Title

Towards an Empirical Understanding of Open Source Ideologies

Author

Yue, Yang

Publication Date

2024

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA,
IRVINE

Towards an Empirical Understanding of Open Source Ideologies

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Software Engineering

by

Yang Yue

Dissertation Committee:
Professor David Redmiles, Chair
Associate Professor James Jones
Professor Yi Wang

2024

DEDICATION

To my parents, Aizhi Wan and Guisheng Yue.

TABLE OF CONTENTS

Page

LIST OF FIGURES **vii**

LIST OF TABLES **viii**

ACKNOWLEDGMENTS **ix**

VITA **x**

ABSTRACT OF THE DISSERTATION **xii**

1 Introduction **1**

2 Background **7**
2.1 A Brief History of Ideology 7
2.1.1 Origin of the Concept of "Ideology" 8
2.1.2 Marx's Contribution to Ideology 9
2.1.3 Modern and Post-modern Development 10
2.2 Exploration of OSS Ideology 12
2.3 Defining OSS Ideology 15

3 Developing an Empirical Theory of OSS Ideology **17**
3.1 Research Methodology 17
3.1.1 Data Sources 18
3.1.2 Data Collection 20
3.1.3 Data Analysis 24
3.2 Empirical Theory of OSS Ideology 26
3.2.1 Norms/Values 26
3.2.2 Membership 37
3.2.3 Goals 43
3.2.4 Activities 49
3.2.5 Resources 59
3.2.6 Positions/Group Relations 65
3.3 Implications 70
3.3.1 Theoretical Implications 70
3.3.2 Practical Implications 74

4 Developing an Assessment Framework of OSS Ideology **76**
4.1 Motivation . . . 76
4.2 Designing Assessment Framework . . . 78
4.3 Pilot Case Study . . . 81

5 Applying Assessment Framework on Open Source Project: Case Study **89**
5.1 Study Design . . . 89
5.2 Three Case Studies . . . 92
5.2.1 Case 1: D3 . . . 92
5.2.2 Case 2: Zephyr . . . 95
5.2.3 Case 3: PyTorch . . . 99
5.3 Cross-Case Comparison & Implications . . . 103
5.4 Feedback from Stakeholders . . . 106

6 Discussion & Implications **109**
6.1 Theoretical Implications . . . 109
6.2 Practical Implications . . . 111
6.3 Limitations . . . 113

7 Conclusions **115**

Bibliography **117**

Appendix A Interview Protocol **127**
A.1 Interview Script . . . 128
A.1.1 Briefing and Verbal Consent . . . 128
A.1.2 Interview Questions . . . 128

Appendix B Assessment Framework for OSS Ideologies **130**
B.1 Category: Values/Norms . . . 130
B.1.1 Appreciation & Recognition . . . 130
B.1.2 Autonomy . . . 131
B.1.3 Comparative Advantages . . . 132
B.1.4 Constantly Changing & Evolving . . . 133
B.1.5 Ethics . . . 133
B.1.6 Openness & Transparency . . . 135
B.1.7 Meritocracy . . . 135
B.1.8 Reciprocity at Multiple Levels . . . 137
B.1.9 Trust . . . 137
B.1.10 Universal Accessibility & Availability . . . 138
B.2 Category: Goals . . . 139
B.2.1 Broad Impacts . . . 139
B.2.2 Built-in Product & Quality Orientations . . . 139
B.2.3 Outreach . . . 140
B.2.4 Sustainability . . . 142
B.2.5 Innovation . . . 142

B.2.6 Ubiquitous Penetration . . . 142
B.3 Category: Activities . . . 144
B.3.1 Copyright, Licensing, & Legal Implications . . . 144
B.3.2 Governance & Decision Making . . . 144
B.3.3 Personal Development . . . 145
B.3.4 Social Production . . . 146
B.3.5 Work Organization & Practices . . . 147
B.4 Category: Membership . . . 147
B.4.1 Individual Identity . . . 147
B.4.2 Ideological Leadership . . . 149
B.4.3 Interpersonal Relationships . . . 149
B.4.4 Member Hierarchy & Roles . . . 151
B.5 Category: Resources . . . 151
B.5.1 Dealing with Barriers & Restrictions . . . 151
B.5.2 Incentives, Financing, & Funding . . . 152
B.5.3 Knowledge & Expertise . . . 152
B.5.4 Supportive Facilities & Mechanisms . . . 154
B.6 Category: Position/Group Relations . . . 155
B.6.1 Interaction with Emerging Technologies . . . 155
B.6.2 Interaction with Commercial Software Development . . . 156
B.6.3 Market & Users . . . 156

Appendix C Report of Assessment on Zephyr 158
C.1 Category: Values/Norms . . . 158
C.1.1 Theme: Appreciation & Recognition . . . 158
C.1.2 Theme: Autonomy . . . 159
C.1.3 Theme: Comparative Advantages . . . 159
C.1.4 Theme: Constantly Changing & Evolving . . . 161
C.1.5 Theme: Ethics . . . 161
C.1.6 Theme: Openness & Transparency . . . 163
C.1.7 Theme: Meritocracy . . . 163
C.1.8 Theme: Reciprocity at Multiple Levels . . . 164
C.1.9 Theme: Trust . . . 164
C.1.10 Theme: Universal Accessibility & Availability . . . 165
C.2 Category: Goals . . . 165
C.2.1 Broad Impacts . . . 165
C.2.2 Built-in Product & Quality Orientations . . . 165
C.2.3 Outreach . . . 166
C.2.4 Sustainability . . . 166
C.2.5 Innovation . . . 168
C.2.6 Ubiquitous Penetration . . . 168
C.3 Category: Activities . . . 169
C.3.1 Copyright, Licensing, & Legal Implications . . . 169
C.3.2 Governance & Decision Making . . . 169
C.3.3 Personal Development . . . 170

C.3.4 Social Production 170
C.3.5 Work Organization & Practices 171
C.4 Category: Membership 171
C.4.1 Individual Identity 171
C.4.2 Ideological Leadership 172
C.4.3 Interpersonal Relationships 173
C.4.4 Member Hierarchy & Roles 173
C.5 Category: Resources 174
C.5.1 Dealing with Barriers & Restrictions 174
C.5.2 Incentives, Financing, & Funding 174
C.5.3 Knowledge & Expertise 174
C.5.4 Supportive Facilities & Mechanisms 175
C.6 Category: Position/Group Relations 175
C.6.1 Interaction with Emerging Technologies 175
C.6.2 Interaction with Commercial Software Development 176
C.6.3 Market & Users 176

LIST OF FIGURES

Page

2.1 The simplified key fundamental developments of ideology theories. 7
2.2 The top ten research focuses in the publications related to human factors in SE between 2016 and 2020 [117]. 14

3.1 Overview of the research process to investigate OSS ideology. 18
3.2 Examples of collected public speeches' snapshots (left-to-right, top-to-bottom: Tracy Hinds, Leslie Hawthorn, Guido van Rossum, and Brian Behlendorf). . 22
3.3 The overview of the empirical theory of OSS ideology 27

4.1 Three assessment types in the assessment framework. 79
4.2 The workflow of assessing Meritocracy. 82
4.3 The process of the pilot case study. 83

5.1 The process of three case studies. 90
5.2 The distribution of centrality degrees across five roles in PyTorch. 102
5.3 The commit activity heatmap of three projects. 105

LIST OF TABLES

Page

2.1 Distribution of the papers relevant to ideology in SE[117]. 14

3.1 Demographics of 22 recruited participants. 21
3.2 Lists of current† and emeritus OSI board members (as of 2022). 23

4.1 Summary of the assessment results on *Norms/Values* in pilot case study. . . 85

5.1 Summary of the collected data from project repositories (April 2023). 91
5.2 The statistical summary of the centrality degrees across five roles in PyTorch. 102
5.3 The summary of three case studies. 103

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor and committee chair, Professor David Redmiles. This dissertation would not have been possible without his guidance and patience. I feel extremely fortunate to have him as my advisor, as he has given me enormous support over the past six years.

I would like to thank my committee members, Professor Yi Wang, and Professor Jim Jones. Prof. Wang not only inspired my research ideas in this dissertation, but also provided generous help in conducting the research. The discussions with Prof. Jones were always constructive, and helped frame this dissertation. I also would like to thank additional members of my advancement committee: Professor Rebecca Black, and Professor Sergio Gago-Masagué.

The studies in this dissertation involved participants recruited from the open source communities, I would like to express my appreciation to those participants for sharing their valuable insights. I also would like to thank the Center for Organizational Research and the Department of Informatics for supporting the studies.

Throughout my years at UCI, it has been a fantastic journey with the support and help from many mentors, lab mates, and friends here. Although I cannot mention everyone, I would like to particularly thank Professor Hadar Ziv, Professor Matthew Bietz, Dr. Zhendong Wang, and Zheng Jiang (BUPT).

I would not have been able to complete my Ph.D. without the support of my family and friends in both the U.S. and China. A special thanks to my parents, Aizhi Wan and Guisheng Yue, who have given me unconditional love across the Pacific Ocean to support my adventure at UCI.

Part of the text in this dissertation is a reprint of the materials as they appear in (Yue, et al., 2021), used with permission from IEEE. Two of the co-authors listed in this publication directed and supervised research which forms the basis for the dissertation.

VITA

Yang Yue

EDUCATION

Doctor of Philosophy in Software Engineering **2024**
University of California, Irvine *Irvine, CA*

Master of Engineering in Software Engineering **2018**
Peking University *Beijing, China*

Bachelor of Engineering in Software Engineering **2015**
East China Normal University *Shanghai, China*

RESEARCH EXPERIENCE

Graduate Research Assistant **2018–2024**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

Instructor **06/2023–08/2023**
University of California, Irvine *Irvine, CA*

Teaching Assistant **09/2020–12/2023**
University of California, Irvine *Irvine, CA*

Reader **09/2018–12/2019**
University of California, Irvine *Irvine, CA*

SELECTED PUBLICATIONS

1. Y. Yue, Y. Wang and D. Redmiles, "Off to a Good Start: Dynamic Contribution Patterns and Technical Success in an OSS Newcomer's Early Career," in IEEE Transactions on Software Engineering, vol. 49, no. 2, pp. 529-548, 1 Feb. 2023, doi: 10.1109/TSE.2022.3156071.

2. Y. Yue, X. Yu, X. You, Y. Wang and D. Redmiles, "Ideology in Open Source Development," 2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), Madrid, Spain, 2021, pp. 71-80, doi: 10.1109/CHASE52884.2021.00016.

3. Y. Yue, I. Ahmed, Y. Wang and D. Redmiles, "Collaboration in Global Software Development: An Investigation on Research Trends and Evolution," 2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE), Montreal, QC, Canada, 2019, pp. 78-79, doi: 10.1109/ICGSE.2019.00028.

MANUSCRIPT IN PROGRESS

1. Y. Yue, Y. Wang, and D. Redmiles, "Towards an Empirical Theory of OSS Ideology," in Revision.

2. Y. Yue, Y. Wang, and D. Redmiles, "A Semi-automated Assessment Framework for OSS Ideology," in Preparation.

ABSTRACT OF THE DISSERTATION

Towards an Empirical Understanding of Open Source Ideologies

By

Yang Yue

Doctor of Philosophy in Software Engineering

University of California, Irvine, 2024

Professor David Redmiles, Chair

Open source development is not only a software development paradigm that is widely adopted nowadays, but also a social movement that has profound impacts in our society. Ideology, as "heart and soul", drives such social movements, and so does open source software ideology (OSS ideology) to open source movement. Thus, it is essential to understand OSS ideology. However, SE literature only conducted preliminary explorations towards OSS ideology, or only focused on particular elements within OSS ideology. In this dissertation, we report our efforts in developing an empirical understanding towards OSS ideology. First, we defined OSS ideology based on a general and explicit definition to encompass our research. Following grounded theory methodology, we collected data from 21 semi-structured interviews with OSS practitioners, and 41 videos of Open Source Initiative (OSI)'s board members' public interviews/speeches; then we developed an empirical theory of OSS ideology. Such a theory consisted of 42 themes that emerged from the collected data, under six broad categories, i.e., *Norms/Values*, *Membership*, *Goals*, *Activities*, *Resources*, and *Positions/Group Relations*. Then, based on the empirical theory of OSS ideology, we developed an assessment framework to investigate the OSS ideologies within particular open source projects. The assessment only relied on public data sources, and reflected OSS ideologies at the project level. Furthermore, we conducted three case studies by applying the assessment framework to three selected open source projects, D3, Zephyr, and PyTorch. The assessment results demonstrated

how open source projects uphold OSS ideology in practice.

This dissertation contributed to SE literature with a foundational understanding of OSS ideology. First, the empirical theory of OSS ideology, featuring *comprehensive*, *contemporary*, and *empirically-grounded* characteristics, significantly extended the SE literature with new themes identified in the study. It also provides an ideological lens to examine open source development. Second, the *practical* and *extensive* assessment framework could facilitate both OSS researchers and practitioners to investigate open source development with an ideological lens. Third, the three case studies provided insights into the practices of open source development from an ideological perspective. Moreover, the studies in this dissertation yielded rich theoretical and practical implications regarding open source movement, they also indicated various future research opportunities to further our understanding of OSS ideology and open source movement.

Chapter 1

Introduction

Open source software (OSS) is not only just a software development paradigm that is prevalent nowadays [37], but also a social movement [24] since OSS realizes the sociological "coming together" of various types of individuals and organizations to form a collective identity for certain purposes on Internet-based platforms [29, 30, 110]. It brings not only substantial changes in the software industry [41], but also has profound implications beyond the technical realm [29]. Its scalability, continuous improvements, and community-driven innovations, offer great opportunities to champion social good and help address many challenging issues such as climate change [75], human trafficking [64], pandemic outbreaks [120], etc.

Understanding a social movement should not neglect its underpinning "heart and soul" [105]. Such "heart and soul" is exactly the **ideology** guiding its membership and members' actions, its issue and agenda selections, its ideas about solutions to problems, and its choice of tactics [66]. Ideology has been recognized for its critical role in driving such social movements [77, 108, 119]. For example, according to Leveille's case study, the dynamic of the Occupy Wall Street (OWS) movement between the fall of 2011 and 2012 was determined by the movement's ideological orientations, which also connected an inner core and an outer range

of more or less active participants [70]. Similar to any other social movements, the open source software movement in the last several decades is also driven by its own ideology, namely OSS ideology.

OSS ideology could be defined as "**the basis of social representations regarding open source development shared by open source community**" [117]. These social representations enable OSS ideology to influence the OSS movement at multiple micro-, meso-, and macro-levels. At the micro-level, OSS ideology influences individuals' beliefs, preferences, decisions, and actions [18], such as choosing particular open source projects to contribute to, and when to leave a project. Then, at the project and community (meso-) level, OSS ideology also influences a team's governance participation, and dynamics [102, 26], e.g., why a project has a BDFL – Benevolent Dictator For Life, and how to achieve high productivity with voluntary contributions. Finally, at the broader societal (macro-) level, OSS ideology accounts for the emergence of open innovation systems [7, 57], or the harnessing of collective intelligence in addressing mission-critical global issues [28]. Thus, developing a deep understanding of OSS ideology would inform us how the OSS movement has grown and exhibited diversities in its practices while maintaining its core premises; and provide an evaluative framework for identifying and assessing the potential misfits among multiple participating entities' ideological orientations [9].

While the impacts of OSS ideology are ubiquitous in almost every aspect of the OSS movement, a striking fact is that relatively little research focuses on it in SE literature, in contrast to increasingly investigated many other issues in OSS [117]. Ideological elements often dispense in the literature studying motivation [46, 98], collaboration and coordination [78, 59], in a fragmented manner. One reason for the dearth of such studies might be that the term "ideology" is inherently vague, or even ambiguous, in its conceptualization. Researchers often took a convenient way of focusing on a very narrow proportion of it, e.g., the ideology of effectiveness [102]. Moreover, applying an ideological lens to investigate the OSS movement

also suffered from the absence of agreed-upon essential elements and an empirically-grounded theoretical framework overarching these elements [117].

Aiming to fill the above knowledge gaps, this dissertation reports our efforts in building a foundational understanding of OSS ideology, and gaining insights on the current OSS movement from an ideological perspective. We intend to develop a *comprehensive*, *contemporary*, and *empirically-grounded* theory of OSS ideology. We designed a qualitative grounded theory study to build the theory. During the entire grounded theory process, we interviewed 22 OSS participants with diverse backgrounds, and compiled a dataset containing 41 Open Source Initiative (OSI) [1] current and emeritus members' most recent public speeches/interviews. The data were analyzed and interpreted in an interactive coding and memo-writing procedure. A substantive theory, as a set of themes that are related to one another under the cohesive umbrella of the OSS ideology, emerged from the data. It consisted 42 themes in six broader categories: *membership*, *norms/values*, *goals*, *activities*, *position and group-relations*, and *resources*, which extends the literature significantly.

The empirical theory reflects the ideologies that are commonly shared across open source community. However, in practice, ideologies upheld by particular open source projects tend to vary, i.e., some themes of OSS ideology might be missing, while the same themes might be upheld to different extend across different projects. Literature has confirmed the existence of such heterogeneity of ideologies in OSS movement, and revealed potential impacts on open source development, i.e., the ideological misfit between contributors and open source projects could influence OSS practices [26]. Thus, an assessment system for OSS ideology tends to be a necessity, as it could benefit open source development, i.e., it could help identify the misfit, avoid potential conflicts in OSS development, and help match projects and contributors with similar OSS ideologies.

Based on the empirical theory of OSS ideology, we developed an assessment framework by

[1] OSI was established in 1998 as a steward organization for the OSS movement.

operationalizing each theme in the empirical theory. The assessment framework was designed with two principles: "focusing on open source projects," and "relying on public data source." 32 out of 42 themes in the empirical theory were successfully operationalized with qualitative or quantitative assessment methods. Furthermore, the assessment framework was refined with a pilot study on project *D3*.

The assessment framework provides a practical ideological lens to examine the current open source projects, e.g., whether open source projects follow particular themes, or how they realize the particular themes in practice. With the assessment framework, we conducted case studies to investigate how current open source projects follow OSS ideology, to gain insights on current open source movement. Three open source projects, *D3*, *Zephyr*, and *PyTorch*, were selected as study subjects. We collected the data required by the assessment framework, including repository data (commits, issues, issues comments, etc.), documentation data (tutorials, and official documents), and related news articles. Then we conducted assessments on the ideologies within each study subject by applying the assessment framework. With the assessment results, we recruited participants from three projects and collected their evaluation and feedback on the assessment results...

This dissertation develops an empirical understanding of OSS ideology, and contributes to the software engineering literature in the following ways:

- A *comprehensive*, *contemporary*, and *empirically-grounded* theory of OSS ideology under a cohesive framework that reflects the current OSS movement;
- A collection of detailed descriptions of ideological themes offering critical empirical and theoretical insights and/or opportunities for future research;
- A *practical* and *extensive* assessment framework, based on the empirical theory, with thorough workflows to conduct the analysis on OSS ideology;

- A set of implications regarding insights into the OSS movement, future research opportunities, and practices, with the ideological lens offered by the empirical understanding.

The rest of the dissertation is organized as follows:

Chapter 2 first briefly introduces the history of ideology. It provides an overview of the development of the conception of ideology in history, to understand the inherent vagueness nature of this concept. Then we derive the definition of OSS ideology from the literature to help encompass our research inquiries. In addition, we also present a few studies related to OSS ideology in SE literature, and briefly discuss the necessity of developing an empirical theory of OSS ideology.

Chapter 3 presents the study to develop an empirical theory of OSS ideology. We introduce our research process following grounded theory methodology, and present each theme in the emerged empirical theory of OSS ideology, with the empirical evidence from the analysis. Furthermore, we discuss both theoretical and practical implications derived from the empirical theory.

Chapter 4 introduces the assessment framework based on the empirical theory of OSS ideology. We present two design principles to develop the framework, and the process of defining workflows for each theme. Then we refine the assessment framework with the result of pilot study on project *D3*.

Chapter 5 reports the case study to investigate the ideologies within current open source projects by applying the assessment framework. We introduce the study design, and present the results of case study on three selected open source projects, *D3*, *Zephyr*, and *PyTorch*. We also discuss some insights into the current practice of open source movement, derived from the case study.

Chapter 6 discusses a set of implications derived from the results of the major studies in this

dissertation, including both theoretical and practical implications, and potential directions for future research on open source development. We also discussed the limitations of the studies in this dissertation.

Chapter 7 summarizes and concludes this dissertation.

