

Link to Actual Report: Link

Current Period: Jun 06 2006~Jul 06 2006

Continued work on the code, working towards a 01.0 release.Added two new committers to the project (Stephen Duncan and Garrett Rooney Stephen is still in discussions with his employer regarding his CLA.Rob Yates' ICLA has been received and account set up.Jira has been set up.

Figure 6.2. Layout of the APEX Dashboard

social networks have directed edges between developer, derived from the email archives, using the method by Bird et al. [19]. We present the social networks in a bipartite graph layout, using Sankey diagram [193]. There are two sets of nodes: on the left are all senders of messages in a given month. On the right are those who either received a direct message, or, in the case of a broadcast message, those who have responded to that message. The edges are directed from the sender to the receiver node, except for the broadcast messages where the edges are directed from a node that has sent an email to a node that has replied to that email [19].

The technical network is also a bipartite graph with two sets of nodes: on the left are the developers that have made commits in a given month and on the right are all the file types committed to in that month (e.g., .java, .html, etc., based on their file extensions). The edges connect the developers to the file types they committed to in that month. We aggregate the network edges in a monthly manner, i.e., the longitudinal networks of each project consist of monthly network snapshots.

Implementation Technology The APEX tool design and pipeline is illustrated in Figure 1. To implement the front-end of APEX we use the *D3* JavaScript library (`https://d3js.org/`). *D3* uses visual layouts and an associated tool-set to improve front-end efficiency. *D3* also provides developers with design flexibility through standardized data manipulation operations. Additionally, one of D3's foci is transitions and animation. This allows our dashboard to quickly adapt to changing inputs, e.g., a change of the current month. To provide interactivity of our tool, we use the SVG rendering technology, which is based on Document Object Model (DOM) operations and supports precise user interaction. We also make use of *jQuery* extensively. jQuery is a JavaScript library that helps simplify and standardize interactions between JavaScript code and HTML elements. We use jQuery to design event listeners: processes in JavaScript that wait for an event to occur. This has allowed us to create a seamless dynamic experience throughout the dashboard, where a change in one section adjusts all related sections and visuals accordingly. We have also made use of a lightweight range slider with multi-touch support called *noUIslider*, and it has an in-built event listener function allowing integration with the rest of the elements in the dashboard. Frequent DOM operations are costly, negatively impacting the user experience by screen flashing and stuttering during the interactions. We relax this cost by keeping each project and month stored in separate json files.

Sustainability Forecasting APEX features the AI based sustainability forecasting model by Yin et al. [22]. They implemented a 3-layer LSTM model: a 64 neurons LSTM layer with a 0.3 rate drop-out layer, and then followed by a dense layer with the *softmax* function to yield the predicted likelihood of project graduation. In the experimental setup, the graduated projects are encoded as 1; retired projects as 0. During training, the monthly socio-technical networks variables (e.g., number of nodes/edges, clustering coefficient, and mean degree in the networks) of each project were fed into the model. This LSTM neural network based model gives a sustainability forecast in each month of the project development. More experimental details can be found in the paper [22].

6.3 Dashboard Elements

Dashboard Panes. There are four main sections to the dashboard, see Figure 2: (1) top pane (panes A,B), (2) left pane (panes C,D), (3) middle pane (panes E and G) (4) right pane (panes F and H).

The *top pane* has two parts to it. The left side (A) allows the user to select a project of interest in the drop-down menu and a specific month through the month slider. Based on these inputs, the other panes change dynamically to display the respective information for the selected project and month. In addition, this section also allows the user to toggle a checkbox to switch to a range slider to display a range of months, e.g., 1-5 months, instead of a single month. To the right (B) is the sustainability forecast visual, which depicts the sustainability forecast for the project, ranging from 0 (not sustainable) to 1 (sustainable), for any given month.

Below those, the left pane consists of two distinct sub-sections: the project info pane (C) and the project report pane (D). The former shows the project name, a link to the official website, and the project's status (i.e., graduated or retired). The ASF sponsor's name (if anyone in particular, otherwise 'incubator') is displayed below that. At the bottom is a short introduction to the selected project. Below that, in the project's report pane (D), we present the report submitted by the project to the ASFI, for the given month.

The *middle pane* consists of the social network visual on top (E) and its related metrics below it (G). The social bipartite graph is presented as a Sankey diagram [193], the height of a node illustrating the % (relative to the total) of emails sent (left) or received/replied-to (right) by that developer in a given month. The sizes of the flows sre proportional to the number of emails exchanged between the developers. Hovering over a developer's name emphasizes all developers that have received a directed email, or responded to a broadcast, from that developer.

In the *right pane*, on top is the technical network of developers who have committed to files (F), and their metrics are below (H). For the visual, we also use a bipartite Sankey diagram. On the left is a list of developers, while the file type (i.e., extension) of the files committed to is on the right. The percentage of one's efforts relative to others' is shown on the left, with the sizes of the flows proportional to that. Hovering over a developer's name, or, respectively, a file type, emphasizes additional information: all file types a

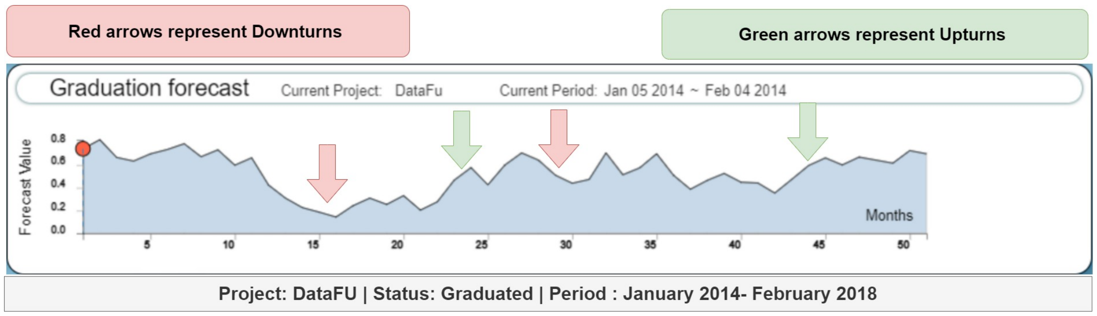


Figure 6.3. Using the sustainability forecast to understand and explore downturns (red) and upturns (green) of project *DataFu*.

developer has touched, or all developers that have touched that file type along with their % contribution, respectively.

Additionally, by hovering over and clicking on a developer's name in the social, respectively the technical networks, we get a button with their name under the network, which when clicked opens a window with a list of all their emails, respectively commits, in that month. They appear in a pop-up window next to the dashboard.

6.4 Use Case Examples

APEX conveniently shows in one place the project info, monthly aggregated code commits, email communications, and two unique features of ASF, the periodic report info and the graduation status. Such rich and fine-grained information can enable researchers and practitioners to study the trajectory of a given project by showing changes over time, including identifiable patterns and up/down trends, in the socio-technical networks and the sustainabiity forecasts.

Use Case I: Studying Sustainability Turning Points Patterns and trends in the longitudinal socio-technical networks can be studied to identify causes for downturns in the sustainability forecast, allowing APEX users to be proactive with changing project trajectories. For example, as shown in Figure 6.3, for the selected project *DataFu*, by simply eyeballing we can identify that there is a big downturn around month 12. Some possible reasons for this could be that (1) The project just launched a big release; or (2) Some core developers left the projects. Going through the email discussions in months adjacent to the downturns may offer reasons for the changes, which in this case is likely

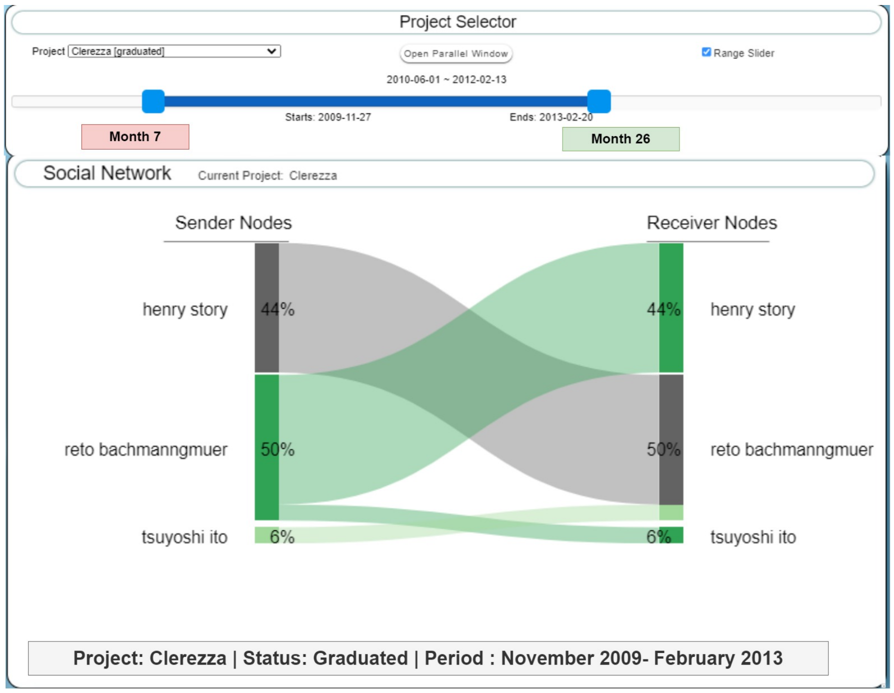


Figure 6.4. Aggregating project Clerezza's social networks over a range of months (7-26) shows longer-term engagements

the latter.

Thus, the monthly social-technical networks combined with the real-time sustainability forecast can allow practitioners and researchers to monitor for downturn events and react proactively.

Use Case II: Studying Different Length Engagements APEX allows aggregating the networks over a range of months. This allows the study and comparison of different length engagements, both social and technical in nature. This can be done by enabling the range slider, which allows multiple months to be selected at once, yielding a time range for the nodes and edges in the networks. Once the range is specified, the metrics and the visuals are adjusted to display multiple consecutive months of interactions.

An example of a social network over a longer range is shown in Figure 6.4. There we see thicker and thinner edges; the former indicate communications that recur over multiple months, attesting to a longer term engagement between those developers, i.e., recurring communication. By comparing the short-term and long-term social-technical networks, we can identify recurring patterns over longer periods of time during the project incubation.