Chapter 2

Background

In this chapter, we first briefly introduce the history of ideology, to understand the conception of ideology. We also derived the definition of OSS ideology to help encompass our research inquiries. Then we present a few studies related OSS ideology in SE literature, to highlight the need of developing a comprehensive theory of OSS ideology.

2.1 A Brief History of Ideology

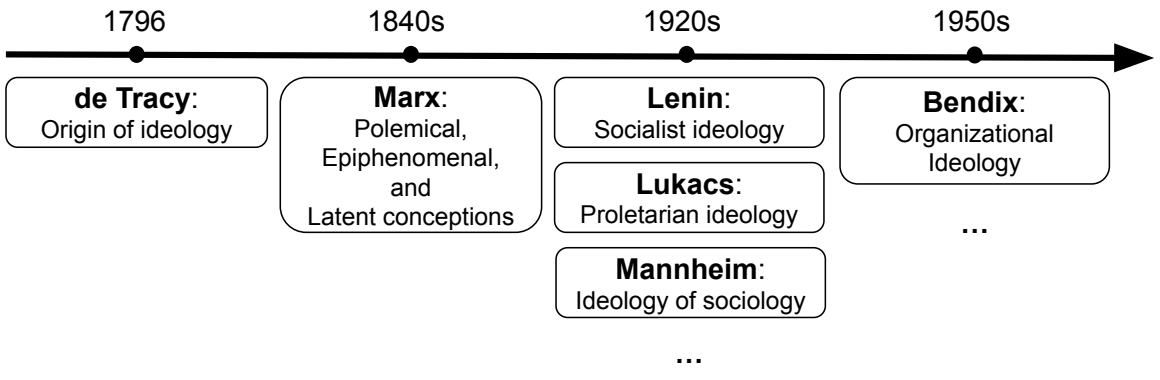

Figure 2.1: The simplified key fundamental developments of ideology theories.

The term "ideology" has a circuitous history [104], and it has been used by different people with various meanings for over two centuries, in the areas of social science, political analysis, etc. Fig. 2.1 shows some key fundamental developments of ideology theories in history.

2.1.1 Origin of the Concept of "Ideology"

French philosopher Destutt de Tracy first used the term "ideology" in 1796. He proposed a new project of systematic analysis of ideas and sensations, and argued that we could only know the ideas formed by our sensation of them, instead of the things themselves, which is our scientific knowledge. Thus, ideology at that time was a "science of ideas", originally a positive, useful, and rigorous concept [65], which de Tracy considered as "the pre-eminent science that would facilitate progression human affairs," [104] corresponding with the spirit of the Enlightenment. Later, he extended the scope of ideology to the social and political areas, analyzing the experience, feeling, thinking, etc. The focus of ideology also shifted to the ideas themselves, instead of the science of ideas at the beginning, and ideology gradually became abstract and illusory ideas [104].

However, Napoleon Bonaparte criticized the idea of ideology in the political context at the time [104]. He argued that ideology intended to determine political principles based on abstract reasoning, which would actually encourage rebellion. Thus, the concept of ideology was viewed as a political threat from republicanism, and he also blamed it for the collapse of his empire. With de Tracy's exploration and Napoleon's opposition, the term "ideology" emerged and started its circuitous history and fuzzy life.

2.1.2 Marx's Contribution to Ideology

Karl Marx played a critical and unique role in the development of the concept of ideology [109]. Marx studied de Tracy's work and Napoleon's attack on ideology, when he was in Paris. Initially, he followed Napoleon's negative view on ideology, and utilized it to criticize the ideas of the "Young Hegelians." In *The German Ideology*, Marx argued that their views are "ideological," and they overestimated the value and role of ideas in history and social life [76]. Consequently, the polemical conception of ideology was derived, and was defined as "a theoretical doctrine and activity which erroneously regards ideas as autonomous and efficacious and which fails to grasp the real conditions and characteristics of social-historical life." [104] Although this conception is a negative view inherited from Napoleon, Marx developed it beyond the original scope, where the ideas should be emerged based on the practice of activity, instead of autonomy.

Then Marx, along with Engels, considered the concept of ideology from a more general perspective, analyzing the production and the relations between classes. They intended to empower the ideology a more general role in the socio-historical analysis, and defined the epiphenomenal conception, which referred as "a system of ideas that expresses the interests of the dominant class but represents class relations in an illusory form." [104]. Thus ideology serves a systematic role in Marx's theoretical framework, and generally reflects "the ideals of the ruling class." In his well-known work *Manifesto of the Communist Party*, Marx explained the socio-economic transformation, unmasked the ideological forms of consciousness of the dominant class in the society, and anticipated the victory of proletariat ideology, with the demise of bourgeois ideology.

Furthermore, in his work, Marx also analyzed symbols in society. He intended to characterize the phenomena of symbolic life in society, such as slogans and traditions. These symbols, even though not abstract enough, play a significant role in orienting people to specific di-

rections. Thus, he developed the latent conception of ideology, referred as "a system of representations which serves to sustain existing relations of class domination by orientation individuals towards the past rather than the future, or towards images and ideals which conceal class relations and detract from the collective pursuit of social change." [104]

The concept of ideology had a giant leap and advanced to a higher level with the unique contribution of Marx, while it also gradually became abstract with various conceptions.

2.1.3 Modern and Post-modern Development

Since Marx, the concept and discussion of ideology are widely viewed in terms of its roots in individual and group interests. It has been used in many fields of social science, and has been developed with rich interpretations, not just limited to Marxism. As Marx took over the concept of ideology from Napoleon, it had a negative sense in most of his work. One of the major tendencies in modern and post-modern development, however, was neutralizing the concept of ideology [104].

The neutralization was first started by some Marxists when Marxist social movements in Europe had challenges in particular social-historical circumstances. Lenin faced the polarized political situation in Russia, and he intended to combat the bourgeois ideology and realize the revolution, thus he elaborated "socialist ideology". Lukaćs, facing a similar situation in the working-class movement, emphasized the importance of "proletarian ideology", which both helped the proletarian class understand their situation and class consciousness [104]. In both Lenin's and Lukaćs's work, the concept of ideology generally remains consistent, since it was inherited from Marx's theory. However, their conceptions tend to be generalized–the ideas to express the interests of the major classes, not associating with a strong negative sense [104].

Besides Marxists, other scholars also contributed to the neutralization process, and they covered a wide spectrum in politics, social science, philosophy, etc. Karl Mannheim's work is representative of them. Different from Marxists, Mannheim did not view ideology as a weapon of a class for the revolution. Instead, he considered it as a method of research in social and intellectual history, which is similar to de Tracy's idea, referring to "the sociology of knowledge" [104]. He intended to use his new approach to analyze all the factors that influence thought or ideas in the social context, which was called the general formation of the total conception of ideology. Thus, the conception of ideology in his theory is "the interwoven systems of thought and modes of experience which are conditioned by social circumstances and shared by groups of individuals, including the individuals engaged in ideological analysis."[104] Moreover, in *La production de l'idéologie dominate*, Pierre Bourdieu and Luc Boltanski offered critical insights into the principal characteristics and functions of ideologies in advanced capitalist societies [13]. For example, they argued that ideology's primary function is to "orient an action or a set of action," which indicated the shift from ideology's theoretical dimensions to practical dimensions in real-life situations.

Since the late 1950s, the concept of ideology started gaining popularity in many social science fields [51], and one strand of work is usually referred to as "organizational ideology." Reinhard Bendix studied the social relations in four centuries to identify the nature, origins, and consequences of "managerial ideologies," an early notion of "organizational ideology." He defined it as "all ideas which are exposed by or for those who seek authority in economic enterprises, and which seek to explain and justify that authority." [8] Under the umbrella of organizational ideology, researchers have used the theoretical lens of ideology to study many aspects of organizations at both individual and organizational levels [47, 55, 63, 74, 115], to name a few, organizational cohesion, performance, structure, innovation, social responsibility, knowledge sharing, individual social action, engagement, and so on.

In summary, the concept of ideology has been embodied with rich interpretations by various

scholars in history. It tends to be general and abstract, but also vague and ambiguous at times. However, it becomes clear that the contemporary uses of ideology have already deviated from the original emphasis on conflicts and tensions among social classes, and come to have considerable overlap with its social psychological impacts on individuals and groups, and their actions.

2.2 Exploration of OSS Ideology

Researchers have been studying the OSS movement and exploring its ideology since its inception. Scholars from multiple disciplines advanced OSS ideology theories. Ljungberg defined OSS ideology from two dimensions [73], One dimension was zealotry, i.e., people believe that OSS is either a new way of life, or a good way to build software. The other dimension was the degree of hostility to commercial software. Steward and Gosain developed the three-tent (beliefs, values, and norms) framework by combining famous advocates' narratives and the prior literature [102]. For example, "open source development methods produce better code than closed source" is a belief in their framework. However, their framework is preliminary and far from comprehensive; even they admitted that [102]. Researchers further applied these frameworks to study OSS ideology's impacts. At the project level, OSS ideology could affect the effectiveness in attracting and retaining contributors, the production of project outputs, and so on [102]. At the individual level, the misfit of OSS ideology between contributors and projects could influence contributors' commitment [26]. For example, the under-fit of OSS ideology–contributors embrace OSS ideology more than their projects do–would lower contributors' commitment.

Besides, researchers also studied various aspects of OSS ideology, but in an implicit and fragmented manner. At the individual level, researchers first investigated the motivations to participate in open source development [11, 46, 112, 94]. Recently, monetary incentives'

effects in motivating contributors were also examined [122]. Then individuals' activities in OSS projects were also studied by researchers, to name a few, communication and social learning [127], contributing and evaluating [42, 48, 107], maintainers' invisible work [45]. Contributors' improper or unethical behaviors and such behaviors' impacts recently received much attention, particularly uncivil comments, and toxic language [39, 87].

Moreover, many project/community-level phenomena have been related to ideology. As the fundamental element of ideology, values continuously received researchers' attention, for example, transparency [25], diversity [12, 111, 15], and code of conduct enforcing these values [72]. Some work extended the focus to individual and group activities that are guided by values. These activities include collaboration and coordination [59, 67, 78], licensing [21, 41], governance & decision making [40, 59, 79], and knowledge sharing [58, 118]. From a structural perspective, researchers deconstructed how individuals form communities. They found that different roles of contributors consist of a centralized, layer-upon-layer structure [60], with pathways to allow leadership to emerge [106, 52].

Furthermore, we conducted a brief literature survey on the studies related to ideology in software engineering. First, papers from four top-tier conferences and journals in software engineering (SE) were selected, i.e., the International Conference on Software Engineering (ICSE), the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), the IEEE Transactions on Software Engineering (TSE), and the ACM Transactions on Software Engineering and Methodology (TOSEM). Regarding ICSE, we included the main track, as well as the Software Engineering in Practice (SEIP), and the Software Engineering in Society (SEIS) tracks, since the papers in those tracks often focus on human or social aspects in SE, which potentially related to ideology in SE; regarding ESEC/FSE, we only considered papers from main track; for both TSE and TOSEM, we included all published papers. In total, 1,698 papers published in those venues from 2016 to 2020 were selected. Then two researchers manually reviewed

and screened the selected papers as relevant or not relevant to ideology, based on the title, abstract, and keywords, and jointly resolved the disagreements. Finally, 119 papers were relevant to ideology, as shown in Tab. 2.1.

Table 2.1: Distribution of the papers relevant to ideology in SE[117].

Venue & Track	**2016**	**2017**	**2018**	**2019**	**2020**	**Total**
ICSE-Main Track	4	4	10	7	13	38
ICSE-SEIP	2	2	6	1	1	12
ICSE-SEIS	4	3	3	4	6	20
ESEC/FSE-Main Track	6	2	4	3	10	25
TSE	2	3	0	5	8	18
TOSEM	0	1	0	2	3	6
Total	18	15	23	22	41	119

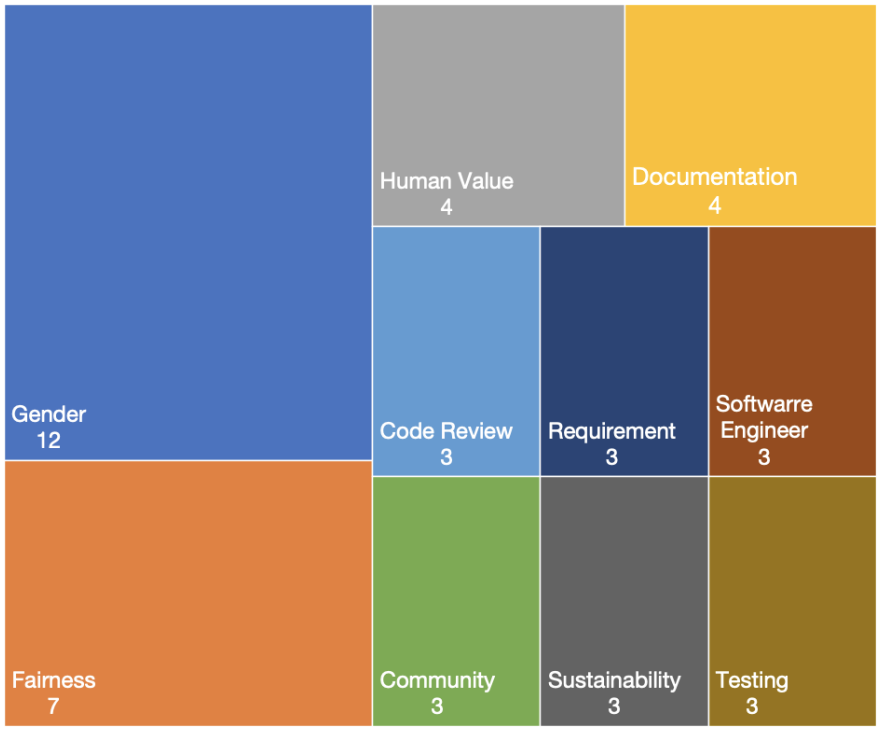

Figure 2.2: The top ten research focuses in the publications related to human factors in SE between 2016 and 2020 [117].

With the selected papers, we further analyzed their research focuses. In general, none of those papers explicitly mentioned the notion of "ideology", nor defined the ideology in SE. However, those papers touched on some elements related to OSS ideology, and the

research interests were increasing [24, 117]. The research interests often focused on ten themes: *gender*, *fairness*, *human value*, *documentation*, *code review*, *requirement*, *software engineer*, *community*, *sustainability*, and *testing* [117], as shown in Fig. 2.2. However, most studies tended to be knowledge fragments that focus on particular aspects, and there were few efforts to integrate them into a coherent body [24]. One of the reasons could be the lack of an explicit definition of OSS ideology [117]. Moreover, the samples in these studies often focused on high-profile OSS projects and famous technical icons, and almost half of them were skewed toward a single project [24]. That potentially overlooked the voice of grassroots and non-technical contributors in the open source movement [117]. Therefore, we first derive an explicit definition of OSS ideology. Then we collect data from both first-line OSS participants and OSI board members, and build an empirical theory of OSS ideology based on the collected data.

2.3 Defining OSS Ideology

The first step to investigating any social construct is to define it in its context. Considering the vagueness of the concept of ideology, it is necessary to explicitly define OSS ideology to help encompass the research inquiries. We derived the definition of OSS ideology based on van Dijk's work [109], as

Definition 2.1 (OSS Ideology)**.** *The basis of social representations regarding open source development shared by open source community.*

This definition also indicates two ways to approach OSS ideology, i.e., cognitive and social ways. First, ideology characterizes the **basis of the social representations** in the context of open source movement, such as ideas and beliefs related, terms used in cognitive science. They reflect how open source participants perceive open source development. Then, the

social way reflects that ideology emerges within **social groups**, through the interaction and communication among members of the open source community, e.g., what kind of people could become members of the open source community. When most of the members reach an agreement on particular things regarding open source development, these usually become part of the ideology. In addition, ideology is the things that are shared in common by the whole community, and reflects the common interests of the open source community [109].

However, the two ways are neither opposite nor separate when approach ideology. Indeed, cognitive and social ways define ideology at different levels. Individuals' social representations (ideas and beliefs) characterize ideology at the micro-level, while the interests of the social groups capture ideology at the meso/macro-level. The individual's ideas and beliefs could be considered as an initial set of ideology; then they interact with others who also have similar ideas and beliefs, and later they could reach an agreement and share the common interests as a whole group. Hence, according to its definition, OSS ideology is intertwined with both cognitive and social ways, which is how we approach and develop an empirical theory of OSS ideology.

Chapter 3

Developing an Empirical Theory of OSS Ideology

3.1 Research Methodology

The study design followed grounded theory methodology [17]. First, there was very limited extant theory examining the ideologies in the OSS movement [117]. Second, empirical evidence related to OSS ideologies was also lacking. Under such circumstances, grounded theory entailed the discovery of concepts from data in an inductive process [17], allowing the generation of novel and accurate insights into the topic under study. Moreover, the grounded theory utilizes an iterative process of data collection and analysis, such a process makes the theory falsifiable, dependent on context, and never completely final [49]. Thus, we could approach OSS ideology by learning from OSS participants' experiences and reflections within particular contexts, rather than the intent to measure something against pre-existing theories. Fig. 3.1 provides an overview of the whole research process to investigate OSS ideology.

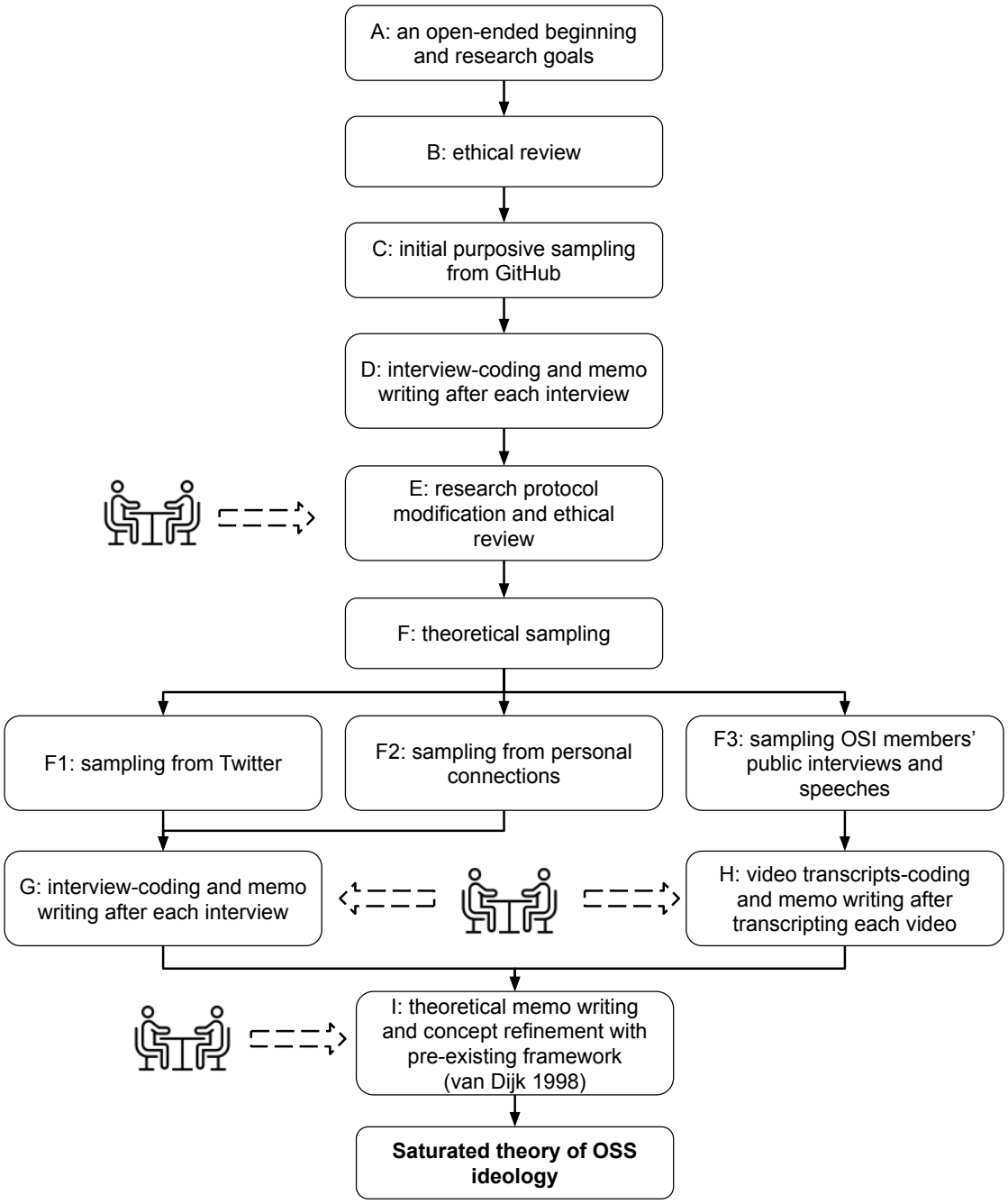

Figure 3.1: Overview of the research process to investigate OSS ideology.