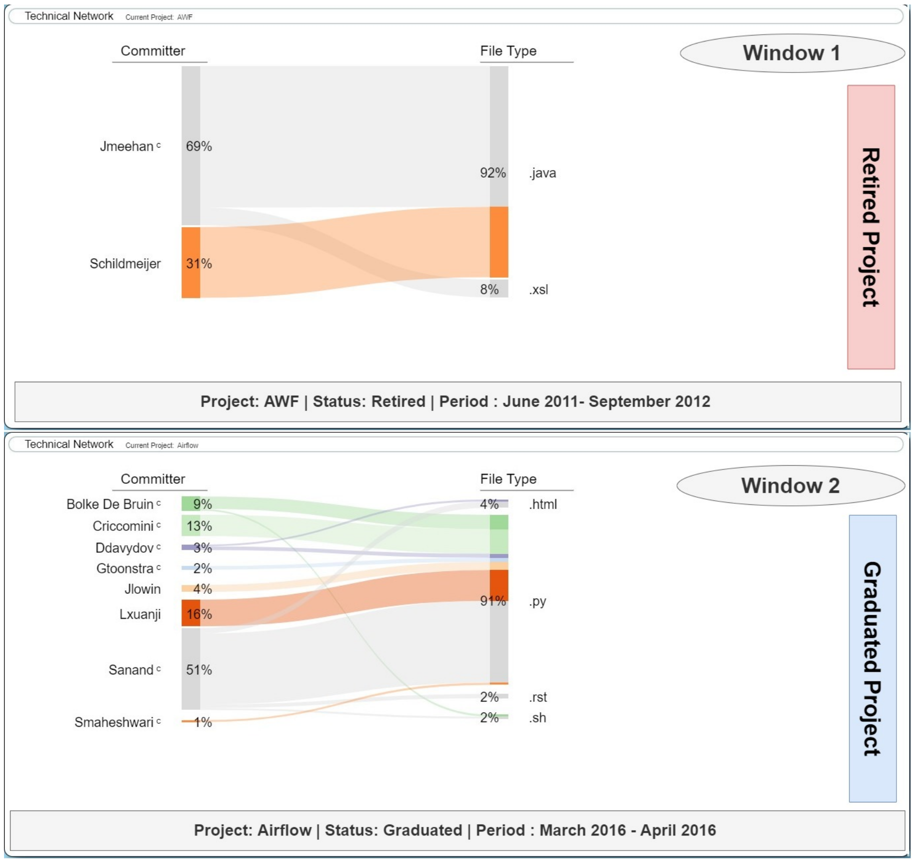


Figure 6.5. Parallel windows can contrast technical networks between project AWF (retired, top) and project Airflow (graduated, bottom)

Use Case III: Cross-Project Comparison APEX also allows users to compare and contrast two projects by opening up parallel windows of our dashboard. Thereby, researchers can explore multiple projects simultaneously to generate hypotheses about relationships between their socio-technical structure and graduation status. E.g., Figure 6.5 shows the technical networks of two projects, one graduated and the other retired. An immediate pattern that emerges is that more developers are committing code changes in the graduated project than in the retired one. Researchers can followup on this hypothesis by looking into the driving factors behind it, using, e.g., productivity studies, or topics of discussions.

6.5 Using APEX Beyond ASF

We have designed APEX to serve ASFI projects that are early in their incubation to monitor and reflect on their progress in a more agile way than previously possible. But APEX is in principle not limited to ASFI data. It takes as input JSON files and visualizes them in different ways. To aid projects outside of ASF that want to benefit from it, we have made our full code and data publicly available. We provide a README file, `https://github.com/anirudhsuresh/APEX/blob/main/README.md`, that details the JSON formats of the required input data. The README file links to scripts with which comma separated values (CSV) files, common outcome of repository mining, can be converted to the required JSON format for all the required APEX components: email networks, email metrics, technical networks, commit metrics, project info, project reports, and sustainability forecasts. The last one will have to be calculated from the others, using the code provided in our previous study [22].

6.6 Limitations and Conclusion

Limitations Our dataset is large and diverse (within ASF) but limited to ASFI projects, so generalizing beyond ASF is risky. However, we provide a README file with instructions to aid non-ASF projects in using APEX. Selecting a range of months can result in very dense networks that are hard to read or interpret. This function is most useful when limited to a few consecutive months.

Conclusion Research into OSS project sustainability can present actionable insights for project maintenance. In this work, we presented a dashboard tool for exploring a longitudinal data-set of technical contributions and developer communication in ASF incubator projects, with extrinsic, graduation success labels and forecasts. The tool can be used for real-time monitoring and study of ASFI projects. It can also help generate hypothesis about OSS project sustainability. Future work will be aimed at adding additional data sets.

Chapter 7

How do OSS projects govern themselves in the wild? An Empirical Study on `Governance.md` Markdown Files

7.1 Introduction

As we increasingly transition into a digital age, open-source software (OSS) repositories are emerging as significant shared resources, echoing the dynamics of traditional commons such as fisheries [25], forests [194], and irrigation systems [195]. However, digital artifacts and collaborations in OSS bring forth a unique set of challenges in sustainability that demand innovative perspectives [196]. In this work, we turn to Elinor Ostrom's governance theories, originally formulated to manage natural resources [82], as a valuable lens through which we can decode the dynamics and organizational mechanisms of OSS projects [138]. Thus, from an empirical perspective, it is important to ask: Can we find mappings from rules in traditional commons to rules in digital commons? And what are the patterns and anti-patterns of adopting the rules?

Open Source Software (OSS) projects leverage markdown files to relay vital information concerning the project in a digestible and readily accessible format. For instance, the `README.md` file offers a comprehensive overview of the project. It typically includes the project's purpose, instructions for setup and usage, and guidelines for making contributions, fostering a more coherent understanding of the project for both current contributors and prospective ones [65]. The `Code_of_Conduct.md` file often plays an instrumental role in defining the behavioral norms and expectations within the project. By articulating these rules of engagement, this file helps to nurture an inclusive and respectful environment within the OSS community. It further provides a recourse for handling conflicts and misconduct, fostering a culture of respect and mutual understanding among participants [197]. Licenses dictate the legal framework for using, modifying, and distributing the software, and these are typically defined in the `License.md` file. The type of license chosen can significantly influence the project's attractiveness to contributors and its potential for reuse [198]. In examining the GOVERNANCE.md file within certain GitHub OSS repositories, we identify its pivotal role as an instrument delineating governance akin to constitutions. Yet, if we apply the Ostrom's model, there's potential for a more nuanced interpretation. The Constitutional level could specify overarching project frameworks like the presence of an oversight committee or the selection of an IP license. Moreover, the collective choice level outlines procedures for role appointments and changes to operational regulations. Meanwhile, Operational rules focus on routine tasks and procedures, such as report submissions or dispute resolutions.

Thus, the necessity to bridge the gap between Ostrom's governance theories and its application in OSS environments becomes self-evident. The rich timestamped data that OSS repositories and specifically, GOVERNANCE.md files offer, presents an unparalleled opportunity to study the governance structures of these digital commons. Within these files, we find recorded histories of project rules and regulations, which can aid in comprehending the project's governance evolution.

We posit that:

> Just as the right dosage is crucial in medicine, striking a balance between under-governance and over-governance is key in OSS projects. Leveraging insights from Ostrom's theory, our research aims to guide software communities in devising more suitable governance rule configurations depending on the projects' context.

The value of this work is twofold. For the field of management science, our research provides novel insights into the application of Ostrom's rule class typology (more details in [138] in digital contexts. Consider, for instance, the world of cryptocurrencies, another instance of digital commons where governance is both emergent and crucial. Here, understanding the implementation and effectiveness of governance rules could help in better designing and managing cryptocurrency networks. From a software engineering perspective, our work has resulted in an effective tool - a high-performing classifier capable of identifying the presence of seven classes of Ostrom governance rule types within the `Governance.md` files. This tool, the first of its kind, allows for automated, large-scale analysis of governance mechanisms in OSS projects. By identifying what types of rules are implemented and how they evolve, we can inform the practices of software development communities, helping them craft more effective and sustainable governance rules.

To study how OSS contributors use `GOVERNANCE.md` markdown file to build rules and govern the community, we gathered project historical data, e.g., code commits, issues, and comments, including all timestamped diff changes to those governance files, from 703 GitHub repositories which contain such `GOVERNANCE.md` markdown file in their root directory.

We have provided an annotated dataset that categorizes statements in `GOVERNANCE.md` based on Ostrom's definition of rules. Notably, a single statement can encompass none, one, or multiple types of rules as per Ostrom's classification. Then we used it for classifying all seven types of governance rules (the position rule and boundary rule are combined due to their similarity in the context of OSS), some for specifically establishing who can participate and under what conditions, some for deciding who can vote, and the selection method for project decision-making. Next, we also modeled the dynamics of these rules, studying their temporal evolution, the actors involved in their changes, and the association between the number of adopted rules and project size. These insights can help software engineers understand how rules and norms evolve in a project and provide them with guidance on when and how to intervene or adapt rules for the benefit of the project.

In summary, the contributions of this work are:

- The introduction of a unique dataset of approximately 703 GitHub repositories with `GOVERNANCE.md` files, provides fresh ground for investigating the application of Ostrom's governance rules in the OSS environment.

- The creation of the first-ever annotated corpus linking Ostrom's governance rule types to software governance text with a high-performing classifier (¿90% overall F-1 score) that can automatically identify and classify sentences into Ostrom's governance rule types.

- A meticulous tracking and analysis of the chronological introduction and modification of governance rules organized by Ostrom's rule classes, providing the first empirical insights into rule evolution within OSS repositories.

This paper is a fresh endeavor to harness the power of traditional governance theories for the sustainability of OSS projects, offering the potential to improve project outcomes. While our investigations are rooted in Ostrom's theories of self-governing communities and GitHub's context, the implications of our findings may be pertinent to numerous other software projects seeking sustainability amidst the digital commons.

7.2 Related Work and Theories

Here we introduce the theories behind this work on governance for commons and related works on governing OSS projects.

7.2.1 Theory of Governing the Commons

The study of commons governance is a well-established field, prominently shaped by the pioneering work of Nobel laureate Elinor Ostrom and her seminal book, Governing the Commons [140]. Ostrom's Institutional Analysis and Development (IAD) framework emerged as a powerful tool to investigate self-managing governance institutions within communities, particularly those related to natural resources, e.g., water [141], marine [142], and forest [24].

A significant challenge in natural resource commons settings is addressing the issue of 'free-riders' [143] – individuals who benefit from resources without contributing to their production or upkeep. In the context of open source software, like forests, fisheries, and water, this free-rider problem can lead to over-exploitation, resulting in ‘Tragedy of the Commons’ [173].

Despite the inherent differences between OSS and natural resources, specifically their non-degradable and reusable attributes, the Open source software, as a form of digital commons, is not invulnerable to its own variant of this tragedy. That is, the free riders can still exist in OSS, and they are the people who only benefit from OSS, but not contributing code, not providing bug reports, not participating in community discussions, and not offering financial or other types of support to the project. This can become problematic, especially for smaller projects that are maintained by volunteers and might require additional resources to manage increasing complexity, technical debt, or user base. Moreover, this can hold true for large companies as well. E.g., Elastic’s (an open-sourced project on distributed search) leadership alleged that Amazon Web Services (AWS) was free-riding by taking their freely available code, creating derivative works, and offering these to customers without significant contributions to the original projects[1].

Therefore, while free-riding is an expected aspect of open-source projects, an imbalance between free-riders and contributors can pose challenges to a project’s longevity and quality. The overutilization of human resources due to the prevalence of ‘free-riders’ can stifle the software’s development or even lead to its abandonment [145]. This phenomenon, described as ‘collective inaction’ by Ostrom and Hess [146], mirrors the adverse effects observed in the natural resource commons. Ostrom’s research provided significant insights into how communities devise and implement effective self-governing institutions to circumvent these undesirable outcomes. The IAD framework became a pivotal instrument in achieving this goal [138], and its utility was further highlighted by its application to the digital or knowledge commons [147, 146]. The study of self-governance in OSS, in particular, continues to benefit from Ostrom’s impactful contributions [82].

7.2.2 Ostrom’s Design Principles and Rules

Elinor Ostrom’s design principles and governance rules function within distinct strata of her Institutional Analysis and Development (IAD) framework [23]. The design principles

[1]Why we had to change Elastic licensing: Link

Rule Name	Definition
Position	Identify roles to be filled by people; requirements of people eligible to fill the role;
Boundary	establish constraints and conditions for entering/exiting positions.
Choice	Specify what participants must or must not or may do in their position and in particular circumstances.
Aggregation	Aggregation rules decide who votes and the selection method for a proposal.
Scope	Scope rules set limits on the actions, outcomes, and events that may occur.
Information	Specify how information can/cannot be communicated.
Payoff	Determine the rewards or penalties assigned to actions.

Table 7.1. Definitions for Seven Types of Ostrom's Governance Rule in OSS context.

provide high-level normative criteria that successful self-governed common-pool resource (CPR) systems often embody, whereas governance rules present a granular taxonomy for comprehending and classifying the operative mechanisms of these institutions [199].

While both design principles and governance rules are aimed at effective and sustainable resource management, they operate at different granularities [196]. Design principles act as overarching directives for maintaining CPR systems [200]. These principles encompass attributes such as definitive boundaries, congruity between rules and local circumstances, collective choice agreements, monitoring, graduated sanctions, conflict resolution mechanisms, and nested enterprises for systems of a larger scale [201]. However, governance rules offer an intricate description of constitutional, collective-choice, and operational-level protocols within a collective action milieu, delineating into seven rule types in the context of OSS [196]: position, boundary, scope, aggregation, information, and payoff rule. These rules strategically guide and regulate the conduct and actions of participants within a specific institutional context.

In the interconnected relationship between Ostrom's rule types and design principles, the rules furnish the methodologies through which the design principles can be actualized within an institution or a CPR system. Schweik et al. developed a rule classification system to understand how OSS communities organize themselves and make decisions [196]. The seven types of rules identified are [202] shown in Table 7.1.

7.2.3 Governance in Open Source Software

Governance mechanisms in OSS projects are complex and multifaceted, and several studies have delved into understanding their evolution and impact, often times researchers study the project-level governance together with OSS success and sustainability [82].

Project-level governance in OSS refers to the rules, practices, and processes that guide decision-making within a specific open-source project [203]. Several studies have examined how the governance model can influence project outcomes. For instance, O'Mahony et al. [204] conducted a multimethod study on an open-source software community and found members established shared formal authority but integrated democratic mechanisms, allowing for authority evolution over time. Schweik et al. [205] analyzed factors impacting open-source project outcomes, focusing on a representative sample. They find that revealing a 'virtuous circle of collaboration, with developer motivations and team assembly mechanisms contributing to sustainability Jensen et al. [206] study OSS development as self-organizing socio-technical interaction networks, effectively managing project activities to produce adaptive software. It further investigates governance practices within such networks, utilizing case studies from a large OSS project Moreover, project-level self-governance can be modeled using a socio-technical approach using developer's email communications (social side) and code collaborations (technical side) [28, 22, 207]. Linåker et al. [208] presents a framework for assessing the health of OSS projects, and the authors identify 107 characteristics across 15 themes, focusing on socio-technical aspects of OSS communities and their maintenance processes.

Some projects follow a Benevolent Dictator For Life (BDFL) model, where a single individual has the final say in decisions [209], like Van Rossum for Python community [210]. The BDFL model can be effective for OSS projects, particularly in the early stages when a clear, unified vision can help drive the project's growth and development. Other projects operate under a more democratic model, where decision-making power is distributed among multiple contributors, such as in the case of the Debian Project [211].

In summary, the governance model in open-source software, whether at the foundation or project level, plays a significant role in project sustainability, community engagement, and overall success.

7.3 Data and Methods

In this work, we leverage a previously published data set [212] consisting of hundreds of GitHub projects which contain a markdown file named as `GOVERNANCE.md`, which serves as a codified guide for governing the respective OSS project. To avoid duplicated counting where projects use other projects which contain a `GOVERNANCE.md` markdown file as a dependency package, the authors only include GitHub projects that have a `GOVERNANCE.md` in their root directory.

On the data end, in addition to the previously published data set, we gather complementary time-stamped trace data of projects' commits, issues, and comments using PERCEVAL [184]. Our final data contains GitHub 703 projects. In total, we collect 2,617,277 commit records from 56,384 unique GitHub committers, 1,247,397 issue records, and 11,1238 comments replying to those issues.

7.3.1 The Role of `GOVERNANCE.md`

This `GOVERNANCE.md` can be analogized to a constitution for the project, setting forth the various norms, regulations, and procedures that dictate the functioning and structure of the project. The inclusion of a `GOVERNANCE.md` file underscores the existence of a well-thought-out governance model within the OSS project, signaling an advanced degree of organizational maturity. The document `GOVERNANCE.md` can also reflect the collective intelligence and experiences of the project's contributors and maintainers, showcasing a deliberate approach towards the decision-making and problem-solving processes inherent to the project.

In a `GOVERNANCE.md` file, several topics are commonly observed, e.g., (1) Defining Roles and Responsibilities: This can involve explicit definitions for distinct roles within the project, such as maintainers or contributors, along with the enumeration of the specific duties and responsibilities that each role entails. (2) Decision-Making Mechanisms: The document may outline the procedural mechanics behind critical project decisions, encompassing topics such as feature inclusion, bug prioritization, and implementation of substantial changes. (3) Contribution Guidelines: The file often includes detailed guidance for making project contributions, specifying the protocol for proposing changes, criteria for code review, and adherence to specific coding standards or conventions. (4) Conflict Resolution Protocols: An essential element of a `GOVERNANCE.md` file is the articulation of systematic procedures for conflict resolution, providing a framework to navigate disagreements or disputes that emerge within the project's lifecycle.

In sum, the existence of a `GOVERNANCE.md` file in a GitHub repository serves as an artifact of project governance, proactively managing the complexity of collaboration and setting the stage for efficient decision-making. This reflection of governance sophistication could consequently play a pivotal role in attracting contributors and ensuring the long-term success of the OSS project.

7.3.2 Classifying Ostrom's Governance Rule Types

Ostrom's design principles often span multiple sentences and can encompass a diverse range of concepts, making it challenging to accurately identify and categorize them. The sentence-level approach, however, allows us to dissect these complex principles into manageable, semantically cohesive units, leading to more accurate and meaningful predictions. This tailored approach underscores the flexibility and precision offered by machine learning in decoding and categorizing the nuances of governance rules within OSS projects. Therefore, the sentence level of granularity accommodated our needs effectively and provided a balance between interpretability and precision. In this study, we choose to employ sentence-level classification for Ostrom's governance rules types.

Initially, we conducted a random sampling of `GOVERNANCE.md` files across multiple OSS projects. Each code diff to a `GOVERNANCE.md` file was split into individual sentences using regular expressions [213]. A total of 1000 samples were manually annotated for the presence of Ostrom's rule types, specifically the Position, Boundary, Aggregation, Information, and Scope rules. We observed a significant scarcity of samples pertaining to the 'Pay-off' rule (which specifies rewards and punishment for certain actions, and only 2 samples exist), and therefore we decided to exclude the Pay-off rule type. Moreover, in the process of refining our annotation scheme, we found the Position and Boundary rules to exhibit significant overlaps, leading us to consolidate them into a single rule category. We also introduce a new 'None' type rule to indicate there is no governance-related rule in the given sentence. During the annotation phase, three independent annotators were employed to ensure the accuracy and reliability of the manual annotations. All annotators are required to read and refer to the annotation guideline when annotating, which is publicly available[2]. The inter-annotator agreement was quantified using Krippendorff's alpha [214], which confirmed a robust level of agreement among the annotators, substantiating the reliability of our annotated data.

Few-shot learning is often used to address the issue of data scarcity and the practical need for models to adapt rapidly to new tasks or domains. Due to a limited dataset, the risk of model overfitting is reduced [215]. The sentence-level granularity was chosen for training a classifier using a high-performing model, named Efficient Few-shot Learning with Sentence Transformers (SetFit) [216], for multi-label classification. In the case of Ostrom's governance rule, one sentence can contain multiple rules. In order to enhance the model's training, we performed text data augmentation [217] to increase the dataset to two times of its original size while maintaining the distribution of rule types. This data augmentation strategy was deemed necessary to ensure an adequate representation of each rule type in the data, thus improving the generalizability of the trained model. Lastly, we employed a grid search approach for fine-tuning the model's hyperparameters, a strategy that systematically works through multiple combinations of parameter tunes, cross-validating as it goes to determine which tune gives the best performance.

The effectiveness of the trained classifier was then assessed using a split of 80-20 for training and testing respectively. Our model demonstrated promising results, achieving an F-1 score of approximately overall 0.91 on the testing set, indicating a high level of accuracy in the classification of Ostrom's governance rules in the context of OSS projects.

7.4 Preliminary Results

In this section, the preliminary results on patterns of adopting Ostrom's governance rules types are presented.

[2]Annotation guideline link (online version): Link

In Fig. 7.1, we present the patterns of rules getting altered in `GOVERNANCE.md` files from 2016 to 2022 across 703 GitHub projects containing a Governance.md file in their root directory (upon which the data is gathered). We show that the most frequently modified rules were 'position' and 'boundary' - the former delineating participant roles, entitlements, and duties, and the latter outlining the boundaries of the shared resource. This preponderance of alterations could be attributed to the iterative and incremental nature of OSS development and expansion. With the increasing complexity and contributor base of these projects, periodic refinement and restructuring of roles, rights, and resource boundaries might become an operational imperative, thereby necessitating continuous modifications to the 'position' and 'boundary' rules. The 'choice' rules, dictating the permissible and impermissible participant actions in relation to the resource, also displayed substantial transformation. This could signify an adaptive mechanism responding to emergent project requirements or challenges. The modifications within the 'none' category (rules not conforming to any of Ostrom's categories) might indicate the existence of a heterogeneous or flexible rule architecture, adaptively accommodating unique or non-traditional circumstances within some OSS repositories. In contrast, the 'scope' and 'information' rules demonstrated the least frequency of changes. 'Scope' rules, defining the affected outcomes of decision-making processes, and 'information' rules, regulating the information flow within the community, displayed relative stability. This could suggest that once the project scope and communication pathways are institutionalized, they are less likely to undergo major alterations. Alternatively, this could reflect the inherent stability provided by digital platforms such as GitHub, where the informational infrastructure is fairly constant, thus necessitating fewer modifications to the respective rules.