3.1.1 Data Sources

We followed an iterative process of *theoretical sampling* to collect data [17]. Data was collected based on the gaps in the emerging theory identified in data analysis [17, 22]. The

data was collected from two sources: (1) interviews with OSS practitioners, and (2) video-recorded public interviews or speeches of current and former OSI board members. Both constituted legitimate data sources in grounded theory methodology, since grounded theory accommodates "interviews, field observations, documents, videos, etc." [22].

The two data sources were not chosen randomly. In particular, with the progress of the interviews and data analysis, we noticed that most interviewees were technical staff, and their experiences were hardly beyond the project level. Thus, such a sample was insufficient for our theory development. Meanwhile, the potential informants representing non-technical aspects of OSS, such as legal counsels, and high-level leaders, were often hard to access directly to arrange interviews. Therefore, we resorted to secondary data sources.

After careful consideration, we chose the video recordings of public interviews or speeches of OSI board members on YouTube and other video-sharing sites. Those board members were often renowned contributors or highly respected individuals in the OSS world and leading proponents of the OSS movement, e.g., Josh Berkus was well known for his contributions to multiple high-profile projects, such as Linux and PostgreSQL. This secondary data source had three advantages. First, OSI board members consisted of a wide range of OSS participants. Many of them were opinion leaders, activists, and lawyers, thus forming a good complementarity with the interviews, as well as satisfying grounded theory's requirement of theoretical saturation [17, 22]. Second, public interviews and speeches of this sample often contained rich ideology-related content and influenced many OSS participants. Third, these videos were available to the public, either falling within the public domain or allowing for fair use, without incurring any additional ethical or copyright concerns.

3.1.2 Data Collection

The data collection started with initial purposive sampling (Step C in Fig. 3.1). We recruited participants and had interviews with each of them. Then we had another iteration of data collection, i.e., theoretical sampling (Step F in Fig. 3.1), based on the analysis of the previous collected data. The theoretical sampling included recruiting more participants for interviews (Step F1 & F2 in Fig. 3.1), and collecting secondary data (Step F3 in Fig. 3.1).

Participants Recruitment and Interview

We recruited participants for interviews in a three-wave process. The first wave (Step C in Fig. 3.1), as the initial purposive sampling, recruited participants from GitHub through emails. We sampled 20 open source projects from multiple domains and of multiple sizes on GitHub, and collected their contributors' emails ($N = 540$) from their public profiles if existed. Then we sent emails to invite potential participants, and 19 of them expressed willingness to join the study (response rate: 3.5%). In the second wave, we shared the recruitment post on X/Twitter to reach broader audiences. Those who signed up were screened and selected based on their responses. However, most of them either lacked OSS experiences, or were scammers, except one eligible participant. The third wave focused on the research team's contacts who were veteran or active OSS practitioners. We asked them if they would like to participate, and two agreed to have interviews. In total, we recruited 22 participants from 14 countries around the world, as shown in Tab. 3.1. They had diverse backgrounds in jobs, experiences, and other demographic factors.

With each participant, we had semi-structured, online interviews to collect data [1]. Each interview was around 30 minutes. First, we asked questions about their demographics and background in software development. The second part was used to collect their personal

[1]The interview protocol is in Appendix A.

Table 3.1: Demographics of 22 recruited participants.

#	Occupation	Country	Experience in SE (in Years)	Experience in OSS (in Years)
P1	Software Engineer/Freelancer	Australia	30	16
P2	Undergraduate Student	India	3	0.5
P3	Graduate Student	Spain	1	0.5
P4	Software Engineer	US	25	22
P5	Senior Manager	US	30	10
P6	Technical Support Engineer	Spain	3	3
P7	Senior Software Engineer	US	12	8
P8	Senior Software Engineer	US	8	3
P9	Technical Solution Engineer	Spain	10	6
P10	Lead Software Engineer	Ireland	5	4.5
P11	QA Engineer	US	9	9
P12	Graduate Student	Austria	10	10
P13	Development Lead	Switzerland	32	27
P14	Software Engineer	Austria	20	10
P15	Graphic Designer	US	10	6
P16	Freelancer/Consultant	Germany	N/A	20
P17	Back-end Developer	Brazil	5	5
P18	Software Engineer	Russia	9	7
P19	Senior Software Engineer	Armenia	5	6
P20	Front-end Engineer	Iran	8	3
P21	Senior Software Engineer	Norway	10	10
P22	Computer Engineer	Columbia	20	16

opinions and stories about the OSS development. We asked a series of open-ended questions, such as *"What is your understanding of open source?"*, *"What makes you start the contribution in open source projects?"*, and so on. Based on their narratives, we asked some follow-up questions for more details, clarifications, and provoking further reflections. Moreover, the participants also had the freedom to share anything relevant to OSS. Upon the participants' consent, all their narratives were recorded or noted; then, these narratives were transcribed as interview data for further analysis. In total, we collected 635 minutes of interviews.

Collecting Secondary Video Data

As shown in Fig. 3.1, video data collection was performed after the first wave of interviews started. Note that we had decided to use OSI board members' public interviews/speeches on video-sharing sites (mostly YouTube) as the secondary data source, and the usage of these videos constituted "fair use" under copyright law. First, we compiled a list of 51 people, including both 10 current OSI board members and 41 emeritus members from OSI's website. Then, we searched online for their most recent public interviews or speeches related to open source. In total, 41 of them had such videos available online, as listed in Tab. 3.2, and Fig. 3.2 are some snapshots of the example videos we collected. These videos were transcribed for further analysis.

Figure 3.2: Examples of collected public speeches' snapshots (left-to-right, top-to-bottom: Tracy Hinds, Leslie Hawthorn, Guido van Rossum, and Brian Behlendorf).

Table 3.2: Lists of current† and emeritus OSI board members (as of 2022).

No.	Name	Link
1	Catharina Maracke†	https://www.youtube.com/watch?v=XrDfZaoesj0
2	Thierry Carrez†	https://www.youtube.com/watch?v=svaCMh7SYpc
3	Amanda Brock†	https://www.youtube.com/watch?v=c5pYZJEM5Vk
4	Justin Colannino†	https://www.youtube.com/watch?v=WDjh_nbAAeg
5	Pamela Chestek†	https://www.youtube.com/watch?v=HAc-wSAW3Ig
6	Hong Phuc Dang†	https://www.youtube.com/watch?v=otmDmwVmKbM
7	Aeva Black†	https://www.youtube.com/watch?v=gTR_AtJVXkw
8	Tracy Hinds†	https://www.youtube.com/watch?v=Z7HCxsFJkoA
9	Josh Berkus†	https://www.youtube.com/watch?v=WnOKjfe3TUM
10	Carlo Piana†	https://www.youtube.com/watch?v=W8za1gOlIWY
11	Eric Raymond	https://www.youtube.com/watch?v=5e0FNskVeRk
12	Bruce Perens	https://www.youtube.com/watch?v=08Xxaaijdgk
13	Matt Asay	https://www.youtube.com/watch?v=iBQ1HL7ia2g
14	Brian Behlendorf	https://www.youtube.com/watch?v=oy2NWfBbfaI
15	VM (Vicky) Brasseur	https://archive.org/details/camundameetup2022-copyrightlicensingbasics
16	Deborah Bryant	https://www.youtube.com/watch?v=YbjO4Jf7sPQ
17	Ken Coar	https://www.youtube.com/watch?v=900T8_WjJx8
18	Danese Cooper	https://www.youtube.com/watch?v=WjPBz5QhQCQ
19	Chris DiBona	https://www.youtube.com/watch?v=wJ5gW0s5olE
20	Karl Fogel	https://www.youtube.com/watch?v=sZnZMow20hk
21	Richard Fontana	https://www.youtube.com/watch?v=ncXa0LFeQY0
22	Rishab Aiyer Ghosh	https://www.youtube.com/watch?v=O0B9bfPEsj8
23	Leslie Hawthorn	https://www.youtube.com/watch?v=dM0F961xze4
24	Joichi Ito	https://www.youtube.com/watch?v=Bkd29EADdpw
25	Jim Jagielski	https://www.youtube.com/watch?v=tipdkR3dr_U
26	Patrick Masson	https://www.youtube.com/watch?v=Y9GkMCod97Q
27	Martin Michlmayr	https://www.youtube.com/watch?v=WNw-aq8qGiw
28	Mike Milinkovich	https://www.youtube.com/watch?v=rltdJ5TNorI
29	Ian Murdock	https://www.youtube.com/watch?v=nyL2Fx0iNZ0
30	Russell Nelson	https://www.youtube.com/watch?v=kHEk7pRXTWU
31	Nnenna Nwakanma	https://www.youtube.com/watch?v=I2lNgUdGXIw
32	Simon Phipps	https://www.youtube.com/watch?v=oOYiXGYTq2E
33	Carol Smith	https://www.youtube.com/watch?v=kzHwc19p3jI
34	Bruno Souza	https://www.youtube.com/watch?v=JYJVQhYbIb8
35	Michael Tiemann	https://www.youtube.com/watch?v=Y51FDukxSKA
36	Guido van Rossum	https://www.youtube.com/watch?v=7kn7NtlV6g0
37	Italo Vignoli	https://www.youtube.com/watch?v=2pMKed601ro
38	Luis Villa	https://www.youtube.com/watch?v=d7YE2TI9eDg
39	Tony Wasserman	https://www.youtube.com/watch?v=hgn3tFymDOM
40	Sanjiva Weerawarana	https://www.youtube.com/watch?v=jmDQW8s3Cjs
41	Stefano Zacchiroli	https://www.youtube.com/watch?v=oQ3y695SPXM

3.1.3 Data Analysis

The data analysis process started immediately after some data was collected (Step D, G, & H in Fig. 3.1), rather than waiting until all the data was gathered. Both interview data and secondary video data were treated equally in the analysis. The theory emerged during data analysis and formed the basis of further theoretical sampling (Step F in Fig. 3.1), so we need to go back and forth between data collection and data analysis. Such iterative process of data collection and data analysis stopped when *Theoretical Saturation* was reached [80].

The data analysis started with *Open Coding* when provisional codes were assigned to them based on the meanings identified by the authors [22, 17]. Concepts that delineated OSS ideology in the data emerged in this step. Some concepts were grouped together as higher-level categories, and they would be refined in the following steps. Two researchers coded the data independently but met virtually almost every day to discuss the codes and resolve disagreements (marked with meeting icons in Fig. 3.1). Thus, the notion of inter-rater reliability was not applicable; rather, the two researchers came to an agreement with most codes. In some rare cases of lack of agreement on certain codes, the researchers chose an inclusive attitude regarding each other's proposed codes, postponing the validation of the codes until the time when more data were collected and analyzed [95]. Doing so helped expose each researcher's individual interpretive act as early as possible, thus avoiding potential misunderstandings and inconsistencies resulting from coders' different theoretical backgrounds to be real, long-lasting threats [32].

In *Axial Coding*, the identified concepts and categories were put together and examined again to find the connection or relations among them. Based on these connections or relations, we could cluster some concepts together as categories. Then we further identified the properties and dimensions of these categories.

After that, we returned to the data again, and re-coded them with these concepts that

emerged from the data. *Constant Comparisons* were performed from the beginning, and through the entire process of data analysis. First, the emerging theory of OSS ideology was repeatedly compared with old and new data. the gaps were identified and provided guidance for theoretical sampling to collect more data. When there were no new concepts emerged from the new data, *Theoretical Saturation* was achieved.

The *Selective Coding* focuses on important categories and concepts, without considering the others. The selection was based on the frequency of codes' occurrences or the patterns that appeared in the data. For example, some concepts that were frequently mentioned in multiple interviews might be more important. Meanwhile, since an OSS ideology theory was desired, categories and concepts less relevant to OSS ideology might not be important. The process of selective coding resulted in a set of critical categories, as well as an in-depth understanding of them.

We further finalized theory building through *Theoretical Memo Writing and Concept Refinement* (Step I in Fig. 3.1). It was the pivotal step in grounded theory [22, 17]. With the space created through theoretical memo writing, we compared the data, codes, categories, and concepts in our analysis, and focused on the theoretical propositions linking them together [80]. Then, we refined the emerging categories and concepts. When the collected data well supported all the emerging categories and concepts, and new data could not provide any further refinements, *Theoretical Saturation* was thus reached, which indicated the collected was sufficient, and we could end the iterative process of data collection and analysis [80]. Moreover, instead of excluding any literature, we reused a pre-existing theoretical framework proposed by van Dijk [109] to facilitate the theory development in this step, guided by [17, 80]. We discussed the emerging categories and concepts, sorted them, and mapped them into the framework. We paid particular attention to ensuring that the framework would be adapted to fit these categories and concepts well, rather than altering the categories and concepts to match the framework. Finally, the empirical theory of OSS ideology emerged

from the analysis.

3.2 Empirical Theory of OSS Ideology

Fig. 3.3 provides an overview of the empirical theory of OSS ideology that emerged from the data. In general, the empirical theory consists of 42 themes in six key categories, i.e., *norms/values*, *membership*, *goals*, *activities*, *positions/group relations*, and *resources*; the code frequency of each theme is also listed. In the remainder of this section, we discuss the identified themes by categories. Note that some themes tend to be straightforward and have been well covered by the extant literature, for example, *Individual Motivation* has been well-examined in [11, 46, 94, 98]. Therefore, when discussing each category, we focus on key themes that either have the potential to offer new empirical and theoretical insights or opportunities for future research.

3.2.1 Norms/Values

Norms/values are commonly defined as guiding principles in people's lives. In addition to the functions at the individual level, human norms/values' social-cultural nature makes them be shared, known, and applied by members in a large variety of OSS practices. The literature on values/norms is vast [90]. For example, values were defined as guiding principles in people's lives in Schwartz's theory of universal content and structure of human values [97]. Our data analysis revealed 14 themes related to norms/values in OSS movement, i.e., (1) *Altruism*, (2) *Appreciation & Recognition*, (3) *Autonomy*, (4) *Comparative Advantages*, (5) *Constantly Changing & Evolving*, (6) *Ethics*, (7) *Freedom*, (8) *Historical Heritage*, (9) *Meritocracy*, (10) *Openness & Transparency*, (11) *Reciprocity at Multiple Levels*, (12) *Reflection & Improving*, (13) *Trust*, and (14) *Universal Accessibility & Availability*.

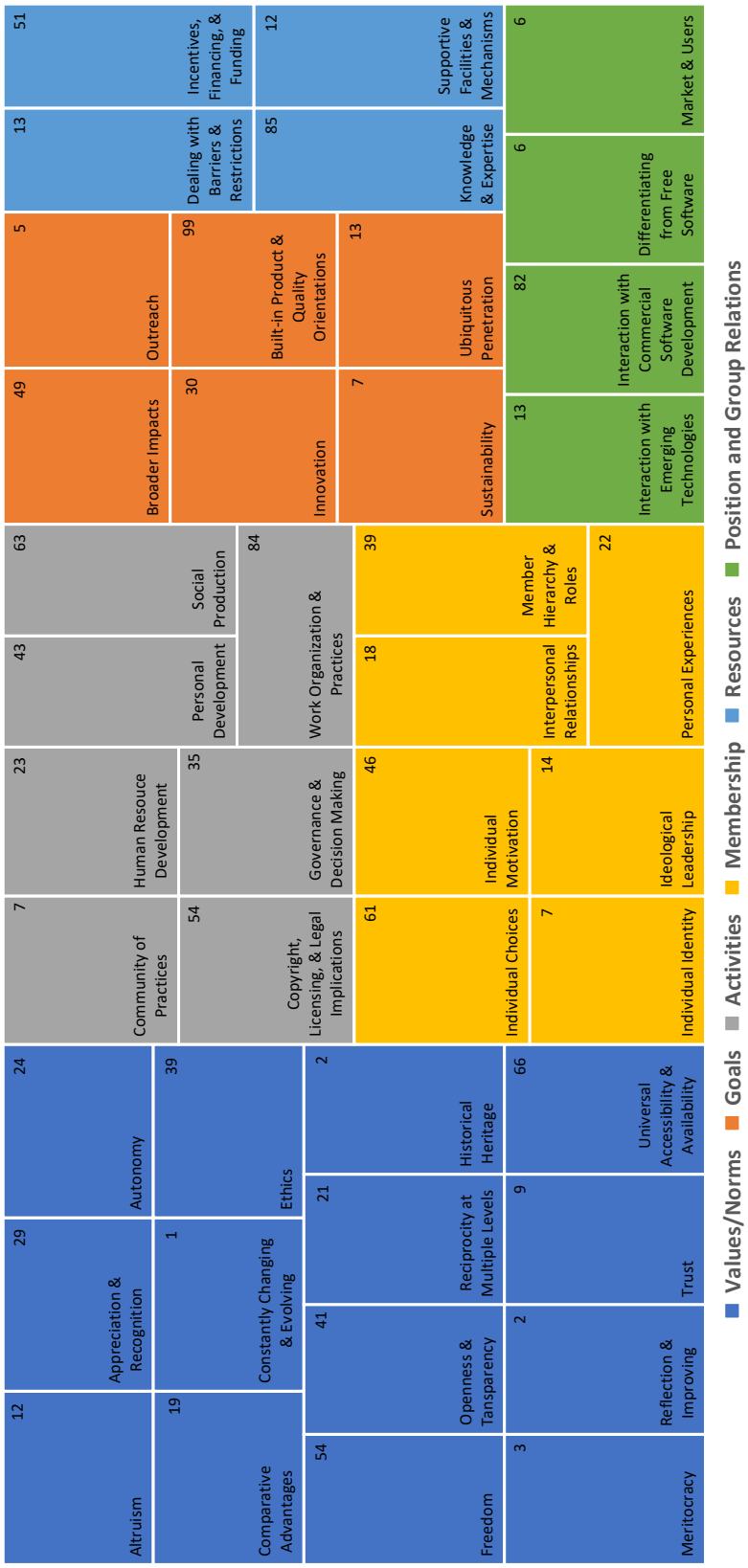

Figure 3.3: The overview of the empirical theory of OSS ideology

Altruism

Baytiyeh & Pfaffman once wrote "*Open source software: A community of altruists*" [6], which well captured such a Utopian value in open source community. Open source development is often driven by *Altruism*. Contributors are voluntarily spending their own time and effort in open source projects, to help others or make something good for others. Our informants shared similar thoughts:

> "*For me, it's (open source) really about making life better for others, maybe not their life specifically, but just helping them solve problems.*" (**P13**)

> "*Most of my projects, I started by myself. Most of the tools that I was writing for myself and I started to publish [as open source projects] to help other people.*" (**P17**)

Appreciation & Recognition

Every contribution in open source development should be recognized by the community. People should "*acknowledge and respect*" (**Deborah Bryant**) altruists' contribution, both technical and non-technical contribution, and give them proper credit. Moreover, in open source projects, there are various ways to express recognition and appreciation to those contributors, as **Hong Phuc Dang** explained "*recognition is not only about the financial reward, but there is also many things that you can do.*" Sometimes it does not need to be very formal, a simple "thank you" could be enough, while other ways include mentioning in comments, listing in release notes, giving a reference letter or certificate, and so on:

> "*Mentioned the name on the issue of the pull requests.*"

"*For young developers and students, it is a certificate and a reference letter that had them for their job seeking or higher education.*"(**Hong Phuc Dang**)

Autonomy

People in open source development tend to be self-governed and self-determined in making choices and decisions, as **Martin Michlmayr** mentioned:

"*There isn't really any one in open source who tells you, oh, you should be doing this, you should be doing that. It's very like people very self-determined.*" (**Martin Michlmayr**)

In open source development, individuals have the autonomy to decide when to contribute, or what to contribute. There is neither a "contributor agreement," nor a "bonding employment," to enforce the contribution in open source projects:

"*There is not much of a daily routine... I pretty much done nothing now, I have free time and for some reason I enjoy spending that free time on hacking an open source project. It's very variable in that sense, there is no fixed commitments.*" (**P12**)

"*Actually, sometimes I do it [open source development] in my working hours, sometimes I do it in holiday, Sunday, or weekends.*" (**P18**)

Comparative Advantages

Open source community shares a strong belief that open source software and its development model have various advantages compared to other types or models. Our informants

mentioned several advantages including faster development pace, better quality, enhanced security, widespread, and so on:

> "*You're getting a much broader access to experts, and so the pace of development increases, the quality of the software increases, and there are benefits that are only possible through open source development.*" (**Patrick Masson**)

> "*I would put security because you can check the sources. You can see what the program is doing. You can compile by yourself and you can have more trust that the software you are using is doing only what you are expecting and not something else.*" (**P22**)

> "*I believe Libre Office is better and could be way better than Microsoft Office.*" (**P16**)

> "*I don't see any other model for software which is spreading that fast.*" (**Amanda Brock**)

Constantly Changing & Evolving

Open source community "*changes over time*" (**P1**). Due to the norms of altruism and autonomy in open source development, newcomers might join open source community, while existing contributors might leave. Thus, the open source community is constantly changing with such dynamics of membership, and gradually evolves over time.