Next in Fig. 7.2, we partitioned projects into five groups based on their number of unique committers, necessitating varying levels of governance. We find that the smallest projects (Group 1: 0-20% of committers), and the intermediate ones (Group 2: 20%-40%; Group 3: 40%-60%) exhibit roughly the same median number of rule changes. This trend suggests a foundational level of governance that slightly escalates as project size grows, indicative of the fact that the addition of more participants may require modest aug-

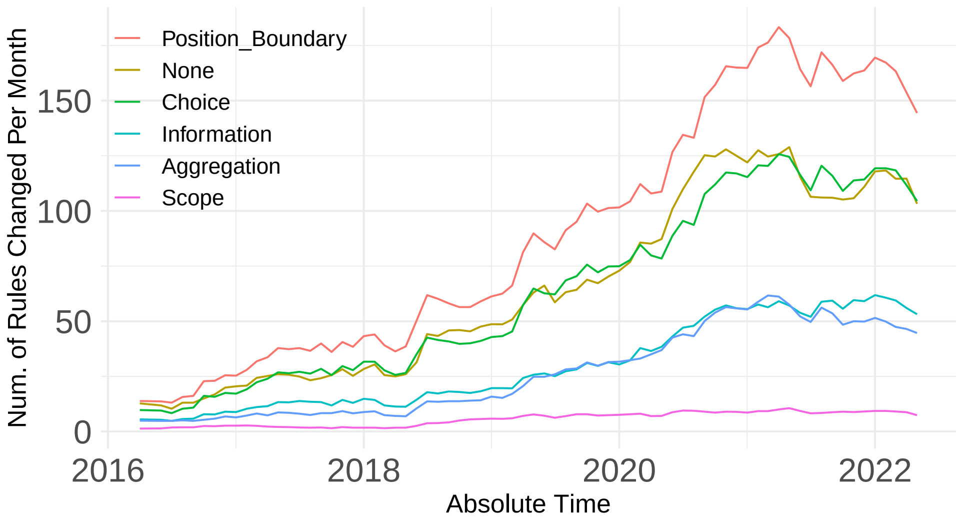
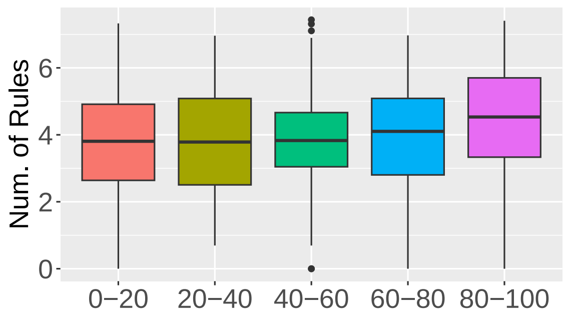
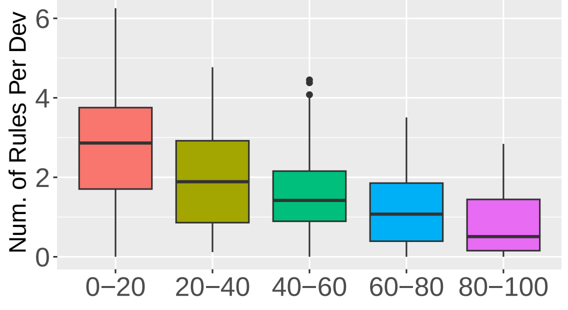


Figure 7.1. The number of changed Ostrom's rule types distribution (position rule and boundary are combined) categories on a monthly basis. The adoptions are smoothed using a 6-month moving average.



(a) Logged Number of rule adoptions across different sizes of projects.



(b) Logged Number of rule adoptions per developer across different sizes of projects.

Figure 7.2. Rule Adoption Distribution across different sizes of projects. The project size is measured in deciles by the number of unique committers in projects. All values are logged.

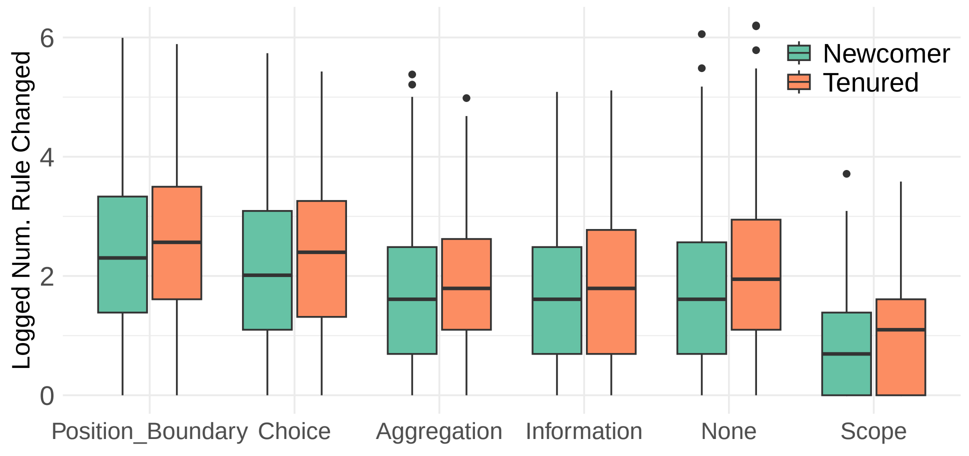


Figure 7.3. The boxplots of newcomer and tenured developers of six rule categories. The number of rules is logged. Wilcoxon test shows that the difference between newcomers and tenured committers is significant with p-values < 0.01.

mentations in rules. However, this relationship between project size and number of rule changes is sub-linear, as evidenced by Group 4 (60%-80%) and Group 5 (80%-100%) - the largest projects - which showcase a significantly higher median number of rule changes. The largest projects thus require a disproportionately larger level of governance adaptation, likely due to increased complexities and a diverse participant base. Conversely, when we adjust for the number of rule changes per developer, a significant decrease is observed from Group 1 to Group 5. This suggests that larger projects diffuse the impact of governance adaptation across a broader developer base, leading to fewer rule changes per developer. It might also indicate that larger projects, with their more established and stable governance structures, may have reduced the necessity for frequent per-developer rule changes. These findings underscore the nuanced relationship between project size and governance dynamics within open-source software repositories, emphasizing the importance of adaptable, scalable governance mechanisms in digital commons management.

Upon analyzing developer influence on governance rule changes, we categorized developers into two groups based on their commit histories: newcomers, with their first commit made within the last six months, and tenured committers, whose initial commit predates this six-month period. Given the overdispersion present in our data, we opted for the non-parametric Wilcoxon test for statistical comparison, as opposed to the t-test.

In Fig. 7.3 Our analyses revealed a notable divergence in the quantity of committed rule changes between the two categories. Across all rule types, including position, boundary, choice, aggregation, information, scope, and 'none,' newcomers demonstrated significantly fewer commits relative to their tenured counterparts, with p-values ¡ 0.01 in the Wilcoxon test. A variety of factors may account for this trend. Newcomers, due to their relatively recent integration into the project, may still be in the process of familiarizing themselves with the project's complexities and its existing governance rules, thereby contributing to their lower level of rule change commits. Additionally, their recent inclusion might limit their command over the codebase and hinder the necessary trust or authority within the community to facilitate impactful rule changes. Conversely, tenured committers, benefiting from their extensive familiarity with the project and its nuances, may possess an enhanced perception of the necessity for rule changes. Their expansive knowledge facilitates the implementation of these changes, while their established standing within the community bolsters the acceptance of proposed modifications. These findings accentuate the role of developer tenure in the interaction with governance mechanisms in open-source software repositories and further underscore the multifaceted nature of digital commons management.

7.5 Prospective Endeavors

The preliminary results presented thus far contribute to an enriched understanding of governance in OSS repositories on GitHub. However, they also open up several promising avenues for future research. (1) Longitudinal Analysis of Governance Rules. The observed trend of increased rule changes from 2016 to 2022 suggests an evolution of governance mechanisms over time. A comprehensive longitudinal study could provide more insights into how these rules evolve and adapt in response to changing technological landscapes, developer demographics, and project needs. Moreover, identifying and understanding cycles or patterns of rule changes might provide invaluable insights into the lifecycle of governance mechanisms in OSS projects. (2) Impact of Governance on Project Outcomes. While our current research has focused on the presence and evolution of governance rules, future work should examine the impact of these rules on project outcomes. Understanding how different rules or rule combinations affect project success, contributor retention, and software quality could guide the formulation of effective governance strategies. (3) Comparative Studies across Different Digital Commons. Given that the implications of our work extend beyond OSS repositories, it would be intriguing to compare governance mechanisms across different types of digital commons. This could reveal commonalities and differences in how various digital commons are managed and could aid in developing universal principles or tools for digital commons governance.

Through these suggested future endeavors, we hope to further illuminate the fascinating dynamics of governance in open-source software repositories and digital commons at large. By expanding our understanding of these mechanisms, we can inform the development of sustainable management practices in these shared digital commons.

Chapter 8

Conclusion

This chapter proffers a succinct reappraisal of each research contribution delineated and elucidated heretofore, complemented by deliberations pertaining to potential avenues for further scholarly inquiry emanating from these explorations.

In summarizing our work through the complexities of sustainability and governance in Open Source Software (OSS), it becomes clear that this dynamic ecosystem is an intricate weave of interactions, relationships, and governance structures that significantly impact the long-term viability of OSS projects. Our examination, detailed in the preceding chapters, elucidated the influential role of socio-technical networks and institutional governance in shaping OSS project trajectories and their ultimate sustainability.

8.1 Studying OSS Team Dynamics in Adopting DevOps Tools

The initial chapter delves into the vital aspect of team dynamics. This work dissects the complex social interactions among developers at a team level, and how these interactions shape project trajectories and the team's reactions to tool adoptions. The insights from this chapter lay a strong foundation for understanding the human elements in OSS. First, tool adoption impacts every team member and often depends on many of their opinions. Factors such as a member's past experiences with the tool, involvement, tenure, and sentiment play crucial roles in discussions about adoption. Therefore, tool adoption can be a multi-phase team negotiation. Moreover, the timing of tool adoption is important as there is a noticeable difference in when tools within the same use category are adopted, complicating decisions for late-adopting projects. We also discovered that teams possess specific attitudes toward certain tools, and these attitudes can shift after adoption. This suggests that tools can be either overvalued or undervalued. Successful tool adoption is likelier when multiple team members have prior experience with the tool. Prolonged discussions, while exhaustive, don't necessarily correlate with successful adoptions. However, having a strong influencer championing a tool can be beneficial.

8.2 Forecasting OSS Sustainability Using Socio-technical Networks

OSS sustainability is a critical aspect given the voluntary and often transient nature of contributions in the OSS landscape. We explore strategies for maintaining and enhancing the long-term sustainability of OSS projects. We leverage empirical data of commits and emails from Apache Software Foundation (ASF) incubator projects. The ASF, a nonprofit foundation, guides OSS projects in governance through its incubation program. By participating in the ASF incubator, projects accept some governance constraints in exchange for resources to improve sustainability. We present the first high-performing deep learning model trained on the socio-technical measures of projects with extrinsically labeled sustainability status. The forecasting result shows that socio-technical network measures have an early indication of project sustainability. By interpreting the model, we find that often times high code changes might indicate a significant shift in a project's direction, potentially prolonging its time in incubation or leading to its discontinuation. Next, we find an increased number of files in the technical network might overwhelm OSS developers affecting code quality and leading to decreased interest and engagement, further lowering the project's sustainability. To address any identified downturns in a project's trajectory, OSS developers can analyze these features from the model to adjust them accordingly, e.g., the metric clustering coefficient representing the count of triangles in the social network, can be amplified by sending emails to all participants, not just the key developers or those initiating threads. On the other hand, adopting a hierarchical, tree-like communication pattern would reduce the clustering coefficient, as this approach removes triangles. Through case studies, we show how practitioners can use the actionable insights from the forecasting model to ensure the sustainable development of OSS projects amid evolving challenges in time. However, this approach is most valid in the earlier months of a project. These early stages are crucial, as interventions can be most effective for new projects.

8.3 Investigating Temporal Patterns of Socio-technical Structure and Institutional Governance

We made the first attempt to study the inter-leaving effects of socio-technical networks and institutional discussions in a holistic way. OSS projects can be at risk due to free-riders who benefit without contributing. Echoing Ostrom's work, ASF incubator avoids such dilemmas through robust self-governance. We build the first sentence-level classifier for institutional statements, which provides a foundation for quantifying institutional analysis and design. Using the classifier, we find projects with more institutional statements are more matured in terms of self-governance, and are more likely to graduate from the ASF Incubator, emphasizing the importance of active self-governance. Next, we find projects that graduated had more diverse contributors, signaling a more involved community. Their focus on documentation further indicated a higher capacity to manage non-coding challenges. Moreover, self-governance isn't one-size-fits-all: individual projects have varied institutional approaches. Therefore, a tailored approach, based on extensive data, is crucial. We suggest that OSS projects could benefit from a separate mailing list dedicated to institutional discussions, assisting faster decision-making. Moreover, utilizing self-monitoring tools can further aid in quick responses to project challenges. Lastly, the findings offer insights into refining existing theories, emphasizing that OSS projects' structure and governance are interdependent: Adopting additional rules can help in organizing a project's actions, echoing established views on institutional governance.

8.4 Quantifying Episodic Changes and Their Effects in OSS projects

Motivated by theories of governing the commons and organizational change, we show that effective governance can induce episodic changes (the changes that are intentional, periodic, and intermittent) in the socio-technical structure, bridging the gap between efficient self-governance and sustainability. We find that graduated projects often have more extended episodic change intervals compared to their retired counterparts in the ASF incubator projects, and these changes are frequently triggered by governance-related discussions. However, such vital conversations are frequently missing during these transformative phases, indicating the need for developers to be actively involved not just in socio-technical aspects but also in governance-related discussions. These episodic changes can sometimes lead to temporary developer disengagement and a rise in negativity among project mentors and contributors. Next, we find that setting clear expectations can counterbalance potential frustrations arising from prolonged discussions, and maintaining a positive discourse can propel the project toward sustainability. Furthermore, timing these episodic changes during periods of reduced inter-team collaboration could reduce their impact. By understanding the reasons some OSS initiatives fall short of ASF incubator's expectations and examining the relationship between institutional design and socio-technical facets, practitioners can better grasp OSS sustainability. This holistic perspective, studying the temporal inter-leaving effects between socio-technical features with institutional discussions, offers a more profound insight into the sustainability of OSS projects.

8.5 Discovering Apache Incubator Project Trajectories with APEX

We introduce our online, interactive, dashboard-like tool, APEX. In previous chapters, we have shown that OSS project sustainability, particularly within the Apache Software Foundation (ASF), can be predicted from project metrics and developer interactions. The ASF, a major OSS community, offers guidelines to support the sustainability of projects.

Projects aspiring for ASF affiliation start in the Apache Software Foundation Incubator (ASFI), from where they either graduate into the ASF if deemed sustainable or are retired. A new tool, APEX, has been implemented to assist these nascent projects in tracking their sustainability over time, allowing for potential improvements in their trajectory. While developed for ASF, APEX can support projects beyond ASF due to its generic design. Although ASF has an existing monitoring tool, Clutch, APEX offers advanced analytical capabilities, emphasizing the longitudinal socio-technical dimensions of projects and suggesting actionable insights. Other analytics tools like Augur and GrimoireLab, part of the CHAOSS initiative, have similar goals but lack APEX's comprehensive approach to sustainability metrics. The APEX tool, available online to everyone, facilitates real-time monitoring and evaluation of ASFI projects and can help generate sustainability hypotheses.

8.6 Fostering Self-governance in GitHub Projects

We present a preliminary study on how OSS projects in-the-wild develop project-level governance by specifying governance rules and regulations in `Governance.md` file. This study leverages Elinor Ostrom's governance theories, which were designed for natural resources, to understand the organizational dynamics of OSS projects. We attempted to answer the following research questions: Can these traditional classes of rules be extracted from digital commons? OSS projects on GitHub are not with strict governance regulations from large organizations like ASF and Linux, projects choose to use markdown files like GOVERNANCE.md as the project's 'constitution' for defining rules and decision-making processes. OSS projects on GitHub are not subjected to more strict governance policies as their counterparts affiliated with nonprofit foundations like the Apache Software Foundation or the Linux Foundation. However, many of these GitHub projects that operate on their own and "in the wild" utilize markdown files like GOVERNANCE.md as the project's means for documenting their governance arrangements and decision-making processes. This research bridges Ostrom's governance theory with OSS settings, using `GOVERNANCE.md` files to study digital commons' governance structures. The work provides insights on applying Ostrom's principles in digital settings and introducing a classifier tool for large-scale analysis of governance in OSS projects. Moreover, this work meticulously examined large-scale GitHub repositories with GOVERNANCE.md files, focusing on the chronological introduction and modifications of governance rules. Specifically, this analysis integrated Ostrom's seven classes of rules, highlighting the untapped potential of utilizing cutting-edge machine learning tools to decode governance structures in collective-action scenarios, particularly where governance is articulated or archived in documents like the GOVERNANCE.md file. This endeavor not only offers deeper insights into the interplay of governance with the sustainability of OSS projects but also underscores its wider implications beyond just OSS research, marking a transformative direction for understanding digital commons governance in the future.

References

[1] H. Hata, T. Todo, S. Onoue, and K. Matsumoto, "Characteristics of sustainable oss projects: A theoretical and empirical study," in *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE, 2015, pp. 15–21.

[2] J. Gamalielsson and B. Lundell, "Sustainability of open source software communities beyond a fork: How and why has the libreoffice project evolved?" *Journal of systems and Software*, vol. 89, pp. 128–145, 2014.

[3] A. Martens, H. Koziolek, S. Becker, and R. Reussner, "Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms," in *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, 2010, pp. 105–116.

[4] C. Cowan, "Software security for open-source systems," *IEEE Security & Privacy*, vol. 1, no. 1, pp. 38–45, 2003.

[5] I. Stamelos, L. Angelis, A. Oikonomou, and G. L. Bleris, "Code quality analysis in open source software development," *Information systems journal*, vol. 12, no. 1, pp. 43–60, 2002.

[6] S. Stănciulescu, L. Yin, and V. Filkov, "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*, 2022, pp. 495–506.

[7] S. Wagner and M. Ruhe, "A systematic review of productivity factors in software development," *arXiv preprint arXiv:1801.06475*, 2018.

[8] M. Joblin and S. Apel, "How do successful and failed projects differ? a socio-technical analysis," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 4, pp. 1–24, 2022.

[9] S. Y. Ho and A. Rai, "Continued voluntary participation intention in firm-participating open source software projects," *Information Systems Research*, vol. 28, no. 3, pp. 603–625, 2017.

[10] M. Gharehyazie, D. Posnett, B. Vasilescu, and V. Filkov, "Developer initiation and social interactions in oss: A case study of the apache software foundation," *Empirical Software Engineering*, vol. 20, pp. 1318–1353, 2015.

[11] H. S. Qiu, A. Nolte, A. Brown, A. Serebrenik, and B. Vasilescu, "Going farther together: The impact of social capital on sustained participation in open source," in *2019 ieee/acm 41st international conference on software engineering (icse)*. IEEE, 2019, pp. 688–699.

[12] P. V. Singh, Y. Tan, and V. Mookerjee, "Network effects: The influence of structural capital on open source project success," *Mis Quarterly*, pp. 813–829, 2011.

[13] M. Guizani, A. Chatterjee, B. Trinkenreich, M. E. May, G. J. Noa-Guevara, L. J. Russell, G. G. Cuevas Zambrano, D. Izquierdo-Cortazar, I. Steinmacher, M. A. Gerosa *et al.*, "The long road ahead: Ongoing challenges in contributing to large oss organizations and what to do," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW2, pp. 1–30, 2021.

[14] L. Hannola, F. Lacueva-Pérez, P. Pretto, A. Richter, M. Schafler, and M. Steinhüser, "Assessing the impact of socio-technical interventions on shop floor work practices," *International Journal of Computer Integrated Manufacturing*, vol. 33, no. 6, pp. 550–571, 2020.

[15] I. Kwan, A. Schroter, and D. Damian, "Does socio-technical congruence have an effect on software build success? a study of coordination in a software project," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 307–324, 2011.

[16] A. Meneely and L. Williams, "Socio-technical developer networks: Should we trust our measurements?" in *Proceedings of the 33rd International Conference on Software Engineering*, 2011, pp. 281–290.

[17] F. Sarker, B. Vasilescu, K. Blincoe, and V. Filkov, "Socio-technical work-rate increase associates with changes in work patterns in online projects," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 936–947.

[18] D. Arya, W. Wang, J. L. Guo, and J. Cheng, "Analysis and detection of information types of open source software issue discussions," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 454–464.

[19] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *Proceedings of the 2006 international workshop on Mining software repositories*, 2006, pp. 137–143.

[20] F. Barcellini, F. Détienne, J.-M. Burkhardt, and W. Sack, "A socio-cognitive analysis of online design discussions in an open source software community," *Interacting with computers*, vol. 20, no. 1, pp. 141–165, 2008.

[21] R. Kaur and K. K. Chahal, "Exploring factors affecting developer abandonment of open source software projects," *Journal of Software: Evolution and Process*, vol. 34, no. 9, p. e2484, 2022.

[22] L. Yin, Z. Chen, Q. Xuan, and V. Filkov, "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*. New York, NY, USA: Association for Computing Machinery, 2021, p. 1056–1067.

[23] E. Ostrom, "Background on the institutional analysis and development framework," *Policy studies journal*, vol. 39, no. 1, pp. 7–27, 2011.

[24] F. Fleischman, B. Loken, G. Garcia-Lopez, and S. Villamayor-Tomas, "Evaluating the utility of common-pool resource theory for understanding forest governance and outcomes in indonesia between 1965 and 2012," *International Journal of the Commons*, vol. 8, no. 2, 2014.

[25] X. Basurto, S. Gelcich, and E. Ostrom, "The social–ecological system framework as a knowledge classificatory system for benthic small-scale fisheries," *Global environmental change*, vol. 23, no. 6, pp. 1366–1380, 2013.

[26] I. Anguelovski and J. Carmin, "Something borrowed, everything new: innovation and institutionalization in urban climate governance," *Current opinion in environmental sustainability*, vol. 3, no. 3, pp. 169–175, 2011.

[27] V. Kostakis, K. Latoufis, M. Liarokapis, and M. Bauwens, "The convergence of digital commons with local manufacturing from a degrowth perspective: Two illustrative cases," *Journal of Cleaner Production*, vol. 197, pp. 1684–1693, 2018.

[28] L. Yin, M. Chakraborti, Y. Yan, C. Schweik, S. Frey, and V. Filkov, "Open source software sustainability: Combining institutional analysis and socio-technical networks," *Proc. ACM Hum.-Comput. Interact.*, vol. 6, no. CSCW2, nov 2022. [Online]. Available: https://doi.org/10.1145/3555129

[29] S. Habchi, X. Blanc, and R. Rouvoy, "On adopting linters to deal with performance concerns in android apps," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018.

[30] G. Von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study," *Research policy*, vol. 32, no. 7, pp. 1217–1241, 2003.