Ethics

> "*The ethical compass become necessary in defining how we deal with new technologies.*" (**Simon Phipps**)

Ethics play a critical role in open source movement. First, people believe that open source movement should incorporate ethics, but in more programmatic ways:

> "*Open source software like a way to use software in a more ethical way.*" (**P17**)

> "*Open source is the pragmatic approach to ethics.*" (**Simon Phipps**)

The reason to highlight programmatic ways is to avoid overemphasizing, which was a lesson drawn from the free software movement:

> "*The open source needs to be decided not to focus on the ethical dimensions of software freedom because they felt those were already being over addressed by the free software movement.*" (**Simon Phipps**)

Our informants further shared their thoughts on the importance of being ethical in open source movement:

> "*Being ethical is what makes [open source software] not that risky for everybody who actually uses these given software... Also a few months ago, there was a scandal, Linus and Greg got really angry about that. When a university, I don't even remember which one was. I think they put some malicious pull-requests and leave there for months. And at times, people started to realize that and they got very angry about that. That is something that's really unethical. Although you've got passed through, I mean, under the light, that was really risky for the whole project, for the whole software.*" (**P9**)

Then, there are particular ethical values, i.e., diversity and inclusion, that the open source community should embrace. Our informants believe they are essential to the whole community:

"*Recognize that each individual person is a human with unique experiences and knowledge and background*" (**P1**)

"*It is crucial we continue to build spaces where all people are welcome, regardless of race or gender or sexually.*" (**Aeva Black**)

Freedom

Influenced by the free software movement, open source movement also values software freedom but in a more practical view. People have full freedom to participate in open source development, as some of our informants shared their experiences:

"*Open source means you can do whatever you want, and that includes taking the software proprietary.*" (**Martin Michlmayr**)

"*I usually get like a couple of emails per day regarding the project. I reply to some, I don't reply to others, I do whatever I want. There's complete freedom.*" (**P12**)

"*If you're just like helping out in your spare time, or if you're just not doing it professionally, then you can do it, you don't need to to professional. It's likely you can do what you want.*" (**P21**)

Historical Heritage

Open source movement should respect the historical heritages of its earlier predecessors, such as the Linux Foundation, and the free software movement. Some of our informants mentioned them when they learned the concept of open source during their careers:

"*That [open source] model originated from Linux, because the way Linux took off is, a bunch of companies kept on looking to hire Linux, and they decided to put a foundation in a place where it was neutral. It's everyone sort of chipped in a bit.*" (**P5**)

"*You always heard things like Linux for open source...*" (**P8**)

Meritocracy

In open source community, power should be assigned according to merit only, which is determined by the value of the contribution in open source projects:

"*Merit is really what you provide to the project, the value of our contribution weighed against how much time you have to put towards it.*" (**Jim Jagielski**)

The reason that open source projects should value meritocracy is because it enables natural collaboration:

"*Because inside of meritocracy, you're gonna be having natural collaboration.*" (**Jim Jagielski**)

Openness & Transparency

The values of openness and transparency are endogenous to open source movement, and should be honored in every aspect of OSS. Generally, in open source projects, the software artifacts, the design and development process, and even the project community should be open and transparent to the public. On the one hand, being open and transparent in open source projects means the public can check or track the artifacts or activities:

"*Openness, obviously like the ability of being able to see the internals [software].*" (**P1**)

"*Open source development would be like publicly visible code bases or teams that work in the public eye.*" (**P10**)

"*Everything appending in development should be transparent and accessible to everyone.*" (**Thierry Carrez**)

On the other hand, openness and transparency indicate that the public can easily participate in open source projects:

"*I would say, that [open source projects] should be open to any kind of member that is interested in learning.*" (**P3**)

"*You have a level of playing field for different voices and opinions to be heard... making sure that all voices get heard and we get the diversity of opinions and thoughts to make good decisions.*" (**P5**)

"*They [open source projects] do allow contribution from external sources.*" (**P8**)

Reciprocity at Multiple Levels

Reciprocal expectations are prevalent, and they bring mutual benefit for both individuals and projects in open source community. At the individual level, OSS practitioners are willing to offer help to others in open source projects:

"*How other people help me, I want to help other people [in the same way].*" (**P14**)

At the project/community level, OSS practitioners who previously benefited from open source projects would like to give back to the projects in return

> "*I seldom met any developer that doesn't want to give back to open source some ways. The good news is everyone that I've ever met has been like, you know, has at least some interest in working on or giving back to open source.*" (**P8**)

> "*Our product is proprietary, but it's solidly built on open source projects, and we just tried to give back to those open source projects in return, basically.*" (**P13**)

Reflection & Improving

Open source projects should be able to frequently self-reflect, and then continuously improve their practices in the development process and community management, as mentioned by our informants:

> "*There is potential for self-correcting [in open source projects].*" (**P5**)

> "*Taking a deep look at what we need to change to modify behavior.*" (**Tracy Hinds**)

Trust

Open source practitioners should maintain certain levels of mutual trust towards each other to ensure cooperation. In open source community, gaining trust tends to be easy, i.e., "*doing what you say you are going to do.*" (**P5**) Once the trust is earned, it can positively influence the contribution in open source development, for example:

"*When someone has built up trust and has full trust... It is found efficient and has been working really well.*" (**P5**)

"*They know they can trust, they know that if you do something, it's gonna be high-quality.*" (**Martin Michlmayr**)

Universal Accessibility & Availability

First, OSS practitioners share a belief that "*source code access was a utilitarian good,*" (**Luis Villa**) thus, the source code of open source projects is always publicly available, and anyone can access it without asking for permissions:

"*Open source means you can see the code.*" (**P7**)

"*Anybody can go and look at the code.*" (**P10**)

Moreover, such accessibility and availability of source code empower OSS practitioners "*the ability to use, modify, and redistribute the software,*" (**Patrick Masson**) and no permissions are required. Our informants shared some experiences:

"*[In open source] Take any code, take any software, and create my own fork of it and make my own modifications, and publish it.*" (**P2**)

"*Be able to add changes, make changes, and easily move code around.*" (**P11**)

"*[Open source] software that anyone can modify and share with anyone with no restrictions on how you can change the code or what you do with the code or with whom you share it.*" (**Karl Fogel**)

"*You are given something [source code], if you want to modify it in whatever way you want, for whatever conditions you feel the need to change it, that's perfectly acceptable. We are not going to restrict you from doing that.*" (**Jim Jagielski**)

3.2.2 Membership

Membership defines the people involved in OSS, including where they came from, why and how they joined, etc. In general, "*people with the same interests*" (**P18**) were welcome to join OSS community of their free will. Seven themes were identified in our data analysis: (1) *Individual Choices*, (2) *Individual Identity*, (3) *Individual Motivation*, (4) *Ideological Leadership*, (5) *Interpersonal Relationships*, (6) *Member Hierarchy & Roles*, and (7) *Personal Experiences*.

Individual Choices

Individual Choices indicated that people's memberships and actions in open source development are of their own choices. There is no time requirement for individuals who participate in open source movement, as most of them choose to spend their spare time in contributing to open source projects:

"*You need to have a balance between your work time and time spent on open source projects. There are no rules that you can't contribute to open source during working time.*" (**P7**)

"*It [open source development] is not my day-to-day job, I'm not contributing every single day.*" (**P11**)

Moreover, some informants also chose to view contributing to open source projects as a hobby in their spare time:

> "*It [open source development] takes time, not just the time that I'm behind the computer, but sometimes I'm just thinking about, I have this problem, how would I fix it? What I'm excited about the problem, it's really like solving a puzzle or totally a hobby at that point. And my wife isn't always happy with me being fully there... But it's the same as if I were really into aquariums or fishing.*" (**P13**)

Individual Identity

Individual Identity refers to people's representation of self in constructing themselves as OSS members. It organizes the beliefs regarding what a member essentially is, thus forming the foundation for the social cognition of one's membership. OSS members' identity is built on the shared understanding of their communities. However, the shared understanding was not necessarily dominating, i.e., people constructed themselves as being a member of several groups rather than a single one [109], and so did our informants. Many contributors developed strategies to manage multiple identities:

> "*I do not mix with the working one [identity] and the private one, and I do use separate computers..., also emails and so on.*" (**P9**)

However, identity struggles might be caused by one's multiple identities [3]. We observed such struggles among OSS contributors, particularly those sent to OSS projects by their employers. Meanwhile, other contributors were also skeptical of these company employees:

> "*He is like an employee of [company], is working in the project because [company] wants to drive these projects*" (**P16**)

Such employees participating in open source development blur the boundary, and make themselves "*just as much capital as they are labor*" (**Karl Fogel**). Therefore, interpersonal conflicts and tensions might arise between these "employed" contributors and other "voluntary" contributors.

Individual Motivation

Individual Motivation reflects what motivated OSS practitioners to contribute to open source projects, including intrinsic and extrinsic ones [11, 112]. Motivation to contribute to open source projects has been extensively studied in the literature. For example, in a recent study, Gerosa et al. identified a set of motivations by surveying OSS contributors, such as fun, reputation, learning, own-use, career, and pay [46]. The motivation identified in our analysis was generally consistent with the existing literature:

> "*I started my first software in my computers and started to play around with open source, so I pretty much transformed my hobby into work with open source.*" (**P9**)

> "*Like an inner need, if I feel the itch, I will scratch it. That's basically it [motivation].*" (**P13**)

> "*But I think that the main reason that people contribute to open source software is portfolio, fame, hope of getting a better job by your contributions, like someone sees your work here and gives you a job.*" (**P17**)

> "*I have two sides: one side is the egoist, individual side of, like, I want to show the world that I know how to code, contribute to these projects; and the other side is like, okay, I want people to help the people that don't know how to write code, people use the tools that I'm contributing to.*" (**P17**)

"*I like programming, want others to see my code review, give suggestions, help me out, I can grow professionally.*" (**P19**)

Ideological Leadership

Ideological Leadership is about the connection between a shared set of OSS ideologies and leadership in communities and the movement, which had been well characterized in [81]. Our informants first shared how such leadership rose from open source projects, and how the leadership was organized within the projects:

"*Allow for leadership and activities to come from anywhere, don't pick your favorites and create channels that really just only allow certain people to participate.*" (**Patrick Masson**)

"*Having that [leadership] as an elected position periodically.*" (**P5**)

"*Depending on the given project, the size of the contributors, even if it's horizontal, it doesn't make sense to have just one leader. And also for huge projects, it makes sense for me to have one leader or at least one leader and a few subleaders.*" (**P9**)

Then our informants further described the responsibilities of these leadership roles in open source projects, including single consistent vision, agenda adjuster, and final decider:

"*Single consistent vision, like having one or two people who really understand what it is they're trying to do, and building software that makes the vision happen.*" (**P1**)

"*The technical steering committee chair acts as a sort of final decider, the adjuster of agendas.*" (**P5**)

Interpersonal Relationships

Interpersonal Relationships describe the relationships between individuals in open source development. Most tend to be in a friendly and professional manner:

"*Be polite, be friendly, I think. Some contributors are new, be friendly to these new people. Everyone starts from a beginner.*" (**P7**)

"*I tried to be as professional as possible, I think being professional really kind of helps.*" (**P11**)

However, conflicts also arise in the relationships sometimes, as **Tracy Hinds** and other informants pointed out:

"*Modern theories consider conflict a natural and inevitable outcome of human interaction.*" (**Tracy Hinds**)

"*People did a lot of conflicts in a [open source] project I worked on before.*" (**P10**)

Member Hierarchy & Roles

Member Hierarchy & Roles identifies the hierarchical social structure of contributors in open source projects, and their privileges and responsibilities defined by their roles in the membership pyramid. First, the hierarchical social structure of contributors is usually described as a pyramid structure:

> "*I have a theory of this that I call the pyramid of participation, which is kind of like food pyramid. You just have like a bottom layer that has the least trust but the most people, and then they gradually go up to the project leadership at the top... like there should be some people at various different levels, and it needs to be like a fat pyramid.*" (**P4**)

Then, within this membership pyramid along the path from the bottom to the top, there exist several roles: "*bug hunter*," "*maintainer*," "*developer*," "*project manager*," "*fundraiser*," etc. (**P1**, **P7**, **P17**, **P20**) These different roles also associate with different *Role-defined Privileges & Responsibilities*. Take maintainers as an example, maintainers in open source projects usually have control privileges to have "*full access to everything*," such as "*approve the change*." (**P19**) In addition, maintainers also need to have more commitment and take more responsibilities to keep project "*well-maintained*." (**P17**)

While such as *Membership Hierarchy & Roles* system made open source project community more organized and "*guarantees some continuity*," (**P4**) it also raises ideological concerns. First, the pyramid social structure effectively created elite and non-elite social classes, which might threaten the ethical value of equity [50]. Second, maintainers are increasingly putting extra burdens beyond their willingness [103], which constitutes violations to *Autonomy*. How to balance these circumstances should be addressed in future research and practices.

Personal Experiences

Personal Experiences capture the personal feelings in open source development. Most of our informants shared positive feelings and experiences, such as fun and nice, for example:

> "*That is really interesting for my own curiosity.*" (**P6**)

"*It's very fun to collaborate with other folks [in open source development].*" (**P12**)

"*It's actually a pretty nice feeling to find the bug, to fix the bug, and to have a little bit of my code in there.*" (**P12**)

However, some of our informants also mentioned their unpleasant experiences in open source development:

"*Open source development broadly has a massive problem with burnout because so much of the work is being carried by volunteers.*" (**P1**)

"*Maintainer burnout is a real thing where people feel committed to their projects.*" (**Patrick Masson**)

Open source practitioners need to carefully deal with the unpleasant feelings, as it could negatively influence the open source development:

"*For example, some people go into your repository and start to say, you should do this or you shouldn't do this, and it's very annoying for some people. They may decide to stop all activities.*" (**P18**)

3.2.3 Goals

Goals describe what members want to achieve or realize in the OSS movement. Six themes emerged in our data analysis, i.e., (1) *Broader Impacts*, (2) *Built-in Product & Quality Orientations*, (3) *Outreach*, (4) *Sustainability*, (5) *Innovation*, and (6) *Ubiquitous Penetration*.

Broader Impacts

Broader Impacts capture the goals that the OSS movement would like to achieve beyond software productions. First, our informants pointed out open source development's influence on the software industry:

> "*I think open source as a development methodology has changed software engineering as an industry. Practices around version control, code review, documentation, and continuous integration have advanced significantly.*" (**P1**)

> "*We started this impossible [open source] movement, if you can imagine, to completely recreate the world of computing.*" (**Danese Cooper**)

Then, beyond the software industry, open source practitioners believe open source software can benefit the public and society, as shared by our informants:

> "*The last ten years I've got into open source, it gave me the opportunities to figure out how we can make things better for other people, and just make things better... You've made other people's lives easier.*" (**P5**)

> "*Especially some of the [open source] projects I worked on, for example, with a humanitarian toolbox. They build an app that helps children in third world countries, they help find missing children in third world countries after disasters, kind of like a real charity kind of project.*" (**P10**)

Furthermore, open source practitioners share an ambitious vision of the open source movement, i.e., "bringing humanity forward":

"*I do think that every open source software should have the same goal, which is bringing humanity forward.*" (**P9**)

"*They [open source projects] are a proof point of how we as one planet and one humanity can work together to empower everyone.*" (**Aeva Black**)

Built-in Product & Quality Orientations

Built-in Product & Quality Orientations is the fundamental goal for all the other goals of the open source movement. Without high-quality products, none of the other goals could be realized. This goal was built in the open source movement and considered to be a key component in Steward & Gosain's OSS ideology model, i.e., "*better quality – open source development methods produce better code than closed source*," [102] since it directly determined many activities in development. Our findings are consistent with the literature that the open source movement aims to provide better quality software products:

"*Obviously, the quality [of open source software] is going to be much much better than the proprietary in counterparts.*" (**P2**)

"*Open source methods really are able to produce higher-quality software faster.*" (**Danese Cooper**)

Outreach

Outreach describes the open source movement's goal of promoting open source software to reach a broader spectrum of users and communities, for example:

> "*I think it [open source project] is getting really stable, I should promote more, and then maybe I find all the people that are interested in this particular solution.*" (**P13**)

> "*I tried to make it [open source project] heat, and known by the community, which they can use it, they can test it.*" (**P20**)

Open source practitioners utilize various channels and approaches to promote open source projects, such as "*social media*" (**P11**). For example, X/Twitter has been proven as an efficient platform to promote open source projects [33, 34], and many open source practitioners actively share and promote their projects there.

Sustainability

Sustainability refers to open source communities' goal of achieving sustainable community dynamics and growth. This goal and its antecedents have been well documented in literature such as Chengalur-Smith et al. and Gamalielsson et al., regarding its importance to open source projects [19, 44]. Similarly, our informants also recognized that open source projects need to be "*long-term sustainable*" (**P1**), and open source communities should "*keep project sustain and grow*" (**Hong Phuc Dang**).

Innovation

Open source development aims to drive *Innovation*:

> "*The real value of open source is that it lets you innovate without having to ask anyone's permission.*" (**Simon Phipps**)

However, innovation in open source development was often neglected in the software engineering literature. Only a few scholars had worked on it [38, 88]. Even those related studies almost always focused on innovations resulting from micro-level adoption of OSS products [41, 54], e.g., using OSS solutions in products. The limited scope of the extant work restricted our understanding of innovation as a goal of open source movement, which repeatedly emerged in our study.

First, the open source movement aims at providing *Discruptive Environment for Innovation*, which "*removes obstacles and boundaries*" (**Thierry Carrez, Simon Phipps**) of different organizations and institutions. Levine & Prietula's four open collaboration principles– goods of economic value, open access to contribute and consume, interaction and exchanges, and purposeful yet loosely coordinated work– were all present in open source development [71].

Then, our informants suggested a much more aggressive belief that the open source movement creates "*Free Market of Both Ideas and Software*" (**Mike Milinkovich**), beyond a playground for open collaboration provided by the disruptive environment in open source development. In open source development, practitioners could break technical rules, explore and experiment with rich technical connotations, and exercise minimal degrees of creativity for selective innovations, with very low or even no cost, as our informants shared:

> "*What you will want is a more broad collection of ideas and directions being brought into the [open source] projects.*" (**P1**)

> "*That you've made it possible for something to exist that didn't exist before.*" (**P5**)

> "*The good thing is with open source, you don't have to give the guys money, they can go ahead and do it because the cost of innovation is so low.*" (**Joichi Ito**)

All indicate opportunities in future software engineering research on open source innovations. Moreover, the OSS innovation's complex and multi-faceted empirical realities make its theoretical significance lie in its interdisciplinary nature. Regarding our current knowledge, the intellectual exchanges between software engineering and innovation management were limited. Software engineering researchers interested in OSS innovations might consider integrating rich open innovation theories developed in their future research to enable cross-fertilization between disciplines [27].

Ubiquitous Penetration

Ubiquitous Penetration reflects the open source movement's goal of penetrating into every aspect of modern society. First, open source software is widely adopted as *Foundation/Infrastructure of Technology*, such as cloud computing infrastructure, and Internet infrastructure, as our informants pointed out:

> "*Open source is foundational to all of modern technology.*" (**Luis Villa**)

> "*Open source has become the heart of most new software.*" (**Simon Phipps**)

> "*[Open source] providing infrastructure for others to build and deploy applications on.*" (**Thierry Carrez**)

Second, our informants noticed *Open Source in Everyday Life*, open source movement nowadays is gradually becoming "*a way of life*" (**P9**), for example:

> "*It [open source development] is not just now recognized, it is becoming part of the day-to-day operations and expectations.*" (**Patrick Masson**)

> "*I think what the major trend has been in the software world over the last 15-20 years is that open source is 'eating' software, if software is 'eating' the world, open source is becoming the mechanism by which that is happening.*" (**Mike Milinkovich**)

3.2.4 Activities

The category of *Activities* deals with questions such as "What do OSS contributors do?" and "What are expected activities in open source projects?" This category is the most important one among the six categories. According to van Dijk ([109], pp. 70-71), an ideology system could be identified by one particular category. For open source ideology, its distinctions mainly lie in the activities, particularly the activities related to copyright & licensing that define it (*The Open Source Definition*, ver. 1.9, available at `https://opensource.org/osd`). Thus, OSS ideology was typically an **activity ideology** representing that OSS contributors loosely gather to form communities for producing software under specific OSS licenses. We identified seven themes in this category, including (1) *Community of Practices*, (2) *Copyright, Licensing, & Legal Implications*, (3) *Governance & Decision Making*, (4) *Human Resource Development*, (5) *Personal Development*, (6) *Social Production*, and (7) *Work Organization & Practices*.

Community of Practices

Community of Practices refers to open source community members' collective learning activities as a group of people who "share a concern or a passion for something they do and learn how to do it better as they interact regularly" [68]. Our informants shared similar perceptions within the open source community:

"*Whenever you start working on a [open source] repository, for whatever reason, you certainly start to come across the same people who are most likely working there, and other people are simply there for helping. There is certainly a sense of community in directs.*" (**P6**)

"*You should try to blend in [open source community], you try to learn the culture and do things in the way that the community expects.*" (**Martin Michlmayr**)

Copyright, Licensing, & Legal Implications

Copyright, Licensing, & Legal Implications refer to the legal activities and practices in open source development, particularly about dealing with copyright and patent, using open source license, and so on. These legal activities provide explicit guarantees on the aforementioned norms/values.

There are no trade secrets in open source software since the source code must be open, no matter under which open source license. Thus, there should be some ways to protect intellectual property. Open source practitioners have certain *Rationales of Dealing with Legal Issues*. First, licensing is required by the definition of open source software, as OSI president **Simon Phipps** pointed out:

"*It [software] doesn't become open source until you put an open source license on it.*" (**Simon Phipps**)

Second, licensing is a way of explicitly offering permissions with software and its source code:

"*Legally, if there is no license, it is automatically all rights reserved. That means the work owner gives no permission, no rights to anyone else to do anything at all.*" (**VM Brasseur**)

Third, licensing has practical benefits by simplifying and hiding many legal details, and open source licenses are designed to be reused in open source community:

> "*While they [open source licenses] are written by lawyers, they are intended to be reused, they are also intended to be reused by people who aren't lawyers, whereas most other licenses in the world, they are not... This makes everything easier for all of us.*" (**VM Brassuer**)

Fourth, licenses could be "*the constitution of a community*" (**Simmon Phipps**), because the licensing reflects the common expectation in open source projects, and contributors' participation decision implied their agreements on licensing:

> "*Licenses tend to be constitutions of communities. They serve as a foundation for how the project that picks the license wants to operate, how it wants to engage with its contributors, what its expectations are for the community of users and contributors.*" (**Patrick Masson**)

Indeed, open source software has complicated relations with some troublesome laws, mainly on *Copyright vs. License*. Under the current copyright law, copyright is often viewed as "*a control point*" (**Simon Phipps**), and "*a form of protection*" (**VM Brasseur**) for the copyright owners. Licenses offer solutions to copyright owners to retain the rights while giving someone else the right to exercise some of them:

> "*By selecting the different open source licenses, you can limit how much of that control you give up, or you can open up the amount of control you give up.*" (**Jim Jagielski**)

The legal definitions of these two terms, copyright and license, imply certain ambiguities. Meanwhile, practitioners have different ideas about who owns the copyright, what could be

given to other parties, and more importantly, whether they would (partially) lose control over the work. For example, being asked to sign a Contributor License Agreement (CLA) to transfer copyright to the project/organization often triggers some boycotts. Therefore, it is not surprising that many practitioners agree that there is still "no flawless license" even when a large pool of legal professionals is serving the OSI board:

> "*You can find fault with any widely used open source license, whether it's the Apache license or the MIT license or any of the copyleft licenses... You can't easily revise the license, if a flaw or an interpretive ambiguity is later highlighted, that's just sort of a general challenge of open source licenses.*" (**Richard Fontana**)

Licensing and other related legal issues not only influence the business model and operations in open source movement, but also have *Consequences on Development*. Licensing is powerful to enable the development and distribution models, which makes open source software "*unprecedentedly spreading across all of the categories of software*" (**Amanda Brock**). Our participants also acknowledged the significant impacts of licensing in open source development, for example:

> "*There's different dynamic behind some of these [open source] communities, like depends on the licensing to some extend, and the licensing shapes the community dynamic in my mind.*" (**P5**)

> "*I think the licenses govern the [open source] communities.*" (**P5**)

> "*Their [open source projects'] licenses are slightly different, where I got one that has like a database for mechanical keyboard switches and the source code for that*

is all public, it's MPL license as opposed to MIT, so that no one can trade market essentially." (**P8**)

Note that such consequences are not necessarily to be positive. Former chief policy advisor of United States Patent and Trademark Office (USPTO), Arti Rai once said "*law can sometimes pose, rather than resolve, problem,*" when discussing open source and legal issues [89]. For example, some company-sponsored projects may use restrictive licenses to avoid competition, which de facto betrayed the OSS ideology.

Legal professionals mostly did the design of the licensing systems. Ordinary developers often had minimal legal knowledge, and as **Pamela Chestek** said, "*I don't think that most human beings are interested at all in legal instruments;*" so they might be confused about why they need to bother with legal issues. Therefore, when encountering these issues, *Reluctance & Resistance in Development* is not uncommon. Developers naturally preferred more permissive licenses:

"*I started off with stuff that was under GPL licensing, and then basically emerged to doing more work under the more permissive licenses.*" (**P5**)

"*There was a trend of developers putting tools and libraries under permissive licenses in this time period.*" (**Richard Fontana**)

"*In these sorts of scenarios, the more permissive licenses are the ones that usually work out better... You really want to reduce the amount of conflict that goes into creating a software project.*" (**Jim Jagielski**)

In addition, open source practitioners also proposed alternative legal arrangements to solve the limitations of licensing. For example, trademarks received some attention recently, as pointed out by **Pam Chestek**:

"*A trademark is owned by the entity that controls the mark and controls the quality of the goods and services with it... If I talk about the trademark, then all of a sudden people grasp it because trademarks are so part of our human experience that everyone understands them.*" (**Pam Chestek**)

Governance & Decision Making

Governance & Decision Making refers to several governance structures in open source projects, i.e., benevolent dictator, walled garden, and true meritocracy, as well as the different decision-making processes that correspond to them.

The first governance structure is called benevolent dictator, sometimes also called benevolent dictator for life (BDFL). In some open source projects, a particular member or a group of members are appointed as the leaders, who are usually the project owners. As the benevolent dictator, they retain the final say regarding the project development:

"*In the benevolent dictator for life, that power exchange is actually bequeath by the community. The open source community says, we trust this individual or this set of individuals to make these controlling decisions for us.*" (**Jim Jagielski**)

The second governance structure is the walled garden. Although the project is available to the public as open source project, the involvement in the development is restricted to certain members only, as **Jim Jagielski** explained:

"*There is a governance model for an open source project called the walled garden. The idea behind the walled garden is that it's available under an open source license, but the community, the external community involvement, as far as figuring out the direction of the project or the list of features that will be included*

in a project is significantly restricted. These are the kinds of projects which are usually controlled and managed by a single company. So unless you are an employee of the company, you probably don't even have commit bits to this open source project... The goal is to make sure that the company or the single entity has overriding control over how that project grows, where it goes, what features are added and things of that nature."(**Jim Jagielski**)

The third governance structure is called true meritocracy, where "*all the members of the [open source] community could contribute and not only contribute but lead if based on meritocracy.*" (**Italo Vignoli**) Every member in such open source projects could gain control of the project based on their merit, which is different from the walled garden and benevolent dictator governance structures that only certain members have control privileges. The development and decision-making processes within the true meritocracy governance structure are driven by "*consensus-based collaboration*" (**Jim Jagielski**)

Human Resource Development

Human Resource Development reflects the developing a workforce for open source development, including *Attracting, Recruiting, and Retaining Contributors*, and facilitating *Growth of Contributors*. First, software engineering literature has revealed the importance of *Attracting, Recruiting, and Retaining Contributors* for open source projects [24], and the findings from our analysis remained consistent:

"*We have been attracting new developers on a regular basis [in open source projects].*" (**Italo Vignoli**)

"*You definitely don't want to turn them away, but try to work with them [in open source projects].*" (**P11**)

"*I think the most important part of open source development is to remain, people should remain [in open source projects].*" (**P19**)

Furthermore, open source community should not overlook the *Growth of Contributors* in open source projects, for example, making the project easily accessible to newcomers, encouraging them to actively contribute, allowing them to choose tasks that suit them, etc. (**P8, P11, Nnenna Nwakanma**)

Personal Development

Personal Development is about the activities related to an individual's professional skills, and seeking career opportunities. Some informants intended to build/enrich their portfolio to demonstrate experiences and skills regarding software development by participating in open source development, which could benefit their career, for example:

"*Basically what got me into my PhD was sending my advisor an email, 'hi, I'm interested in this and I worked on this open source project,' and the next day I had an appointment with them. Of course you can do that without open source, but it's a very good way if you have those projects as show case.*" (**P12**)

"*Actually, when I joined the company, one of the deciding factors of my employment is the participating in open source, like the employer could easily verified that I was capable of writing good code.*" (**P19**)

"*When I applied to jobs, you can show them, here I have worked on this [open source] software, and you can see the code, the pull requests, some of comments, some reviews and all that stuff.*" (**P21**)

Then, open source community is also full of professionals nowadays. Within open source development, practitioners also aim to improve their professional skills:

> "*Treating our contributors as tech professionals as they are, not as cultural hobbyists. This is 2022, people aren't using doing open source for a hobby anymore. And if professionals working for you, they should be treated as one.*" (**Deborah Bryant**)

> "*Actually, I like programming, want others to see my code review, give suggestions, help me our, I can grow professionally.*" (**P19**)

Social Production

Social Production refers to the collaborative nature of its members' activities that features the collective efforts of multiple entities. People around the world can collaborate together within open source development, which makes open source development "*a global effort, multi-timezone efforts*" (**P4**), and further foster a strong sense of community:

> "*We have now created a global community of software developers who have for the first time, the ability to truly cooperate on a global basis.*" (**Michael Tiemann**)

> "*Open source is ultimately about community, it's ultimately about people, about enabling those people.*" (**Matt Asay**)

In addition, such mass collaboration in open source development is also cross-organization and with no boundaries:

> "*Pulling together communities of like-minded developers working for different companies on common infrastructural technology.*" (**Brian Behlendorf**)

"*We will need to collaborate without boundaries all of our communities, and not limit ourselves to the project that we jointly support.*" (**Thierry Careez**)

Work Organization & Practices

Work Organization & Practices refers to practices regarding how work is organized in open source development, such as communication and coordination activities, and routines of development activities. It includes *Decentralized Mass Collaboration*, *Communication & Coordination*, and *Regulations, Routines, & Rituals (3Rs)*. Software Engineering and Computer-Supported Cooperative Work researchers have spent tremendous efforts in investigating the first two practices since late 1990s. However, the 3Rs received much less attention than it was deemed in the related literature.

These 3Rs are derivatives of the Values/Norms in the OSS ideology. As concrete social control mediums, they ensure the realization of Values/Norms in daily activities. First, regulations are explicit rules defining uncompromising boundaries of activities [2]. It helps produce appropriate individuals and excludes those who did not respect community values/norms. Regulations could be applied to both technical and social activities. For example, **Tracy Hinds** mentioned "*how peer conflict regulation work to dismiss an individual do not respect community value*" in the Node.js Foundation. Second, routines are a repetitive patterns of interdependent activities that can be learned and exercised by members. We observed that routines widely existed in the practice of open source development. Moreover, our informants also pointed out that routines might be evolving along with the development of communities and projects. In open source projects, the performances of routines are continuously evaluated, which opens opportunities for changes. Thus, an informant (**P12**) claimed that "*no routine work in open source.*" This coincided with findings from organizational theorists such as Martha Feldman and Brian Pentland [35, 36]. Given that the dynamics of routines could be reproduced from software repositories, there are opportunities for further theory

development. Third, rituals can be characterized as standardized behaviors undertaken in conditions demanding explicit expectations. For example, one of our informants shared some rituals of bug fixing: "*one type of contribution is they open issues seen in repository, they report bugs that would be considered as contribution, too, and they participating in testing software.*" (**P19**)

Another vital function of the 3Rs is their channeling roles among organizational cognition of values/norms, activities, and community/project development. Studying such roles would probably lead to establishing a complete network from values/norms to products, by embedding these constructs' complex relationships. Thus, we could answer critical questions such as how value/norms were reflected in activities and project outcomes or how to leverage such channels to ensure the values/norms were honored, and hereby offer insights for making open source movement more value-sensitive [85].

3.2.5 Resources

Resources are essential for a community to survive and develop. Conservation of Resources (COR) theory argues that an individual or a group shall preserve and protect those resources that they value [56]. Resources are not restricted to tangible resources but also include various intangible ones. We identified four themes: (1) *Dealing with Barriers & Restrictions*, (2) *Incentives, Financing & Funding*, (3) *Knowledge & Expertise*, and (4) *Supportive Facilities & Mechanisms*; only the second theme, *Incentives, Financing & Funding* falls into the class of tangible resources, while other themes are intangible resources.

Dealing with Barriers & Restrictions

Dealing with Barriers & restrictions mentions the resources that individuals and a community used to deal with barriers & restrictions they faced. A typical individual-level resource is some personality traits, e.g., resilience, since they could help people deal with burnout and frustrations, particularly in one's onboarding and early career phase, which is identified by both the literature [101] and our informants:

> "*I didn't know how to navigate the big codebase by myself. I tried a lot of times and kept failing.*" (**P15**)

> "*At the same time for people who are new to open source, sometimes it can be a little bit challenging in how you get started.*" (**Martin Michlmayr**)

> "*I do want to focus today on developers and contributors because they are experiencing burnout in a big way.*" (**Deborah Bryant**)

Incentives, Financing & Funding

While the contributions in open source development tend to be voluntary mostly, a healthy cash flow is still quite important for many open source projects, especially the large ones, to maintain community infrastructures, e.g., paying for project communication services [126]. However, *Incentives, Financing & Funding* was often overlooked or purposefully ignored by researchers in contrast to voluntary contributions.

First, *Project Donation & Sponsorship* is the major way to fund an open source project [84], if not the only one, because most open source projects have no revenue stream. The financial support to the project might be viewed as commercial entities' obligation to open

source communities. Open source development helped many companies save substantial costs in their business. Therefore, paying back to open source is not only a philanthropic action but also a way to engage in a mutually beneficial endeavor. Nevertheless, there are no effective mechanisms to force a company to offer donations or other forms of funding. Some of our informants pointed out this issue:

> "*Companies that use open source often get to it because it is free... But they are not obligated to give back even when they are making a lot of money with that.*" (**P1**)

> "*If there wasn't this core infrastructure available as free and open source software... Scale up to a huge number of servers if they are paying a per license cost for every single one of them, that just would be have been possible.*" (**Leslie Hawthorn**)

Therefore, many open source projects lack reliable funding support. Indeed, the donations are not necessarily to be large; our informants mentioned that even "*small donation is a great booster*" (**P12**). Besides, many informants shared that they would like to keep a vigilant attitude to donations from companies, because sometimes "*money backs up decision*" (**P14**). If an open source project overly relies on the donation from a company, it would be inevitable influenced by the donor, which is probably not aligned with its own objectives and interests.

Then, *Financing Individual Developers*, providing financial support to individuals, is considered to be legitimate for certain contributors. For instance, if someone worked full-time for open source and without other means to support their basic life needs, receiving money as their living income would be acceptable and appropriate. However, *Monetization of Contribution as Incentives*, i.e., "providing monetary incentives for contributors" is much more controversial. There are opposite opinions on this issue. The arguments from incentive

supporters include:

> "*I think it would be nice to get paid to make contributions to open source projects... A lot of work isn't fun like writing tests, and you are testing on different devices. I think it'll lead to higher quality work because I can justify if spending more time on it personally.*" (**P15**)

> "*The lack of incentive in a lot of open source projects is the main reason to limit the prosperity [of those projects].*" (**P17**)

In contrast, more practitioners held neutral to negative views of money incentives beyond supporting contributors' basic needs, for example:

> "*When you contribute [to open source project], it means you want that project to be better. But when you contribute because of money, that totally changes the intention of contribution, it's more like a job.*" (**P7**)

> "*I'm okay with it [money incentive], within boundaries, I guess. If I know someone is being paid one million a year for an open source project, good for them. But if I have worked on it for months and months, and I don't get any compensation, that might be hurt a little bit.*" (**P13**)

Even those who are neutral to incentive schemes agree that the money incentive should be within boundaries, unfortunately, drawing such boundaries is non-trivial.

Some influential figures, such as **Eric Raymond** – the author of *The Cathedral & the Bazaar*, emphasized that the current supports for individuals should be reformed because "*no money goes to people maintaining critical infrastructures.*" Those load-bearing people

often work full-time for open source projects, but current open source operations "*fail in funding the people institutionally.*" Thus, he proposed a decentralized patron system to fund these developers. In such a system, not companies but ordinary people who have regular jobs provide small amounts of money gift, e.g., $30 per month, to those load-bearing open source contributors directly. The decentralized patron mechanism, if well executed, might help solve some problems in funding open source and some individual developers in need. There have been several experimental platforms for decentralized funding, for example. But almost all of them are *incentivizing/rewarding work* rather than providing living income to people in need.

To sum up, there are many unsolved and controversial issues in establishing funding systems for open source projects and contributors. Researchers and practitioners should pay extra attention to financial issues and develop innovative solutions to ensure the financial sustainability of projects and load-bearding individuals.

Knowledge & Expertise

Knowledge & Expertise is another important resource in open source development. First, open source projects provide a knowledge sharing platform, practitioners could freely share their knowledge and expertise with others, and various knowledge and expertise, e.g., programming knowledge, development experience, discipline, and practices, could be shared across open source community:

> "*The open source community is partly how we share these practices.*" (**P4**)

> "*Professionally, when I'm working as a technical support, I think there are actually many solutions are supported, contributed by the community. And it's definitely a great thing because one person may not have all the knowledge ob-*

viously. But the community can, in a sense, and we can help each other find different solutions." (**P6**)

"*The main thing I see from open source is the fact that we are sharing code, and we are sharing ways of doing stuff.*" (**P10**)

Then, any individuals involved in open source development could learn and benefit from those shared knowledge and expertise, as our informants mentioned their learning experiences:

"*I have a PhD in computer science, but I never learned how to build anything to do with the web. And [open source project] documentation was the first time that I read something about the web and got 'Oh, that's what you're trying to do,' because they actually explained it. I went through a tutorial process that that got you from knowing nothing to knowing how the project work.*" (**P1**)

"*Discipline is important and people learn discipline by learning it in school, but also they learn discipline from seeing others practice that same discipline [in open source development].*" (**P4**)

In addition, according to Michel Foucault, power is based on knowledge and makes use of it; on the other hand, power reproduces knowledge by shaping it in accordance with its anonymous intentions [43]. The dynamics between *Knowledge & Expertise* and power in open source development should be well worth some future investigations.

Supportive Facilities & Mechanisms

Open source community has implemented various *Supportive Facilities & Mechanisms*, such as tutorials/documentation of the open source projects, mentorship for newcomers, and contribution template/guideline, for example:

> "*I think it's good to have people that will help guide you, and mentor you, and things like that.*" (**P11**)

> "*I think that is really useful for newbies, like in that template, there's also a guide on how to do a bug report, what kind of information people need to provide.*" (**Hong Phuc Dang**)

Note that all of those supportive facilities and mechanisms have been well documented in the related literature [48, 101, 100], and our findings are consistent with the literature.

3.2.6 Positions/Group Relations

Position and Group Relations deals with a series of questions such as "What is our social position?" "Who are like us, and who are different?" "Who are our opponents?" For this category, four themes were identified: (1) *Interaction with Emerging Technologies*, (2) *Interaction with Commercial Software Development*, (3) *Differentiating from Free Software*, and (4) *Market & Users*.