[31] T. B. Jordan, B. Johnson, J. Witschey, and E. Murphy-Hill, "Designing interventions to persuade software developers to adopt security tools," in *Proceedings of the 2014 ACM Workshop on Security Information Workers*, ser. SIW '14. New York, NY, USA: ACM, 2014, pp. 35–38.

[32] D. Kavaler, A. Trockman, B. Vasilescu, and V. Filkov, "Tool choice matters: Javascript quality assurance tools and usage outcomes in github projects," in *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press, 2019, pp. 476–487.

[33] C. Subramaniam, R. Sen, and M. L. Nelson, "Determinants of open source software project success: A longitudinal study," *Decision Support Systems*, vol. 46, no. 2, pp. 576–585, 2009.

[34] P. M. Leonardi, "Social media, knowledge sharing, and innovation: Toward a theory of communication visibility," *Information systems research*, vol. 25, no. 4, pp. 796–816, 2014.

[35] B. Johnson, R. Pandita, J. Smith, D. Ford, S. Elder, E. Murphy-Hill, S. Heckman, and C. Sadowski, "A cross-tool communication study on program analysis tool notifications," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering.* ACM, 2016, pp. 73–84.

[36] J. Zhu, M. Zhou, and A. Mockus, "Effectiveness of code contribution: From patch-based to pull-request-based tools," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 871–882.

[37] S. Xiao, J. Witschey, and E. Murphy-Hill, "Social influences on secure development tool adoption: why security tools spread," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing.* ACM, 2014, pp. 1095–1106.

[38] J. Witschey, O. Zielinska, A. Welk, E. Murphy-Hill, C. Mayhorn, and T. Zimmermann, "Quantifying developers' adoption of security tools," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering.* ACM, 2015, pp. 260–271.

[39] D. Cornell, "Remediation statistics: what does fixing application vulnerabilities cost," *Proceedings of the RSAConference, San Fransisco, CA, USA*, 2012.

[40] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin, "A model for technology transfer in practice," *IEEE software*, vol. 23, no. 6, pp. 88–95, 2006.

[41] S. L. Pfleeger, "Understanding and improving technology transfer in software engineering," *Journal of Systems and Software*, vol. 47, no. 2-3, pp. 111–124, 1999.

[42] C. K. Riemenschneider, B. C. Hardgrave, and F. D. Davis, "Explaining software developer acceptance of methodologies: a comparison of five theoretical models," *IEEE transactions on Software Engineering*, vol. 28, no. 12, pp. 1135–1145, 2002.

[43] J. Marlow and L. Dabbish, "Activity traces and signals in software developer recruitment and hiring," in *Proceedings of the 2013 conference on Computer supported cooperative work.* ACM, 2013, pp. 145–156.

[44] F. Fagerholm, A. S. Guinea, J. Borenstein, and J. Münch, "Onboarding in open source projects," *IEEE Software*, vol. 31, no. 6, pp. 54–61, 2014.

[45] M. S. Ackerman, J. Dachtera, V. Pipek, and V. Wulf, "Sharing knowledge and expertise: The cscw view of knowledge management," *Computer Supported Cooperative Work (CSCW)*, vol. 22, no. 4-6, pp. 531–573, 2013.

[46] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *2013 International Conference on Social Computing.* IEEE, 2013, pp. 188–195.

[47] J. T. Hancock, K. Gee, K. Ciaccio, and J. M.-H. Lin, "I'm sad you're sad: emotional contagion in cmc," in *Proceedings of the 2008 ACM conference on Computer supported cooperative work.* ACM, 2008, pp. 295–298.

[48] B. C. Hardgrave, F. D. Davis, and C. K. Riemenschneider, "Investigating determinants of software developers' intentions to follow methodologies," *Journal of Management Information Systems*, vol. 20, no. 1, pp. 123–151, 2003.

[49] M. V. Zelkowitz, "Assessing software engineering technology transfer within nasa," *NASA technical report NASA-RPT-003095. National Aeronautics and Space Administration, Washington, DC*, 1995.

[50] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why don't software developers use static analysis tools to find bugs?" in *2013 35th International Conference on Software Engineering (ICSE).* IEEE, 2013, pp. 672–681.

[51] M.-A. Storey, A. Zagalsky, F. Figueira Filho, L. Singer, and D. M. German, "How social and communication channels shape and challenge a participatory culture in software development," *IEEE Transactions on Software Engineering*, vol. 43, no. 2, pp. 185–204, 2016.

[52] A. Poller, L. Kocksch, S. Türpe, F. A. Epp, and K. Kinder-Kurlanda, "Can security become a routine?: a study of organizational change in an agile software development group," in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing.* ACM, 2017, pp. 2489–2503.

[53] E. M. Rogers, "Diffusion of preventive innovations," *Addictive behaviors*, vol. 27, no. 6, pp. 989–993, 2002.

[54] L.-G. Singer, *Improving the adoption of software engineering practices through persuasive interventions.* Lulu. com, 2013.

[55] D. Bertram, A. Voida, S. Greenberg, and R. Walker, "Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams," in *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, 2010, pp. 291–300.

[56] J. Tantisuwankul, Y. S. Nugroho, R. G. Kula, H. Hata, A. Rungsawang, P. Leelaprute, and K. Matsumoto, "A topological analysis of communication channels for knowledge sharing in contemporary github projects," *Journal of Systems and Software*, vol. 158, p. 110416, 2019.

[57] B. Brehmer, "Social judgment theory and the analysis of interpersonal conflict." *Psychological bulletin*, vol. 83, no. 6, p. 985, 1976.

[58] L. Thompson and T. DeHarpport, “Social judgment, feedback, and interpersonal learning in negotiation,” *Organizational Behavior and Human Decision Processes*, vol. 58, no. 3, pp. 327–345, 1994.

[59] S. Graham and S. Golan, “Motivational influences on cognition: Task involvement, ego involvement, and depth of information processing.” *Journal of Educational psychology*, vol. 83, no. 2, p. 187, 1991.

[60] W. R. Ferrell, “Combining individual judgments,” in *Behavioral decision making.* Springer, 1985, pp. 111–145.

[61] J. Rohrbaugh, “Improving the quality of group judgment: Social judgment analysis and the delphi technique,” *Organizational Behavior and Human Performance*, vol. 24, no. 1, pp. 73–92, 1979.

[62] T. L. Kelley, “The applicability of the spearman-brown formula for the measurement of reliability.” *Journal of Educational Psychology*, vol. 16, no. 5, p. 300, 1925.

[63] S. Soroka, P. Fournier, and L. Nir, “Cross-national evidence of a negativity bias in psychophysiological reactions to news,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 38, pp. 18 888–18 892, 2019.

[64] A. Trockman, S. Zhou, C. Kästner, and B. Vasilescu, “Adding sparkle to social coding: an empirical study of repository badges in the npm ecosystem,” in *Proceedings of the 40th International Conference on Software Engineering.* ACM, 2018, pp. 511–522.

[65] G. A. A. Prana, C. Treude, F. Thung, T. Atapattu, and D. Lo, “Categorizing the content of github readme files,” *Empirical Software Engineering*, vol. 24, pp. 1296–1327, 2019.

[66] L. Christophe, R. Stevens, C. De Roover, and W. De Meuter, “Prevalence and maintenance of automated functional tests for web applications,” in *2014 IEEE International Conference on Software Maintenance and Evolution.* IEEE, 2014, pp. 141–150.

[67] M. Hilton, J. Bell, and D. Marinov, “A large-scale study of test coverage evolution.” in *ASE*, 2018, pp. 53–63.

[68] P. Skolka, C.-A. Staicu, and M. Pradel, “Anything to hide? studying minified and obfuscated code in the web,” in *The World Wide Web Conference.* ACM, 2019, pp. 1735–1746.

[69] F. R. de Souza, A. C. Domingues, P. O. Vaz de Melo, and A. A. Loureiro, “Mocha: A tool for mobility characterization,” in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems.* ACM, 2018, pp. 281–288.

[70] K. F. Tómasdóttir, M. Aniche, and A. v. Deursen, "Why and how javascript developers use linters," in *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering.* IEEE Press, 2017, pp. 578–589.

[71] J. Ruohonen and V. Leppänen, "Toward validation of textual information retrieval techniques for software weaknesses," in *International Conference on Database and Expert Systems Applications.* Springer, 2018, pp. 265–277.

[72] S. Mirhosseini and C. Parnin, "Can automated pull requests encourage software developers to upgrade out-of-date dependencies?" in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE).* IEEE, 2017, pp. 84–94.

[73] V. Cosentino, J. L. C. Izquierdo, and J. Cabot, "Findings from github: methods, datasets and limitations," in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR).* IEEE, 2016, pp. 137–141.

[74] S. Jiang, A. Armaly, and C. McMillan, "Automatically generating commit messages from diffs using neural machine translation," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE).* IEEE, 2017, pp. 135–146.

[75] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[76] J. Kottmann, G. Ingersoll, J. Kosin, and B. Galitsky, "The apache opennlp library."

[77] C. Li, Y. Lu, J. Wu, Y. Zhang, Z. Xia, T. Wang, D. Yu, X. Chen, P. Liu, and J. Guo, "Lda meets word2vec: a novel model for academic abstract clustering," in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 1699–1706.

[78] F. Jurado and P. Rodriguez, "Sentiment analysis in monitoring software development processes: An exploratory case study on github's project issues," *Journal of Systems and Software*, vol. 104, pp. 82–89, 2015.

[79] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "[journal first] sentiment polarity detection for software development," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE).* IEEE, 2018, pp. 128–128.

[80] M. R. Islam and M. F. Zibran, "A comparison of software engineering domain specific sentiment analysis tools," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER).* IEEE, 2018, pp. 487–491.

[81] S. Nakagawa and H. Schielzeth, "A general and simple method for obtaining r2 from generalized linear mixed-effects models," *Methods in ecology and evolution*, vol. 4, no. 2, pp. 133–142, 2013.

[82] C. M. Schweik and R. C. English, *Internet success: a study of open-source software commons.* MIT Press, 2012.

[83] U. Raja and M. J. Tretter, "Defining and evaluating a measure of open source project survivability," *IEEE Transactions on Software Engineering*, vol. 38, no. 1, pp. 163–174, 2012.

[84] J. C. Dueñas, F. Cuadrado, M. Santillán, J. L. Ruiz *et al.*, "Apache and eclipse: Comparing open source project incubators," *IEEE software*, vol. 24, no. 6, pp. 90–98, 2007.

[85] T. Chaikalis and A. Chatzigeorgiou, "Forecasting java software evolution trends employing network models," *IEEE Transactions on Software Engineering*, vol. 41, no. 6, pp. 582–602, 2014.

[86] E. S. Andersen, A. Dysvik, and A. L. Vaagaasar, "Organizational rationality and project management," *International Journal of Managing Projects in Business*, 2009.

[87] H.-F. Lin and G.-G. Lee, "Effects of socio-technical factors on organizational intention to encourage knowledge sharing," *Management decision*, 2006.

[88] M. Palyart, G. C. Murphy, and V. Masrani, "A study of social interactions in open source component use," *IEEE Transactions on Software Engineering*, vol. 44, no. 12, pp. 1132–1145, 2017.

[89] C. Amrit and J. Van Hillegersberg, "Exploring the impact of socio-technical core-periphery structures in open source software development," *journal of information technology*, vol. 25, no. 2, pp. 216–229, 2010.