Interaction with Emerging Technologies

Interaction with Emerging Technologies reflects the fundamental positions of open source software in enabling emerging technologies, such as cloud computing, and AI, and forming Internet-wide infrastructures together with these technologies. Take cloud computing as an example, **Matt Asay** explained the complex relationship between open source software and cloud computing. First, open source software made cloud computing possible, or enabled cloud computing:

> "*The cloud [computing] would not be possible without open source, and I think that's true... With open source, you don't really have to think about that. With traditional enterprise software, proprietary software, you did. And it makes such that the cloud would be, if not impossible, and I would argue impossible, at lease extraordinarily difficult... So we know that open source enables the cloud.*" (**Matt Asay**)

Then those cloud computing companies contribute to open source software as a return:

> "*If you're playing your odds of the top ten open source contributors, seven of these top ten are our cloud vendors [cloud companies].*"
>
> "*It's because cloud companies don't really have anything to lose by contributing software, they don't sell software, they sell services... This is the sort of thing that the cloud enables because it takes away the competitive need to keep software proprietary. If you run some sort of calculus as two contributes equals two benefits. Well, it also makes sense that the cloud companies would be big contributors because we're benefiting a great deal from open source.*" (**Matt Asay**)

Interaction with Commercial Software Development

Proprietary software produced in commercial development has a lot to do with open source software. To some degree, they are enemies. However, the relationships are much more complicated, even only from the ideological perspective. Open source practitioners have diverse views, i.e., *Coexisting, Competing, & Replacing*. Some believe most proprietary software already has open source counterparts, so the commercial offerings could be eventually replaced by open source software, as our informants shared an example in their organization:

> "*Then it [open source software] became the basis, it replaced the commercial offerings.*" (**P4**)

A less optimistic view is that these two types of software are competing with each other, and it is still too early to conclude who will dominate the software market in the long run. **Karl Fogel** explained this view with an example:

> "*Suppose you wrote an online service... But as soon as someone wants to challenge you in that marketplace, the easiest way for them to do it is to build something that's open source or take something that is already open source.*" (**Karl Fogel**)

A few others hold the opinion that both could coexist:

> "*Certainly there's room for proprietary software in the world.*" (**Brian Behlendorf**)

> "*Open source and proprietary software, it can coexist.*" (**Tony Wasserman**)

Interestingly, those practitioners' narratives often put open source in an inferior position by downplaying open source software as "*non-core business*" (**P4**) or something similar.

Meanwhile, there is a consensus that the participation of commercial companies is not motivated by ideological reasons, but practical benefits, which is consistent with Wagstrom's findings of vertical integration of commercial and open source software [113]. For a commercial company, they usually consider *Involving Open Source as Strategic Decisions*. Our informants discussed dozens of such practical benefits, including branding and promotion on the market, cost-reduction by reusing and trimming down product portfolios, diversifying assets, experimenting with innovations, and hiring, among many others:

"*Many companies are formalizing their strategies, and creating open source offices and programs.*" (**Deborah Bryant**)

"*Implied value for the branding.*"

"*It's a business that needs to reduce cost, so open source helps reduce cost.*" (**P4**)

In addition, the involvement of commercial entities has profound impacts (*Positive & Negative Impacts Combined*) on the practice of open source development, and may reshape the ideologies subtly. Prior literature often emphasized the positive impacts of a company's engagements [125], which was also confirmed by our informants:

"*You could increase the number of participants by a lot [in open source development].*" (**P4**)

"*Some stuff like writing excellent documentation in multiple languages is just nothing you see in those personal projects, because it's not a lot of fun, and nobody pays you to do it. There is something where say [companies], they just throw money at the problem and then they have those multilingual documentation.*" (**P12**)

However, some recent studies revealed the negative impacts [124, 123], many informants expressed similar worried and concerns that companies would manipulate the open source projects to maximize their benefits or some misbehavior:

"*My company also donates to open source development, and donates a lot more, and I think that they do sort of expect some, at least the owner to have a conversation with them sometimes. I do know that there is one of the projects that*

my company does donate to, the owner explicitly says if you donate this much money, I'll make sure or I'll consult with you if it breaks one of your releases or something like that." (**P8**)

Differentiating from Free Software

Another group that is relevant to the open source movement is the free software from which the latter emerged. The relations between them were much simpler. From the historical perspective, as **Luis Villa** claimed: "*there is a lot of overlap [between free software and open source software].*" For instance, they both share the idea of making source code available, and most open source software is actually free to use. However, they stand for views based on fundamentally different values [99]. The free software movement is an ethical imperative, while open source movement is a pragmatic approach to ethics, as one of our informants explained:

"*The open source movement was a lot more about the removing the extremist political dogma from the side of the free software project, and a free software as an idea and making it more business compatible.*" (**P1**)

Therefore, the free software movement and open source movement are not enemies, but compete in the ideological and philosophical views in people's mindsets. [41]

Market & Users

Market & Users summaries the relationships among open source software, market, and users from multiple perspectives such as economics, management, HCI, etc,. The narratives of our informants are generally consistent with findings in the extant literature [4, 20, 41, 69, 23], such as no liability to users, and marketing and promoting the projects:

"*I think it [open source project] is getting really stable, I should promote more, and then maybe I find people that are interested in this particular solution.*" (**P13**)

"*There is a form which says that we are not responsible for anything at open source.*" (**P20**)

3.3 Implications

3.3.1 Theoretical Implications

The major contribution of this study to the literature is the development of the theory of open source ideology. Our theory features three characteristics: *comprehensive*, *contemporary*, and *empirically-grounded*.

A Comprehensive Framework

The study realized the transition from *fragmented* knowledge pieces of open source ideology to a *comprehensive* framework. Open source ideology in extant literature appeared as knowledge fragments, and mostly limited to particular topics, such as gender, fairness, and community, as shown in Fig. 2.2 [117]. Our empirical theory in total contained 42 themes under six broad categories, i.e., *Norms/Values*, *Membership*, *Goals*, *Activities*, *Resources*, and *Positions/Group Relations*. The topics in extant literature are all covered in our empirical theory, thus, our empirical theory helps organize, connect, and synthesize extant literature regarding open source development systematically. While the newly added themes would inform researchers various future research opportunities, and reminder them to not overly focus on a narrow set of themes.

Moreover, the empirical theory with these themes is organized under a cohesive framework, and provides a panoramic overview of open source ideology for research community. Such an advance facilitates multidisciplinary academic exchanges, and offers references for future research involving ideological issues in the open source movement. The empirical theory provides a terminological system that allows the discovery of analogy or relation between two academic fields, which leads to each helping the other's progress, repeatedly proven in the modern history of natural and social sciences [1].

A Contemporary Understanding

Being *contemporary* contained the meanings of two aspects. First, when people thought of "open source ideology", they might come the the early-days manifestos, i.e., Eric Raymond's *The Cathedral and the Bazaar* [91]. However, after two decades, open source movement has became a polymorphic movement involving millions of diverse participants, which brought new ideas and interpretations into open source ideology. For example, former Microsoft CEO Steve Ballmer once said that "Linux is a cancer that attaches itself in an intellectual property sense to everything it touches," and considered open source software as a significant threat to Microsoft's business [5]. However, Microsoft later changed the stance, started embracing open source development, and heavily involved in various open source projects [114]. Thus, modern theories should update these early manifestos and metaphors, and our empirical theory could serve this purpose well.

Second, being contemporary also means that our theory reflects the current social reality of the open source movement. It servers as a snapshot of current open source movement, also as a referential point when examining the future evolution of the open source movement. In addition, we identified some tensions and ongoing discussion existed in current open source ideology, such as ethical licensing, and paid participation. Take ethical licensing as an example, current open source licensing allow any individuals and organizations to use open

source projects, but open source practitioners started to be concerned about using open source projects for unethical purposes, such as malware development, and mass surveillance and censorship. Thus, some informants shared ideas of ethical licensing in open source movement:

> "*The moral problem [in open source development], the ethical open source licenses, for example, licensee shall use the software in a manner consistent with the human rights principles, where the human rights principles are defined, among other things, in the United Nations Universal Declaration of Human Rights.*" (**Luis Villa**)

These tensions and ongoing discussions indicated potential evolution of open source movement, and shed lights on the future directions of open source movement.

An Empirically-Grounded Approach

Ideological theories were conventionally developed deductively from several key principles rather than inductively from empirical evidence, which neglected the direct experiences and opinions of members of social movements, particularly the grassroots [92]. Extant literature usually explored open source ideology with high-profile open source projects through case studies [24]. However, in our study, we collected data from opinion leaders, experts, and first-line contributors in open source community, a broader and more diverse sample of open source practitioners. Furthermore, we followed grounded theory methodology, to inductively develop the empirical theory of open source ideology [17, 80]. Such a methodological shift is not arbitrary, it is based on the critical changes of the open source movement in the last couple of decades. Open source software has penetrated almost every aspect of our society, and no longer a hacker culture [41, 16]. Both the collected data and research methodology are

empirically-grounded thus provides a partial remedy by improving the breadth of theoretical scope. The enhancements to the extant work evidence that exactly. Moreover, an empirical theory could also help avoid metaphysical arguments of ideologies and inform practitioners of some applicable solutions [10].

The empirical theory also enhanced our understanding of many phenomena in open source development across micro-, meso-, and macro-levels. The discussions of the newly identified themes suggest various potential research opportunities. We list a few as examples in the form of research questions:

- How to deal with identity struggles of members from companies sponsoring the project? How to reduce tensions and conflicts resulting from that? (*Membership*, see **§3.2.2**)
- What are the impacts of licensing on open source development? How to encourage ordinary contributors' involvement in licensing conversations? How to reduce the legal complications in open source development? (*Activities*, see **§3.2.4**)
- How do routines evolve in open source development? How do the 3Rs' channeling functions work? (*Activities*, see **§3.2.4**)
- How to balance companies' donations and projects' independence? How to design effective and efficient mechanisms to fund load-bearing contributors? (*Resources*, see **§3.2.5**)
- How to ensure that commercial agencies honor rather than distort open source ideologies in their business agenda? What should be done to compete contributors' minds with free software? (*Position and Group Relations*, see **§3.2.6**)

Note that the above research questions have not enumerated all potential opportunities inspired by the empirical theory. Then also exhibits the breadth and depth of the knowledge

body around open source ideology, where interdisciplinary research becomes the unrivaled choice to tackle them.

3.3.2 Practical Implications

Anchors for Dealing with Ideological Issues

Open source practitioners have to deal with ideological issues. In our collected data, **Tracy Hinds** mentioned that projects' conflict management involved many ideological issues. However, people may overlook some critical ideological elements. Our theory provides anchors for them to fulfill such tasks. Besides, many communities would like to explicitly establish ideological guidelines for regulating individual and organizational practices and branding themselves, but in an overly simplified way. One example is the "Four Opens" of OpenStack. While the "Four Opens" contains some ideologies, e.g., "Open Community" means "ensuring that the community is a cohesive, inclusive, level playing ground where all the voices are heard and anyone can rise to leadership positions," many other ideological elements are ignored in the guidelines but caught in our theory. For example, **Eric Raymond** emphasized that infrastructure developers build and maintain the most critical backbones of the Internet but often have to worry about tomorrow's lunch, because they have little means to monetize their work. As an infrastructure foundation, OpenInfra should make some resources for supporting these developers' basic living needs explicitly, but not yet. With our theory of open source ideology, practitioners could check if their ideological guidelines include all necessary parts without missing any important ones.

Theory as An Assessment Framework

The empirical theory could also serve as an assessment framework for open source practitioners to investigate if contributors and projects uphold the desirable ideologies and to what extend. Researchers revealed that hte misfit of open source ideology between contributors and projects could influence open source development, e.g., the under-fit of open source ideology (individuals embrace OSS ideology more than they perceive others do) decreases individuals' commitment to open source community [26]. Thus, such assessment is of high necessity to facilitate efficient open source development. Our empirical theory set a solid foundation for developing an assessment system for ideologies, whether qualitative or quantitative [31]. Such as assessment system could help identify the misfit, avoid potential conflicts in open source development, and help match projects and contributors with similar open source ideologies to improve productivity.

Chapter 4

Developing an Assessment Framework of OSS Ideology

4.1 Motivation

The aforementioned theory of open source ideology provides an overview of ideologies within the open source community, it reflects the ideologies that are commonly shared across the open source community. However, the heterogeneity of open source ideology is evident within the open source community. In practice, ideologies upheld by open source projects tend to vary, i.e., some themes of OSS ideology may not be followed in certain open source projects, while some other themes may be upheld, but to various degrees across different open source projects. In addition, the individuals who participate in open source development also perceive open source ideology differently, based on their backgrounds and understanding of open source development.

Extant literature has identified such heterogeneity in the perception and adoption of open source ideology, and further explored its potential impacts on open source development.

Steward and Gosain [102] explored the impact of open source ideology on the project effectiveness, and they found that the adherence to open source ideology would impact the effectiveness of open source development in open source projects, i.e., communication, and trust. Later, Daniel et al. [26] defined the differences in ideology between company employees and open source projects as "OSS ideology misfit", i.e., OSS ideology over-fit means employees perceive that their colleagues embrace the OSS ideology more than they do, while OSS ideology under-fit means employees embrace the OSS ideology more than they perceive their colleagues do. Their investigation revealed that both OSS ideology under-fit and over-fit would decrease employees' commitment to open source project, but OSS ideology over-fit would increase employees' commitment to the company. Both studies indicate the existence of heterogeneity of open source ideology, and its significant impact on open source development. Therefore, it is a necessity to be able to assess how individuals and open source projects perceive and follow open source ideology in practice.

Although the extant literature developed some methods to assess the ideologies perceived by individuals in open source development or upheld within open source projects, those assessment methods were constrained by several limitations. First, the assessment was based on a three-tent framework of OSS ideology, i.e., norms, values, and beliefs [102]. Even the authors themselves admitted that such a framework was preliminary, and some other important tenets might not be revealed and included. Thus, it limited the scope of the assessment results and insights. Second, the assessment method, particularly the data collection, was through the survey method, which required researchers to carefully design the survey questions, and then analyze the collected data. Such a method is often knowledge-intensive and labor-intensive, requiring substantial knowledge and effort from researchers. For example, Daniel et al. [26] only sampled contributors from GNOME community to conduct an investigation on the misfit, which limited the generalization of the findings and implications. Thus, the existing assessment methods are not practical, especially for ordinary open source projects and practitioners.

Based on the empirical theory of open source ideology, we intended to develop a more *extensive* and *practical* assessment framework to help the open source community examine OSS ideologies within open source projects. Researchers and practitioners in open source development would be able to conduct assessments on ideologies upheld by particular open source projects, gain insights into open source development from an ideological perspective, and improve the practice of open source development accordingly.

4.2 Designing Assessment Framework

First, we adopted two design principles in designing the assessment framework. The first design principle is *focusing on project-level ideologies*. OSS ideologies could be reflected at micro-, meso-, and macro-levels, i.e., individual open source contributors, open source projects, open source organizations, and the overall open source community. The assessment framework focuses on particular open source projects, and the assessment results should reflect the OSS ideologies within those projects. The rationale for having such design principle is to set an explicit unit of analysis, so that the OSS practitioners could utilize the assessment results accordingly.

The second design principle is *utilizing public data sources only*. The assessment framework is designed to rely on data from public sources, such as project documentation, code repository, and online media outlets. Any individual, no matter whether affiliated with the open source projects, can easily access and collect these data, and then conduct assessment with the framework. It would ensure the robustness and flexibility of the assessment framework.

Then, with the two design principles, we designed an assessment framework based on the empirical theory of open source ideology. Note that the empirical theory of OSS ideology contains 42 themes. With each theme within the theory, we went through a three-step design

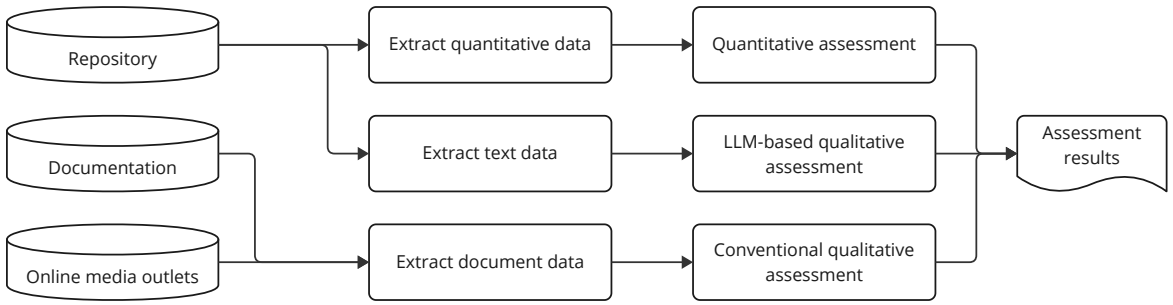

Figure 4.1: Three assessment types in the assessment framework.

process to develop the assessment framework, i.e., brainstorming potential methods, selecting assessment methods, and developing assessment workflow. The first step was brainstorming potential methods. In this step, two researchers worked independently. We reviewed the interpretation of the theme to be assessed, and listed a set of potential assessment methods that could reflect or indicate the theme. Then in the second step, selecting assessment methods, we collectively discussed those potential assessment methods from the previous step, and selected one or two methods as the assessment methods for the theme. The selected methods might assess the given theme either quantitatively or qualitatively. In the third step, we further developed a workflow to conduct assessment on the given theme, with the selected assessment methods. The workflow generally included the type of data required for the assessment, data processing, data analysis, and deriving assessment results. In total, following this process, 32 of 42 themes within the empirical theory of OSS ideology were operationalized, which constituted the assessment framework (see Appendix B for more details). Note that there were ten themes we were unable to operationalize following the two design principles, for example, the theme *Individual Motivations* mainly reflects OSS ideology at the individual level, and are usually assessed through questionnaires and interviews with contributors [46], rather than observed from external views, thus, this theme is excluded in the assessment framework.

In general, there are three different types of assessments in the workflows, depending on the data and the assessment methods, i.e., semi-automated quantitative assessment, Large

Language Model (LLM)-based qualitative assessment, and conventional qualitative assessment, as shown in Fig. 4.1. The first type is semi-automated quantitative assessment. It collects and extracts quantitative data from the project repository, such as commit time data, and then processes and generates the results to reflect the themes. The results are often presented as charts, e.g., heatmap charts and bar charts. Such type of assessment could be implemented by computer programs and scripts, which could be semi-automated in the assessment framework.

The second type is LLM-based qualitative assessment. We utilize LLM to facilitate the analysis of text data extracted from the project repository, such as issue comments. The rationale for utilizing LLM is determined by the large volume and complexity of the collected text data. Within an active open source project community, there are often lots of contributors who actively participate in various discussions related to the project, e.g., reporting bugs, requesting new features, reviewing pull-requests, etc. Some of the text is even embedded with source code and system logs. All of those make human analysis very costly or even impossible. Thus, some tools such as LLM could help perform such Natural Language Processing (NLP) tasks in a more efficient way. Furthermore, there exist various tools for NLP tasks in the literature and the marketplace. We choose to use LLM, such as GPT-3.5, rather than some SE-specific tools, considering multiple types of NLP tasks in the assessment framework, e.g., sentiment analysis, and trust estimation. Most SE-specific tools are designed for particular NLP tasks [82], e.g., Senti4SD for sentiment analysis only [14], which requires us to utilize different tools for different NLP tasks, with potential model tuning. While LLMs such as GPT-3.5, as off-the-shelf models [96], are capable of various NLP tasks, and are often provided as a cheap and reliable online service. Thus, it is a practical choice to utilize LLM for text analysis in the assessment, Additionally, LLM could also automate the process. In general, we only need to compile the collected text data, send it to LLM with task prompts, and summarize the results, without dealing with different tools for different tasks.

The last type of assessment is conventional qualitative assessment. With the collected documents from the project documentation and online media outlets, we intend to identify and review the content that related to the themes to be assessed, and derive insights as the assessment results. Generally, as a rigorous assessment, multiple individuals are involved to review the collected documents, and then discuss collectively to derive the insights. However, it is not necessary if the assessment is not for academic inquiries. Such a decision should be based on the users of the assessment framework and the purpose of the assessment.

Note that more than one type of assessment might be adopted in the workflows that form the assessment framework. For example, when assessing the theme *Meritocracy* (see Fig. 4.2), it starts with the conventional qualitative assessment to identify different roles in the project community, some typical roles could be contributor, maintainer, etc. Then, if such roles exist in the project community, we would conduct a semi-automated quantitative assessment and a conventional qualitative assessment. In the semi-automated quantitative assessment, we collect the list of project contributors and their code commits, then calculate the centrality degree [61] for each contributor, and group the centrality degrees based on the role of contributors in the project. Then we compare and analyze the centrality degrees across those groups to examine if there are significant differences between groups. With the conventional assessment, we collect the project documentation related to these roles, review the collected documents for the content regarding role nomination, removal, or promotion, and generate the review result. Based on the results of two assessments, we could derive some insights into *Meritocracy*, and write the assessment summary.

4.3 Pilot Case Study

With the designed assessment framework for OSS ideology, we conducted a pilot case study. The goal of the pilot case study is to test and refine the assessment framework [116]. Fig.