[90] W. Sack, F. Détienne, N. Ducheneaut, J.-M. Burkhardt, D. Mahendran, and F. Barcellini, "A methodological framework for socio-cognitive analyses of collaborative design of open source software," *Computer Supported Cooperative Work (CSCW)*, vol. 15, no. 2-3, pp. 229–250, 2006.

[91] B. K. Kasi, "Minimizing software conflicts through proactive detection of conflicts and task scheduling," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 807–810.

[92] A. E. Hassan and P. C. Rigby, "What can oss mailing lists tell us? a preliminary psychometric text analysis of the apache developer mailing list," in *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*, 2007, pp. 23–23.

[93] S. Marru, L. Gunathilake, C. Herath, P. Tangchaisin, M. Pierce, C. Mattmann, R. Singh, T. Gunarathne, E. Chinthaka, R. Gardler *et al.*, "Apache airavata: a

framework for distributed applications and computational workflows," in *Proceedings of the 2011 ACM workshop on Gateway computing environments*, 2011, pp. 21–28.

[94] A. Potdar and E. Shihab, "An exploratory study on self-admitted technical debt," in *2014 IEEE International Conference on Software Maintenance and Evolution.* IEEE, 2014, pp. 91–100.

[95] I. Steinmacher, T. Conte, M. A. Gerosa, and D. Redmiles, "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*, 2015, pp. 1379–1392.

[96] I. Steinmacher, M. A. G. Silva, M. A. Gerosa, and D. F. Redmiles, "A systematic literature review on the barriers faced by newcomers to open source software projects," *Information and Software Technology*, vol. 59, pp. 67–85, 2015.

[97] K. Crowston and I. Shamshurin, "Core-periphery communication and the success of free/libre open source software projects," *Journal of Internet Services and Applications*, vol. 8, no. 1, p. 10, 2017.

[98] L. Duboc, S. Betz, B. Penzenstadler, S. A. Kocak, R. Chitchyan, O. Leifler, J. Porras, N. Seyff, and C. C. Venters, "Do we really know what we are building? raising awareness of potential sustainability effects of software systems in requirements engineering," in *2019 IEEE 27th International Requirements Engineering Conference (RE).* IEEE, 2019, pp. 6–16.

[99] V. Midha and P. Palvia, "Factors affecting the success of open source software," *Journal of Systems and Software*, vol. 85, no. 4, pp. 895–905, 2012.

[100] J. Piggott, "Open source software attributes as success indicators," *Univ. of Twente*, 2013.

[101] B. Gezici, N. Özdemir, N. Yılmaz, E. Coşkun, A. Tarhan, and O. Chouseinoglou, "Quality and success in open source software: A systematic mapping," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA).* IEEE, 2019, pp. 363–370.

[102] A. H. Ghapanchi, A. Aurum, and G. Low, "A taxonomy for measuring the success of open source software projects," *First Monday*, vol. 16, no. 8, 2011.

[103] C. Rahmani and D. Khazanchi, "A study on defect density of open source software," in *2010 IEEE/ACIS 9th International Conference on Computer and Information Science.* IEEE, 2010, pp. 679–683.

[104] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and mozilla," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 11, no. 3, pp. 309–346, 2002.

[105] J. W. Kuan, "Open source software as consumer integration into production," *Available at SSRN 259648*, 2001.

[106] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, "The rise and fall of a central contributor: Dynamics of social organization and performance in the gentoo community," in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE).* IEEE, 2013, pp. 49–56.

[107] N. McDonald and S. Goggins, "Performance and participation in open source software on github," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, 2013, pp. 139–144.

[108] J. Wu, K.-Y. Goh, and Q. Tang, "Investigating success of open source software projects: A social network perspective," *ICIS 2007 Proceedings*, p. 105, 2007.

[109] K. Crowston, J. Howison, and H. Annabi, "Information systems success in free and open source software development: Theory and measures," *Software Process: Improvement and Practice*, vol. 11, no. 2, pp. 123–148, 2006.

[110] N. Cerpa, M. Bardeen, B. Kitchenham, and J. Verner, "Evaluating logistic regression models to estimate software project outcomes," *Information and Software Technology*, vol. 52, no. 9, pp. 934–944, 2010.

[111] D. Surian, Y. Tian, D. Lo, H. Cheng, and E.-P. Lim, "Predicting project outcome leveraging socio-technical network patterns," in *2013 17th European Conference on Software Maintenance and Reengineering.* IEEE, 2013, pp. 47–56.

[112] J. Coelho and M. T. Valente, "Why modern open source projects fail," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 186–196.

[113] M. Valiev, B. Vasilescu, and J. Herbsleb, "Ecosystem-level determinants of sustained activity in open-source projects: A case study of the pypi ecosystem," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 644–655.

[114] M. S. Zanetti, "The co-evolution of socio-technical structures in sustainable software development: Lessons from the open source software communities," in *2012 34th International Conference on Software Engineering (ICSE).* IEEE, 2012, pp. 1587–1590.

[115] A. H. Ghapanchi, "Predicting software future sustainability: A longitudinal perspective," *Information Systems*, vol. 49, pp. 40–51, 2015.

[116] N. Ducheneaut, "Socialization in an open source software community: A socio-technical analysis," *Computer Supported Cooperative Work (CSCW)*, vol. 14, no. 4, pp. 323–368, 2005.

[117] E. Trist, *The evolution of socio-technical systems: A conceptual framework and an action research program.* Ontario Ministry of Labour, 1981.

[118] T. Herrmann, M. Hoffmann, G. Kunau, and K.-U. Loser, "A modelling method for the development of groupware applications as socio-technical systems," *Behaviour & Information Technology*, vol. 23, no. 2, pp. 119–135, 2004.

[119] G. Fischer and T. Herrmann, "Socio-technical systems: a meta-design perspective," *International Journal of Sociotechnology and Knowledge Development (IJSKD)*, vol. 3, no. 1, pp. 1–33, 2011.

[120] J. M. González-Barahona, L. Lopez, and G. Robles, "Community structure of modules in the apache project," in *Proceedings of the 4h International Workshop on Open Source Software Engineering.* IET, 2004, pp. 44–48.

[121] A. Smith and A. Stirling, "Moving outside or inside? objectification and reflexivity in the governance of socio-technical systems," *Journal of Environmental Policy & Planning*, vol. 9, no. 3-4, pp. 351–373, 2007.

[122] J. R. Turner and R. Müller, "Communication and co-operation on projects between the project owner as principal and the project manager as agent," *European management journal*, vol. 22, no. 3, pp. 327–336, 2004.

[123] T. Cooke-Davies, "The "real" success factors on projects," *International journal of project management*, vol. 20, no. 3, pp. 185–190, 2002.

[124] S. Wearne and A. Stanbury, "A study of the reality of project management: Wg morris and gh hough, john wiley, uk (1987) 29.95, isbn 0471 915513 pp 295," *International Journal of Project Management*, vol. 7, no. 1, p. 58, 1989.

[125] R. Joslin and R. Müller, "The impact of project methodologies on project success in different project environments," *International Journal of Managing Projects in Business*, 2016.

[126] D. W. Barclay, "Interdepartmental conflict in organizational buying: The impact of the organizational context," *Journal of Marketing Research*, vol. 28, no. 2, pp. 145–159, 1991.

[127] P. Lehtonen and M. Martinsuo, "Three ways to fail in project management and the role of project management methodology," *Project Perspectives*, vol. 28, no. 1, pp. 6–11, 2006.

[128] L. Yin, Z. Zhang, Q. Xuan, and V. Filkov, "Apache software foundation incubator project sustainability dataset," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR).* IEEE, 2021, pp. 595–599.

[129] J. Zhu and J. Wei, "An empirical study of multiple names and email addresses in oss version control repositories," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR).* IEEE, 2019, pp. 409–420.

[130] M. Wessel, B. M. De Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, "The power of bots: Characterizing and understanding bots in oss projects," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 1–19, 2018.

[131] C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu, "Putting it all together: Using socio-technical networks to predict failures," in *2009 20th International Symposium on Software Reliability Engineering.* IEEE, 2009, pp. 109–119.

[132] F. Zhang, A. E. Hassan, S. McIntosh, and Y. Zou, "The use of summation to aggregate software metrics hinders the performance of defect prediction models," *IEEE Transactions on Software Engineering*, vol. 43, no. 5, pp. 476–491, 2016.

[133] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[134] J. Friedman, T. Hastie, and R. Tibshirani, "glmnet: Lasso and elastic-net regularized generalized linear models," *R package version*, vol. 1, no. 4, 2009.

[135] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[136] M. T. Ribeiro, S. Singh, and C. Guestrin, "”why should i trust you?” explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[137] C. Casalnuovo, B. Vasilescu, P. Devanbu, and V. Filkov, "Developer onboarding in github: the role of prior social links and language experience," in *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, 2015, pp. 817–828.

[138] E. Ostrom, *Understanding Institutional Diversity.* Princeton University Press, 2009.

[139] Y. Benkler, *The wealth of networks.* Yale University Press, 2008.

[140] E. Ostrom, *Governing the commons: The evolution of institutions for collective action.* Cambridge university press, 1990.

[141] W. Blomquist *et al.*, *Dividing the waters: governing groundwater in Southern California.* ICS Press Institute for Contemporary Studies, 1992.

[142] R. L. Gruby and X. Basurto, "Multi-level governance for large marine commons: politics and polycentricity in palau's protected area network," *Environmental science & policy*, vol. 33, pp. 260–272, 2013.

[143] M. Olson, "The logic of collective action [1965]," *Contemporary Sociological Theory*, vol. 124, 2012.

[144] G. Hardin, "The tragedy of the commons: the population problem has no technical solution; it requires a fundamental extension in morality." *science*, vol. 162, no. 3859, pp. 1243–1248, 1968.

[145] C. M. Schweik and R. English, "Tragedy of the foss commons? investigating the institutional designs of free/libre and open source software projects," *First Monday*, 2007.

[146] C. Hess and E. Ostrom, *Understanding knowledge as a commons: From theory to practice.* JSTOR, 2007.

[147] B. Frischmann, M. Madison, and K. Strandburg, *Governing Knowledge Commons.* Oxford University Press, 2014.

[148] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2008, pp. 24–35.

[149] S. Crawford and E. Ostrom, "A grammar of institutions," *American Political Science Review*, vol. 89, no. 3, pp. 582–600, 1995.

[150] S. Siddiki, T. Heikkila, C. M. Weible, R. Pacheco-Vega, D. Carter, C. Curley, A. Deslatte, and A. Bennett, "Institutional analysis with the institutional grammar," *Policy Studies Journal*, 2019.

[151] T. O'Reilly, "Lessons from open-source software development," *Communications of the ACM*, vol. 42, no. 4, pp. 32–37, 1999.

[152] A. Narduzzo and A. Rossi, "The role of modularity in free/open source software development," in *Free/Open source software development.* Igi Global, 2005, pp. 84–102.

[153] S. Hissam, C. B. Weinstock, D. Plakosh, and J. Asundi, "Perspectives on open source software," Carnegie Mellon Univ Pittsburgh PA - Software Engineering Inst, Tech. Rep., 2001.

[154] G. Ropohl, "Philosophy of socio-technical systems," *Techné: Research in Philosophy and Technology*, vol. 4, no. 3, pp. 186–194, 1999.