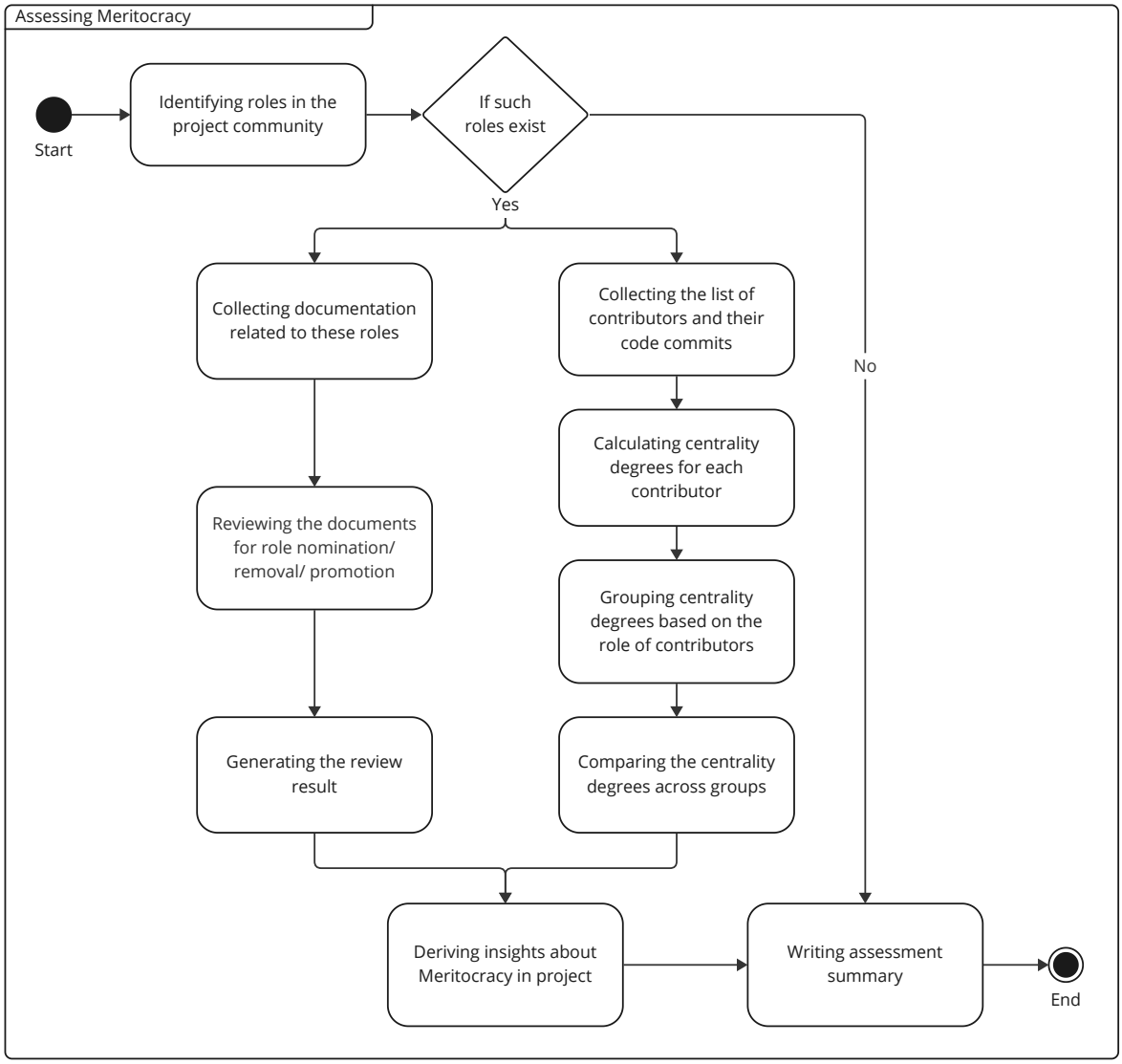

Figure 4.2: The workflow of assessing Meritocracy.

4.3 provides an overview of the process of the pilot case study, and we will present the pilot case study in detail in the remainder of this section.

Pilot Case Study Design

First, we selected D3 [1] as the study subject in the pilot case study. The rationale for choosing this project is twofold. First, D3 project is a popular open source project widely adopted

[1]The project is hosted on GitHub at: `https://github.com/d3/d3`.

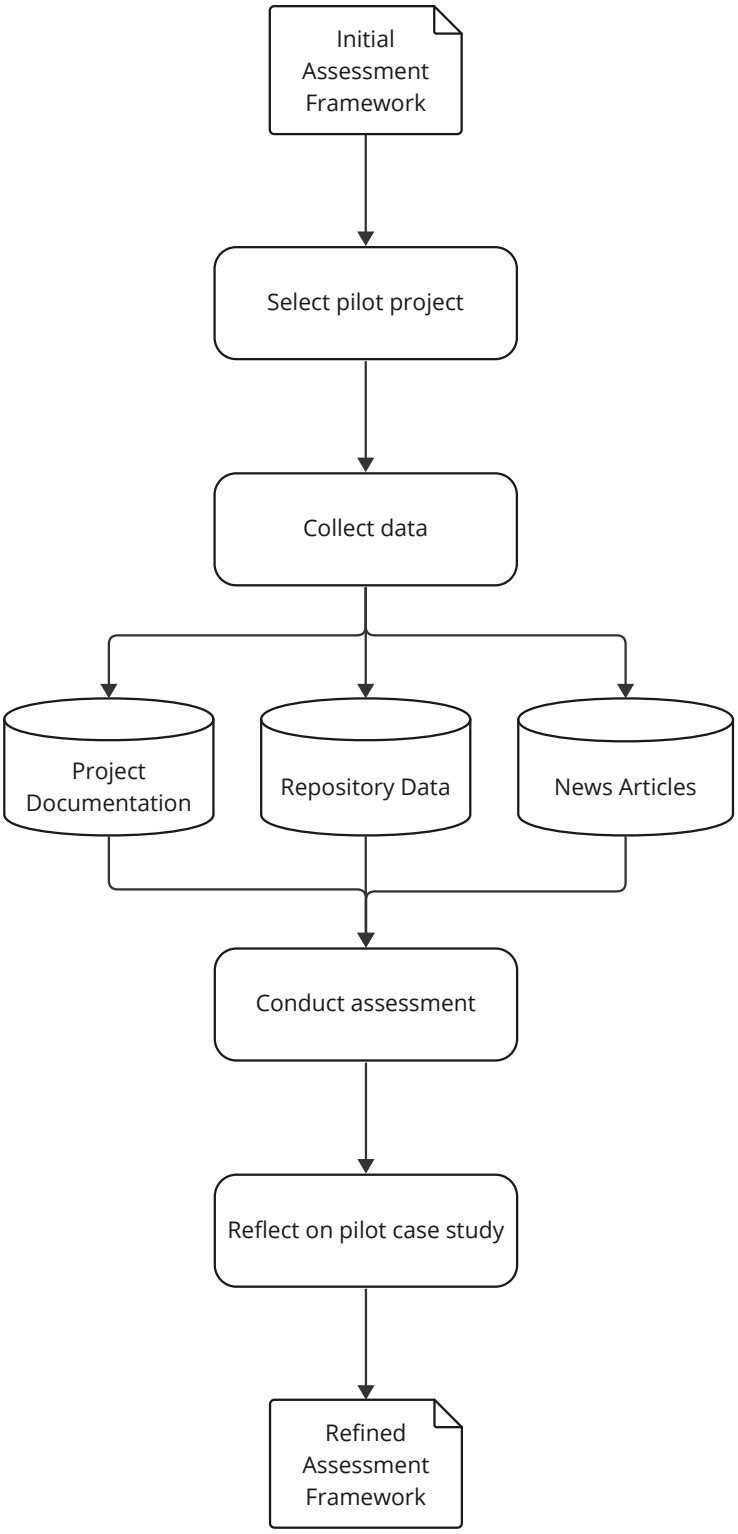

Figure 4.3: The process of the pilot case study.

for data visualization, and it has been established since 2011. The project contributors have formed a community around the project, which could be an ideal subject to assess the OSS ideology within the community. Second, D3 offers a rich yet manageable amount of data for the pilot case study, i.e., 4355 commits, 3224 issues, 9127 issue comments, and 192 releases. We would be able to complete the assessment efficiently for the pilot case study, and then refine the assessment framework based on the results.

According to the assessment framework, we need to collect data from three sources, i.e., the project repository, the project documentation, and online media outlets. The first data source is the project repository. We collected data in two ways, i.e., querying GitHub API, and extracting from the code repository. In general, the commit data, issue data, issue comment data, release data, and contributor data were collected. The second data source is the project documentation. Project D3 provides documentation at two sites: the official project website, and the GitHub repository site. The documents were accessed directly through the hosting websites. The online media outlets are the third data source for the assessment. We searched online for the news articles related to the project, and collected them for the pilot case study.

With the collected data, we conduct the assessment on OSS ideology within D3 community. Following the workflow of each theme in the assessment framework, we analyzed the 32 themes one by one. Then, based on the process of conducting assessments and the assessment results, we further reflected on the pilot case study, and refined the assessment framework.

Result of Pilot Case Study

Tab. 4.1 provides a summary of the assessment results on category *Norms/Values* in the pilot case study, as an example to demonstrate the results of the assessment on OSS ideology in D3. Note that the goal of the pilot case study is to test and refine the assessment framework,

thus, we primarily focus on the assessment framework and the workflows, and the detailed case study results will be presented in the next chapter.

Table 4.1: Summary of the assessment results on *Norms/Values* in pilot case study.

Theme	**Assessment Result**
Appreciation & Recognition	Sometimes express the appreciation to the contributors but not always.
Autonomy	A large number of contributors might voluntarily contribute, while most commits were submitted during working time.
Comparative Advantages	Several advantages in data visualization were listed with particular requirements.
Constantly Changing & Evolving	The community tends to be stable rather than changing or evolving.
Ethics	Explicit guidelines on several ethics values, as well as reporting and enforcement process.
Openness & Transparency	Limited guideline on openness, but nothing about transparency.
Meritocracy	No evidence.
Reciprocity at Multiple Levels	Reciprocity exists, and tends to be positive and supportive.
Trust	Trust exists, and is evident in collaboration, community support, etc.
Universal Accessibility & Availability	Public code repository, allowing to view/fork, and submit pull-requests, but no explicit documents.

Framework Refinement & Reflection

Two researchers who conducted the pilot case study collectively reviewed the whole process of conducting the assessment, and refined the assessment framework accordingly. The refinement of the workflow could be classified into three types:

- Adding new methods to the workflows. We noticed that some of the assessment methods might only partially reflect the given theme, or might not adequately capture the essence of the given themes. Then we added new methods to enhance the assessment workflow. For example, the initial workflow of assessing theme *Innovation* focused on

the issues related to new features or functions. Then we added a sentimental analysis of the comments on those issues, to reflect the community's reaction on those new features or functions, such as accepting/welcoming the ideas, or rejecting the ideas. The additional method could provide more insights into the *Innovation* within the open source project.

- Improving current methods in the workflows. When following some of the workflows to assess the themes, some issues emerged, particularly with some steps within the assessment methods. We revised those steps with more well-defined instructions or adjusted some steps. For example, in the workflow of assessing *Ideological Leadership*, we revised the step of "analyzing the comments with LLM tool" to "identify leadership content in the comments with LLM tool," which tends to be more explicit. Another example was assessing *Individual Identity*, in the initial method, we grouped the comments by contributors before the analysis. Then we realized that the results would reflect individuals rather than the whole project community. Thus, we removed such grouping step, and analyzed all the comments directly.

- Replacing with new methods in the workflows. After reviewing the results of the pilot case study, some of the methods might not be ideal for reflecting the given themes, thus, we need to replace those methods. Take assessing *Personal Development* as an example, we intended to find evidence in the comments that contributors indicated that they learned from the project, or improved themselves in the project. However, contributors rarely shared those through comments on the projects, they might mention that on their personal websites or blogs, which is out of the scope of our data collection. Therefore, we adopted contributors' trace of centrality degrees within the project as the indicator of personal development, and updated the workflow accordingly.

Furthermore, the pilot case study also provides some guidelines regarding the usage and implementation of the assessment framework. First, there are some traditional qualitative

methods in the assessment framework. In the pilot case study, two researchers were involved in the assessment process, which is a typical strategy in Consensus Qualitative Research (CQR) to guarantee the quality of the analysis [53]. However, it is not specified in the assessment framework that multiple individuals are required for the assessment, instead, we only specify the general assessment workflow. The rationale was that we aim to provide a practical assessment framework with high usability. Both open source practitioners and researchers could utilize the framework to conduct assessments on OSS ideology, i.e., following the same workflows, researchers could adopt a rigorous CQR process to conduct qualitative assessments, while individual practitioners could also conduct the same assessment on their own and gain some insights into open source projects.

Second, LLM could be a reliable proxy in analyzing large volumes of text data, and extracting key information for the assessment. However, we encountered some challenges in prompt engineering and hallucination. There were several NLP tasks included in the assessment framework, such as sentiment analysis and trust estimation. We need to develop appropriate prompts for those NLP tasks, since task performance significantly depends on the quality of the prompts [128]. For example, when extracting text data from issue comments related to discussions among different contributors, and using LLM to analyze the sentiment, the LLM occasionally failed to perform the NLP task, instead, it generated responses such as "You're welcome! If you have any more questions or need further assistance, feel free to ask," as if it were participating in the discussion. Since we adopted GPT-3.5 provided by OpenAI in the pilot case study, we followed the official guideline on prompt engineering [83] to develop prompt templates for each NLP task. Note that the prompts are closely related to the LLM adopted in the assessments, prompt engineering is required when adopting different LLMs or even different versions of the LLM.

Hallucination is another challenge with LLM. Although the research community has dedicated lots of efforts to mitigate hallucination, LLM still generates hallucinate unintended

text [62]. To reduce hallucination and its negative impact on the assessment, we adopted two strategies in the pilot case study, one is aforementioned prompt engineering, and the other is answer traceability. With answer traceability, we recorded both the requests sent to LLM and the responses received from LLM. With these records as references, human experts could efficiently examine whether the hallucination happened in the particular response, then remove or current them to avoid misinformation. Thus, answer traceability is recommended when implementing the assessment framework with LLM to reduce the impact of hallucination.

Chapter 5

Applying Assessment Framework on Open Source Project: Case Study

We conducted three case studies, by applying the refined assessment framework on open source projects, to investigate how those projects follow the OSS ideology in practice, and gain insights into the open source movement.

5.1 Study Design

In general, the study included six key steps, i.e., selecting projects, collecting data, conducting assessments, writing assessment reports, collecting feedback, and conducting cross-case comparisons. Fig. 5.1 shows the overall process of the case studies.

First, we selected three open source projects as the subjects for case studies, i.e., D3, Zephyr, PyTorch. D3[1] is an open source JavaScript library for data visualization, es-

[1] https://d3js.org/

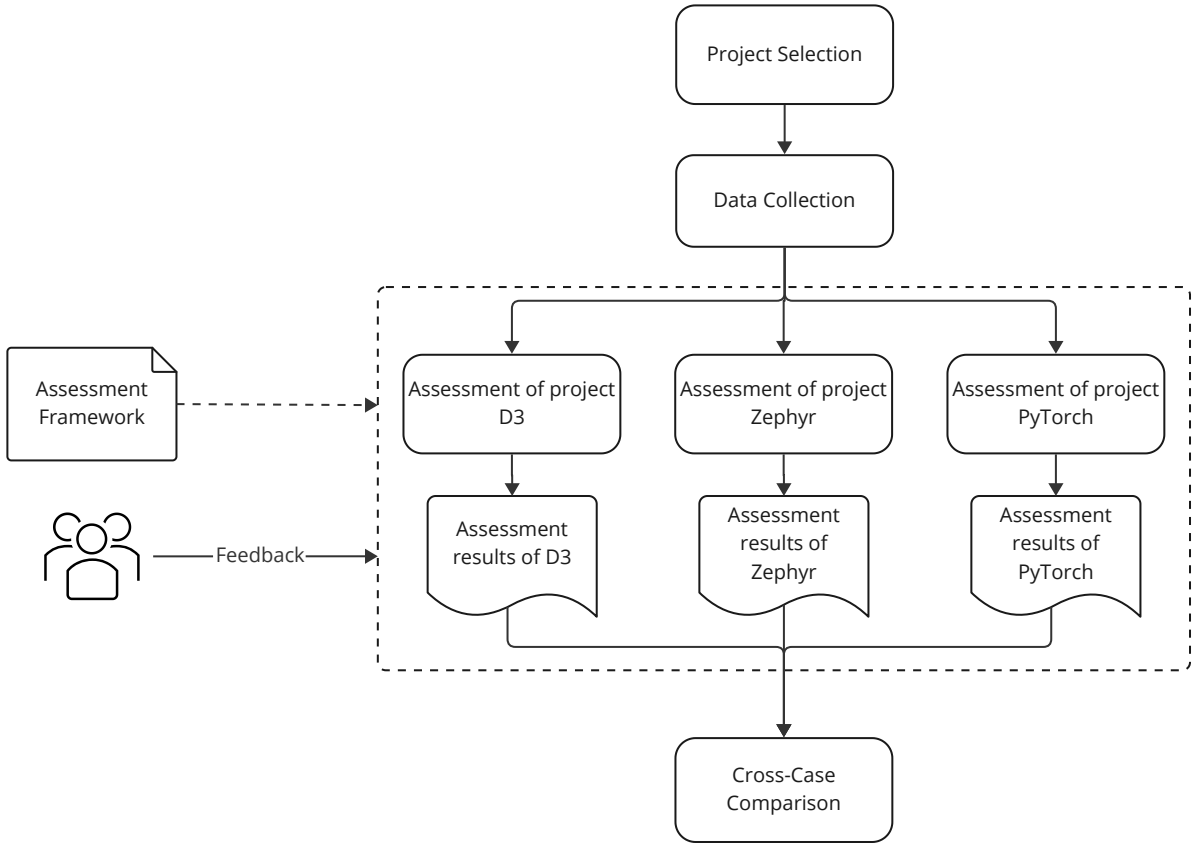

Figure 5.1: The process of three case studies.

tablished in 2011; ZEPHYR[2] is an open source real-time operating system since 2016; PYTORCH[3] is an open source framework widely adopted for deep learning, launched in 2016. These three projects are popular open source projects that attract lots of contributors to build communities around them, which are ideal subjects to assess the OSS ideology within their communities. Then all three projects host their project repositories on GITHUB, thus, making it convenient to collect data under the same schema. Moreover, these three projects could well represent diverse open source projects and communities, i.e., D3 was released as an open source project in data visualization field but solely governed by Observable; ZEPHYR originated as a commercial project but later became an open source projects under Linux Foundation; PYTORCH represents the machine learning/deep learning community, and it

[2]https://www.zephyrproject.org/
[3]https://pytorch.org/

was initially developed by Meta (formerly known as Facebook), then released as an open source projects governed by PyTorch Foundation. The diversity and representativeness of the three projects could provide various insights into open source movement in the case studies.

Then, with the selected three open source projects as case study subjects, we collected data for assessment. The data sources were the same as those in the pilot case study, i.e., project repository, project documentation, and public news (see Sec. **??**). Tab. 5.1 shows the collected data from project repositories.

Table 5.1: Summary of the collected data from project repositories (April 2023).

Project	**#Commits**	**#Issues**	**#Issue Comments**	**#Releases**
D3	4,355	3,224	9,127	192
Zephyr	78,319	56,363	174,183	106
PyTorch	58,701	98,659	404,614	44

With the collected data, we followed the same procedure in the pilot case study to conduct assessments with each of the selected projects. Furthermore, based on the assessment results of three study subjects, we compared across three study subjects to gain more insights into OSS ideologies in practice.

After completing the assessment, we also recruited OSS practitioners to collect their feedback on the assessment. The goal of collecting feedback was not to examine the assessment results, but to investigate if the assessment results could provide valuable information for OSS practitioners to better understand the projects, such as inspiring some reflection on open source development, and providing information to facilitate decision-making. To achieve this goal, we recruited three participants to conduct feedback sessions. The criteria to recruit participants included familiarity with open source development but not limited to the study subjects; independence from our case studies; and the availability to provide feedback after reading the assessment results. Three participants were recruited from our department

based on the criteria. The assessment results were sent to the participants as a report, and their feedback was collected through interviews. Since the feedback sessions aimed at the assessment in general, rather than particular open source projects, it was not necessary to conduct feedback sessions in all three case studies. Therefore, we only conducted such feedback sessions in the case study of Zephyr with multiple participants.

5.2 Three Case Studies

In this section, we present the results of the case study, i.e., the assessment results of each selected open source project. To keep this section concise, we briefly summarize the overall assessment results[4], and then present assessment results of some selected themes in detail, which serve as best practice examples.