[155] E. Ostrom, M. Janssen, and J. Anderies, "Going beyond panaceas," *Proceedings of the National Academy of Sciences*, vol. 104, no. 39, pp. 15 176–15 178, 207.

[156] A. Sen, C. Atkisson, and C. M. Schweik, "Cui bono: Do open source software incubator policies and procedures benefit the projects or the incubator?" *Available at SSRN*, 2021. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3966496

[157] A. Cohan, I. Beltagy, D. King, B. Dalvi, and D. S. Weld, “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.* Hong Kong, China: Association for Computing Machinery, 2019, p. 3693–3699.

[158] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[159] H. Yu and J. Yang, “A direct lda algorithm for high-dimensional data—with application to face recognition,” *Pattern recognition*, vol. 34, no. 10, pp. 2067–2070, 2001.

[160] R. Řehůřek, P. Sojka *et al.*, “Gensim—statistical semantics in python,” *Retrieved from genism. org*, 2011.

[161] E.-I. Dumitrescu and C. Hurlin, “Testing for granger non-causality in heterogeneous panels,” *Economic modelling*, vol. 29, no. 4, pp. 1450–1460, 2012.

[162] Y.-W. Cheung and K. S. Lai, “Lag order and critical values of the augmented dickey–fuller test,” *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 277–280, 1995.

[163] J. H. Lopez, “The power of the adf test,” *Economics Letters*, vol. 57, no. 1, pp. 5–10, 1997.

[164] J. Ferreira and A. Zwinderman, “On the benjamini–hochberg method,” *The Annals of Statistics*, vol. 34, no. 4, pp. 1827–1849, 2006.

[165] A. Ramchandran, L. Yin, and V. Filkov, “Exploring apache incubator project trajectories with apex,” in *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR).* IEEE, 2022, p. Accepted.

[166] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters, “Sustainability design and software: The karlskrona manifesto,” in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2. IEEE, 2015, pp. 467–476.

[167] K. Crowston, K. Wei, J. Howison, and A. Wiggins, “Free/libre open-source software development: What we know and what we do not know,” *ACM Comput. Surv.*, vol. 44, no. 2, mar 2008.

[168] K. A. Dawood, K. Y. Sharif, A. Zaidan, A. A. Abd Ghani, H. B. Zulzalil, and B. Zaidan, “Mapping and analysis of open source software (oss) usability for sustainable oss product,” *IEEE Access*, vol. 7, pp. 65 913–65 933, 2019.

[169] S. Sunder, “Minding our manners: Accounting as social norms,” *The British accounting review*, vol. 37, no. 4, pp. 367–387, 2005.

[170] A. D. Berkowitz, “An overview of the social norms approach,” *Changing the culture of college drinking: A socially situated health communication campaign*, vol. 1, pp. 193–214, 2005.

[171] J. A. Tainter, “Social complexity and sustainability,” *ecological complexity*, vol. 3, no. 2, pp. 91–103, 2006.

[172] T. Sandler, *Global collective action.* Cambridge University Press, 2004.

[173] E. Ostrom, “Tragedy of the commons,” *The new palgrave dictionary of economics*, vol. 2, pp. 1–4, 2008.

[174] A. H. Van de Ven and M. S. Poole, “Explaining development and change in organizations,” *Academy of management review*, vol. 20, no. 3, pp. 510–540, 1995.

[175] P. Dawson, “Reflections: On time, temporality and change in organizations,” *Journal of Change Management*, vol. 14, no. 3, pp. 285–308, 2014.

[176] P. R. Krysinski and D. B. Reed, “Organizational change and change leadership,” *Journal of Leadership Studies*, vol. 1, no. 2, pp. 65–72, 1994.

[177] R. Thomas and C. Hardy, “Reframing resistance to organizational change,” *Scandinavian Journal of Management*, vol. 27, no. 3, pp. 322–331, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0956522111000558

[178] S. K. Piderit, “Rethinking resistance and recognizing ambivalence: A multidimensional view of attitudes toward an organizational change,” *Academy of management review*, vol. 25, no. 4, pp. 783–794, 2000.

[179] A. Sharma, G. A. A. Prana, A. Sawhney, N. Nagappan, and D. Lo, “Analyzing offline social engagements: An empirical study of meetup events related to software development,” in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER).* IEEE, 2022, pp. 1122–1133.

[180] M. Cataldo, J. D. Herbsleb, and K. M. Carley, “Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity,” in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, 2008, pp. 2–11.

[181] K. Crowston and J. Howison, “The social structure of free and open source software development,” *First Monday*, vol. 10, no. 2, 2005.

[182] T. Bock, C. Hunsen, M. Joblin, and S. Apel, “Synchronous development in open-source projects: A higher-level perspective,” *Automated Software Engineering*, vol. 29, no. 1, pp. 1–53, 2022.

[183] Q. Xuan, M. Gharehyazie, P. T. Devanbu, and V. Filkov, "Measuring the effect of social communications on individual working rhythms: A case study of open source software," in *2012 International Conference on Social Informatics.* IEEE, 2012, pp. 78–85.

[184] S. Dueñas, V. Cosentino, G. Robles, and J. M. Gonzalez-Barahona, "Perceval: software project data at your will," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings.* ACM, 2018, pp. 1–4.

[185] J. Howison, A. Wiggins, and K. Crowston, "Validity issues in the use of social network analysis with digital trace data," *Journal of the Association for Information Systems*, vol. 12, no. 12, p. 2, 2011.

[186] P. Ellaway, "Cumulative sum technique and its application to the analysis of peristimulus time histograms," *Electroencephalography and clinical neurophysiology*, vol. 45, no. 2, pp. 302–304, 1978.

[187] L. Bao, X. Xia, D. Lo, and G. C. Murphy, "A large scale study of long-time contributor prediction for github projects," *IEEE Transactions on Software Engineering*, vol. 47, no. 6, pp. 1277–1298, 2019.

[188] D. Surian, D. Lo, and E.-P. Lim, "Mining collaboration patterns from a large developer network," in *2010 17th Working Conference on Reverse Engineering.* IEEE, 2010, pp. 269–273.

[189] J. M. Pérez, J. C. Giudici, and F. Luque, "pysentimiento: A python toolkit for sentiment analysis and socialnlp tasks," 2021.

[190] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[191] D. Q. Nguyen, T. Vu, and A. T. Nguyen, "Bertweet: A pre-trained language model for english tweets," *arXiv preprint arXiv:2005.10200*, 2020.

[192] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "Sentiment polarity detection for software development," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1352–1382, 2018.

[193] P. Riehmann, M. Hanfler, and B. Froehlich, "Interactive sankey diagrams," in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* IEEE, 2005, pp. 233–240.

[194] E. Ostrom and H. Nagendra, "Insights on linking forests, trees, and people from the air, on the ground, and in the laboratory," *Proceedings of the national Academy of sciences*, vol. 103, no. 51, pp. 19 224–19 231, 2006.

[195] E. Ostrom and R. Gardner, “Coping with asymmetries in the commons: self-governing irrigation systems can work,” *Journal of economic perspectives*, vol. 7, no. 4, pp. 93–112, 1993.

[196] C. M. Schweik and M. Kitsing, “Applying elinor ostrom’s rule classification framework to the analysis of open source software commons,” *Transnational Corporations Review*, vol. 2, no. 1, pp. 13–26, 2010.

[197] R. Li, P. Pandurangan, H. Frluckaj, and L. Dabbish, “Code of conduct conversations in open source software projects on github,” *Proceedings of the ACM on Human-computer Interaction*, vol. 5, no. CSCW1, pp. 1–31, 2021.

[198] X. Cui, J. Wu, Y. Wu, X. Wang, T. Luo, S. Qu, X. Ling, and M. Yang, “An empirical study of license conflict in free and open source software,” in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2023, pp. 495–505.

[199] D. S. Wilson, E. Ostrom, and M. E. Cox, “Generalizing the core design principles for the efficacy of groups,” *Journal of economic behavior & organization*, vol. 90, pp. S21–S32, 2013.

[200] L. Evans, N. Ban, M. Schoon, and M. Nenadovic, “Keeping the ‘great’in the great barrier reef: large-scale governance of the great barrier reef marine park,” *International Journal of the Commons*, vol. 8, no. 2, 2014.

[201] E. Ostrom, “Institutional analysis, design principles and threats to sustainable community governance and management of commons,” in *ICLARM Conf. Proc.*, vol. 45, 1994, pp. 34–50.

[202] E. Ostrom and M. Cox, “Moving beyond panaceas: a multi-tiered diagnostic approach for social-ecological analysis,” *Environmental conservation*, vol. 37, no. 4, pp. 451–463, 2010.

[203] D. Margan and S. Čandrlić, “The success of open source software: A review,” in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2015, pp. 1463–1468.

[204] S. O’mahony and F. Ferraro, “The emergence of governance in an open source community,” *Academy of Management Journal*, vol. 50, no. 5, pp. 1079–1106, 2007.

[205] C. M. Schweik, “Sustainability in open source software commons: Lessons learned from an empirical study of sourceforge projects,” *Technology Innovation Management Review*, vol. 3, no. 1, 2013.

[206] C. Jensen and W. Scacchi, “Governance in open source software development projects: A comparative multi-level analysis,” in *IFIP International Conference on Open Source Systems*. Springer, 2010, pp. 130–142.

[207] J. S. Baek, "Socio-technical design for resilience: A case study of designing collaborative services for community resilience," in *DS 80-8 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 8: Innovation and Creativity, Milan, Italy, 27-30.07. 15*, 2015, pp. 071–080.

[208] J. Linåker, E. Papatheocharous, and T. Olsson, "How to characterize the health of an open source software project? a snowball literature review of an emerging practice," in *Proceedings of the 18th International Symposium on Open Collaboration*, 2022, pp. 1–12.

[209] N. Schneider, "Admins, mods, and benevolent dictators for life: The implicit feudalism of online communities," *New Media & Society*, vol. 24, no. 9, pp. 1965–1985, 2022.

[210] P. N. Sharma, B. T. R. Savarimuthu, and N. Stanger, "Boundary spanners in open source software development: A study of python email archives," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2017, pp. 308–317.

[211] M. Rogiński, "Governance in peer production communities: the case of debian project leader elections," *Polish Sociological Review*, vol. 222, no. 1, 2023.

[212] Y. Yan, S. Frey, A. Zhang, V. Filkov, and L. Yin, "Github oss governance file dataset," *arXiv preprint arXiv:2304.00460*, 2023.

[213] J. A. Brzozowski, "Derivatives of regular expressions," *Journal of the ACM (JACM)*, vol. 11, no. 4, pp. 481–494, 1964.

[214] A. F. Hayes and K. Krippendorff, "Answering the call for a standard reliability measure for coding data," *Communication methods and measures*, vol. 1, no. 1, pp. 77–89, 2007.

[215] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[216] L. Tunstall, N. Reimers, U. E. S. Jo, L. Bates, D. Korat, M. Wasserblat, and O. Pereg, "Efficient few-shot learning without prompts," *arXiv preprint arXiv:2209.11055*, 2022.

[217] V. Kumar, A. Choudhary, and E. Cho, "Data augmentation using pre-trained transformer models," *arXiv preprint arXiv:2003.02245*, 2020.