Disclaimer: The assessment results were based on public data sources, some of them might not represent the official opinion of the projects. The assessment we conducted was a snapshot of the OSS ideologies in the project at a particular time, the assessment results for particular projects were limited to this snapshot, and could not reflect the future changes within the projects. The data and assessments were indicators that reflect the themes, they might not reflect all practices related to the assessed themes.

5.2.1 Case 1: D3

We were able to compile evidence from the collected data to assess 25 out of 32 themes in the project, and the remaining seven themes remained unclear with the collected data. Note that it did not necessarily indicate the non-existence of those seven themes in the project. Rather, it suggested that we were unable to assess them with the current assessment framework. The

[4]see Appendix C for a detailed report.

assessment results are summarized as follows:

- **Values/Norms**: eight of the ten assessed themes were evident in the project, except *Constantly Changing & Evolving* and *Meritocracy*. The project sometimes expressed *Appreciation & Recognition* in the release notes and the comments, but only a few times. A large number of contributors were not affiliated with the project, and the submission time of code commits tended to spread across both working time and after-work time, which indicated *Autonomy* in the project. Several *Comparative Advantages* were listed in the documentation, such as "high flexibility for creativity," "works directly with web standards," and "maximal expressiveness for bespoke visualization." Both the monthly commit frequency and newcomers tended to be stable and slightly decreasing, which indicated the project was not *Constantly Changing & Evolving*. The Code of Conduct documents provided guidelines on some values of *Ethics*, as well as the reporting and enforcement process. For *Openness & Transparency*, there were limited guidelines on openness, but nothing about transparency. No evidence was found to reflect *Meritocracy*, as no explicit description or different roles in the project. *Reciprocity at Multiple Levels* existed in the project community, and tended to be positive and supportive in general. *Trust* was positively identified in the project community, and was evident in collaboration, community support, etc. No documentation related to *Universal Accessibility & Availability*, but the code repository on GitHub could partially guarantee that.

- **Goals**: five themes were assessed, except *Innovation*. *Built-in Product & Quality Orientations* were reflected by the relatively stable bug reports and fixes. According to the documentation, the major *Broader Impacts* of the project was to make data visualization easier and more accessible for both technical and non-technical users. The project utilized various social media platforms, e.g., YouTube, and X/Twitter, and meetup events to achieve the goal of *Outreach*. The project has been widely adopted

in data analysis and visualization fields, in both industry and academia, which reflected its *Ubiquitous Penetration*. The number of newcomers who participated in code contribution and issue discussion tended to decrease, which indicated some potential challenges with *Sustainability*. Since there was no way to identify new feature/function requests, *Innovation* remained unclear.

- **Activities**: we were only able to assess four of five themes with supporting evidence. For *Copyright, Licensing, & Legal Implications*, the project was under ISC license, a permission open source license. No Contributor License Agreement or other documents are required for external contribution. *Governance & Decision Making* was unclear, since there was no description in the documentation. *Personal Development* was only evident with some of the contributors, while the centrality degrees of the majority of contributors were relatively low and not in an increasing trend. *Social Production* was reflected by multiple contributors involved in issue discussions. For *Work Organization & Practice*, multi-timezone collaboration existed but was not prevalent, since the majority of code commits were submitted from three timezones; GitHub issues, community forum, and Slack were three communication channels, but exclusively for project development.

- **Membership**: only three of four assessed themes were reflected in the assessment. For *Individual Identity*, most contributors addressed themselves as individuals, while only a small number of them addressed themselves as part of the community. *Ideological Leadership* was evident in the issue discussions, particularly in providing guidance, sharing important information, etc. *Interpersonal Sentiment* in the project tended to be neutral to slightly positive, with friendly, supportive, and collaborative sentiments. No documentation was found to describe *Member Hierarchy & Roles*.

- **Resources**: three of the four assessed themes were reflected. For *Dealing with Barriers & Restrictions*, the project provided some guidelines about "Getting Involved",

but it was primarily for the project users rather than developers. No resources related to *Incentives, Financing, & Funding* were found in the project. *Knowledge & Expertise* were only limited to detailed documentation on project APIs and tutorials. No documentation on *Supportive Facilities & Mechanisms*, except adopting GitHub for development, and Slack for communication.

- **Position/Group Relations**: only two of three themes were indicated by the collected data. No documentation or news articles indicated *Interaction with Emerging Technologies*. For *Interaction with Commercial Software Development*, only one company, Observable, participated in the development, since the company owned and managed the project; the project was adopted in the company's commercial product/service, and no other adoption identified. The project's *Market & Users* were primarily data analysis and visualization market, e.g., media, consultant, and users with those data analysis and visualization requirements.

Since most of the assessed themes had room for improvement based on the results, it was impossible to find one as the example of the best practice in this case study. Thus, we will not select a theme and present it in detail.

5.2.2 Case 2: Zephyr

In general, we found evidence to assess 31 out of 32 themes in the project, *Broader Impacts* was the one we were unable to find evidence. The assessment results are summarized as follows:

- **Values/Norms**: all ten themes were evident. The project would sometimes express *Appreciation & Recognition* to contributors in the release notes and the comments of pull requests, but such behaviors often depended on the discretion of the members or

reviewers involved. The majority of contributors were not affiliated with the project, and some code commits were submitted outside of the regular working time, while the majority were submitted during working time, which indicated *Autonomy* existed in the project but to a limited extent. The project's *Comparative Advantages* included "extensive suite of kernel services," "cross architecture," "memory protection," and so on. The monthly commit frequency and newcomers were increasing, which indicated *Constantly Changing & Evolving* in the project community. We found a Code of Conduct document to discuss some values of *Ethics* within the project, as well as some examples of acceptable/unacceptable behaviors and enforcing processes. *Openness & Transparency* was partially addressed in the documentation, i.e., mentioning openness as one of the values in the community without emphasizing transparency. *Meritocracy* existed in the community according to centrality degrees of different role groups, but lacked of explicit description in the documentation. *Reciprocity at Multiple Levels* tended to be neutral most of the time, while slightly positive in some discussions. *Trust* was evident, and tended to be at a high level. *Universal Accessibility & Availability* was mentioned in the documentation, and also guaranteed by the hosting repository on GitHub.

- **Goals**: we found evidence to reflect five themes, except *Broader Impacts*. *Built-in Product & Quality Orientations* was indicated by the slightly increasing trends of reported and fixed bugs. The project took advantage of multiple social media platforms, such as Facebook and YouTube, as well as meetup events, to achieve the goal of *Outreach*. *Sustainability* was evident with newcomers joining every month for code contributions and issue discussions. *Innovation* was reflected by the slightly increasing trend of new features/functions requests, but the general sentiments towards when were not always welcoming. *Ubiquitous Penetration* was achieved by the wide adoption of the project in many commercial products.

- **Activities**: all five assessed themes were evident. The project followed a good practice regarding *Copyright, Licensing, & Legal Implications*, i.e., adopting standard permissive open source license, and not requesting copyright transfer from contributors. A Technical Steering Committee (TSC) formed by a group of contributors was responsible for *Governance & Decision Making*, and also reflected true meritocracy. All contributors' centrality degrees were increasing, which indicated *Personal Development* in the project. *Social Production* was evident as the majority of issue discussions were involved with more than one contributor. *Work Organization & Practices* was reflected by multi-timezone effort on the development, and different communication channels to facilitate collaboration.

- **Membership**: all four assessed themes were reflected in the assessment. For *Individual Identity*, a limited number of contributors addressed themselves as part of the community, while many of them tended to address themselves as individuals. *Ideological Leadership* was reflected in the issue discussions, primarily focusing on technical aspects. *Interpersonal Relationships* generally tended to be positive, with professional and collaborative sentiments. For *Member Hierarchy & Roles*, three roles, i.e., Maintainers, Collaborators, and Contributors, were clearly defined, and they formed a hierarchical structure in the project.

- **Resources**: all four assessed themes were indicated in the project. Detailed contributor guidelines were provided for *Dealing with Barriers & Restrictions*. Some *Incentives, Financing, & Funding* were identified, i.e., members for project sponsorship, and career opportunities within the project ecosystem and community, but no support for individual donations. *Knowledge & Expertise* were reflected by detailed and well-structured documentation for both developers and users. A set of tools/toolchains and three different communication channels were adopted as *Supportive Facilities & Mechanisms*.

- **Position/Group Relations**: all three assessed themes were supported by the evi-

dence from the documentation and related news. Robotics, AI, Ultra-Wideband RTLS would be *Emerging Technologies* that the project mostly interacted with. *Interaction with Commercial Software Development* was reflected in the official website and news articles, i.e., commercial companies such as Google and T-Mobile were platinum/silver members in the project community, and their employees were active in the project; the project has been widely adopted in many commercial products. The project's *Market & Users* were primarily IoT and embedded system markets, and the developers in those fields.

Next, we choose the assessment on the theme *Copyright, Licensing, & Legal Implications* as an illustration example in this case study. First, the project was under Apache 2.0 license, a permissive open source license. It is a standard license without any modification or customization by the project. The license was listed in the code repository, with a short explanation of the rights granted to the public, i.e., "allows you to freely use, modify, distribute, and sell your own products that include Apache 2.0 licensed software." [121] Moreover, the licensing documentation also informed that some components were under different licenses, and provided a list of those components and the corresponding licenses.

Second, unlike some open source projects that require contributors to sign a Contributor License Agreement (CLA) to transfer the copyright, Zephyr only required a signed Developer Certification of Origin (DCO), attached to contributions. A DCO is a short legal document to certify that the contributors own the work and have the right to submit to the project. It does not require contributors to transfer the copyright of their work. The signing process tends to be simple and easy, i.e., adding the statement "Signed-off-by: Name <Email>" in the commit message. Compared with CLA, DCO tends to be less complex and intimidating for contributors, since it does not involve copyright transfer, but still ensures licensing criteria are met.

In summary, the project was under a standard permissive open source license, with an explanation of the rights granted by the license. The DCO simplified the legal implications, which potentially made contributors less intimated by the licensing and legal issues.

5.2.3 Case 3: PyTorch

All 32 themes were reflected by the evidence in the project, and the assessment results are summarized as follows:

- **Values/Norms**: all ten themes were reflected in the project. The project only expressed *Appreciation & Recognition* to the community in general, but did not explicitly thank individual contributors. The majority of the contributors were not affiliated with the project, but the majority of the code commits were submitted during working time, which reflected a limited extent of *Autonomy* in the project. The project documentation listed several *Comparative Advantages*, such as "distributed training," and "robust ecosystem." Both monthly commit frequency and newcomers were increasing in general, which indicated the project community was *Constantly Changing & Evolving*. The project followed a standard Code of Conduct with explicit guidelines to deal with *Ethics* related issues. *Openness & Transparency* were reflected by the public repository on GitHub, and were partially addressed in documentation, i.e., "an open and welcoming environment." *Meritocracy* existed in the project, different roles existed in the project community, and their centrality degrees indicated merit-based promotion. *Reciprocity at Multiple Levels* tended to be neutral mostly, but sometimes slightly positive. *Trust* was evident in the project, and tended to be strong. *Universal Accessibility & Availability* was guaranteed by the public code repository, but direct code commit was only restricted to particular contributors.

- **Goals**: all six themes were evident in the project. *Built-in Product & Quality Orienta-*

tion was indicated by the increasing number of bug reports, and clear documentation to explain the project testing; but the overall bug-fix ratio was relatively low. The project's *Broader Impacts* was presented as "providing advanced and accessible deep learning tooling to the public," and "fostering collaboration on development/research at local and regional levels." The project utilized various popular social media platforms, and conference events to engage with the public, for *Outreach* purpose. *Ubiquitous Penetration* was reflected by the wide adoption of the project as the key machine learning infrastructure. Newcomers actively joined and participated in the development process, i.e., code commits, and issue discussions, which indicated *Sustainability* in the project. *Innovation* was evident in the project, the number of proposed new features/functions was increasing in general, and the sentiments towards them tended to be positive and enthusiastic.

- **Activities**: all five themes were reflected in the project. For *Copyright, Licensing, & Legal Implications*, the project was under BSD-3-Clause license, a permissive open source license, and a Contributor License Agreement (CLA) was required for contributors. For *Governance & Decision Making*, the project community followed a hierarchical governance structure, and collective discussion in the decision-making process. The centrality degree traces of most contributors tended to be increasing, indicating *Personal Development* in the project. *Social Production* was evident, as most issues were commented by multiple contributors. For *Work Organization & Practice*, four communication avenues were adopted for development, i.e., RFC, Design Docs, forum, and Slack, as well as developer meetup events; the code commits were submitted mostly from three timezones, while a small portion was submitted from 18 different timezones.

- **Membership**: all four themes were evident in the project. For *Individual Identity*, many contributors tended to address themselves as individuals, while some contributors

addressed themselves as part of the community. *Ideological Leadership* was reflected in issue comments, primarily focusing on technical aspects of the project development. *Interpersonal Relationships* tended to be positive, with collaborative, supportive, and professional sentiments. For *Member Hierarchy & Roles*, five roles, i.e., Lead Core Maintainer, Core Maintainers, Model-level maintainers, Library-level maintainers, and Contributors, were clearly defined in the documentation.

- **Resources**: all four themes were reflected in the project. The project provided detailed documents on contribution guide, onboarding guide, developer workflow, etc., for *Dealing with Barriers & Restrictions*. For *Incentives, Financing, & Funding*, project donation and sponsorship were through PyTorch Foundation, but no support for individual developers. *Knowledge & Expertise* were reflected with detailed documentation for both developers and users. Documentation, bots, automatic tools, and meetup events were *Supportive Facilities & mechanisms* in the project.

- **Positions/Group Relations**: all three themes were reflected by the collected data. For *Interaction with Emerging Technologies*, the project primarily focused on machine learning/deep learning technologies, and also expanded to edge devices. For *Interaction with Commercial Software Development*, many commercial companies participated in the project development, and the project has been widely adopted in various industries and commercial products. The primary *Market & Users* were related to machine learning and AI, and users included both researchers and developers.

We select *Meritocracy* as an example to illustrate the OSS ideology within project PyTorch. First, five different roles, i.e., Lead Core Maintainer, Core Maintainers, Model-level maintainers, Library-level maintainers, and Contributors, were clearly defined in the project documentation, as well as their rights and responsibilities. For example, the responsibilities of Core Maintainers included "articulating a cohesive long-term vision for the project," "negotiating and resolving contentious issues in ways acceptable to all parties involved," and

"receiving broad requests for changes from stakeholders of PyTorch and evaluating/accepting them." [86] Moreover, the processes of nominating, confirming, and removing those maintainers were also explicitly described in the documentation, and merit basis is the first principle in such processes.

Then, we calculated contributors' centrality degrees and grouped them based on those roles, as shown in Fig. 5.2 and Tab. 5.2. In general, as individuals progressed from Contributors to Lead Core Maintainers, their centrality tended to increase, reflecting merit-based promotion in the project. Therefore, we could conclude that *Meritocracy* was evident in the project, it was not only clearly described in the project documentation, but also indicated by the increasing centrality degrees across different roles.

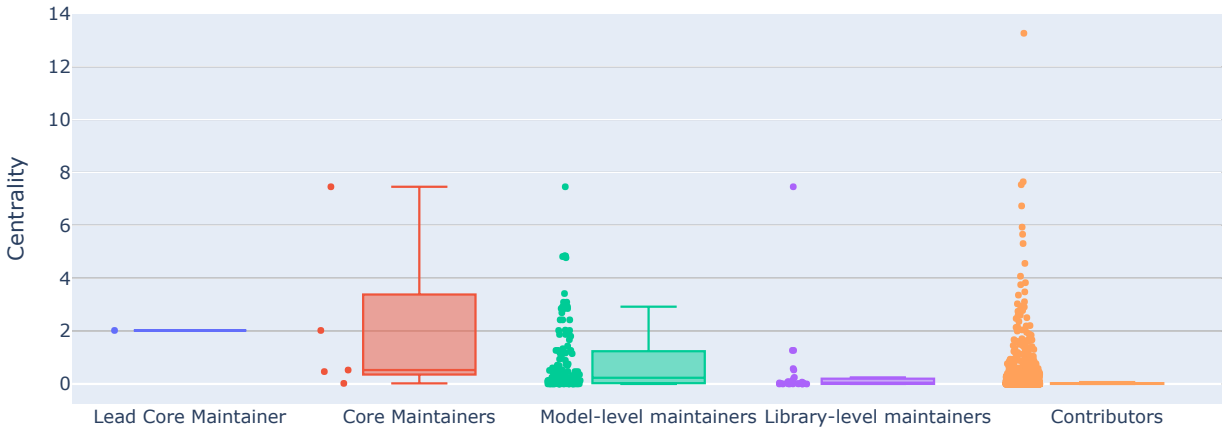

Figure 5.2: The distribution of centrality degrees across five roles in PyTorch.

Table 5.2: The statistical summary of the centrality degrees across five roles in PyTorch.

Role	Mean	Median	SD.
Lead Core Maintainer	2.02	2.02	0.00
Core Maintainers	2.10	0.53	2.76
Model-level maintainers	0.88	0.23	1.33
Library-level maintainers	0.46	0.04	1.44
Contributors	0.08	0.01	0.44

5.3 Cross-Case Comparison & Implications

Table 5.3: The summary of three case studies.

Category	**Theme**	D3	Zephyr	PyTorch
Norms/Values	Appreciation & Recognition	√	√	√
	Autonomy	√	√	√
	Comparative Advantages	√	√	√
	Constantly Changing & Evolving		√	√
	Ethics	√	√	√
	Openness & Transparency	√	√	√
	Meritocracy		√	√
	Reciprocity at Multiple Levels	√	√	√
	Trust	√	√	√
	Universal Accessibility & Availability	√	√	√
Goals	Broad Impacts	√		√
	Built-in Product & Quality Orientations	√	√	√
	Outreach	√	√	√
	Sustainability		√	√
	Innovation		√	√
	Ubiquitous Penetration	√	√	√
Activities	Copyright, Licensing, & Legal Implications	√	√	√
	Governance & Decision Making		√	√
	Personal Development	√	√	√
	Social Production	√	√	√
	Work Organization & Practices	√	√	√
Membership	Individual Identity	√	√	√
	Ideological Leadership	√	√	√
	Interpersonal Relationships	√	√	√
	Member Hierarchy & Roles		√	√
Resources	Dealing with Barriers & Restrictions	√	√	√
	Incentives, Financing, & Funding	√	√	√
	Knowledge & Expertise	√	√	√
	Supportive Facilities & Mechanisms		√	√
Position/Group Relations	Interacting with Emerging Technologies		√	√
	Interaction with Commercial Software Development	√	√	√
	Market & Users	√	√	√

With the three case studies, we further compared the assessment results, to gain more insights

into how open source projects nowadays uphold OSS ideology in practice. Tab. 5.3 provides a summary of the themes we found evidence across three selected projects.

First, although the three selected open source projects were in different fields, i.e., data visualization, machine learning, and operating system, some of the themes were very similar or even the same across the three projects. For example, the theme *Universal Accessibility & Availability* was basically the same across three projects, i.e., they all provided code repositories for the public, and the source code was publicly available. Another example was *Copyright, Licensing, & Legal Implications*. Three selected open source projects were under different licenses, but those licenses were all open source licenses, and they basically granted similar rights to the public. Thus, those themes tended to be more fundamental for open source projects, and open source projects usually uphold them to a similar extent.

Second, some themes might be associated with each other, and potentially affect each other. In D3, we noticed that no newcomers joined the project since May 2021, which negatively affected *Constantly Changing & Evolving* of the community. Then, without a growing project community, there would be no need for *Member Hierarchy & Roles*. Furthermore, since no newcomers were joining the project, it was impossible to achieve the goal of *Sustainability* as well. However, with Zephyr and PyTorch, newcomers were joining the projects every month, and those themes within those two projects were completely different compared to D3. However, the case studies conducted were not sufficient to confirm or prove such associations among themes, and further investigation is essential to confirm these findings.

Third, the case studies indicated the heterogeneity in the perception and adoption of OSS ideology in the open source community. Different open source projects might uphold OSS ideology differently and to different extents, and there was no universal best practice for following OSS ideology. Take *Autonomy* as an example, Fig. 5.3 shows the commit activity within three projects, there were commits submitted to the three projects outside of the regular working hours, indicating autonomy existed across three projects. However, there

was a pattern in both Zephyr and PyTorch, in which the majority of the commits were submitted during regular working hours, and the pattern tended to be more significant in Zephyr; while such pattern was not explicit in D3. Those differences reflected the varying extents of *Autonomy* across the three projects.

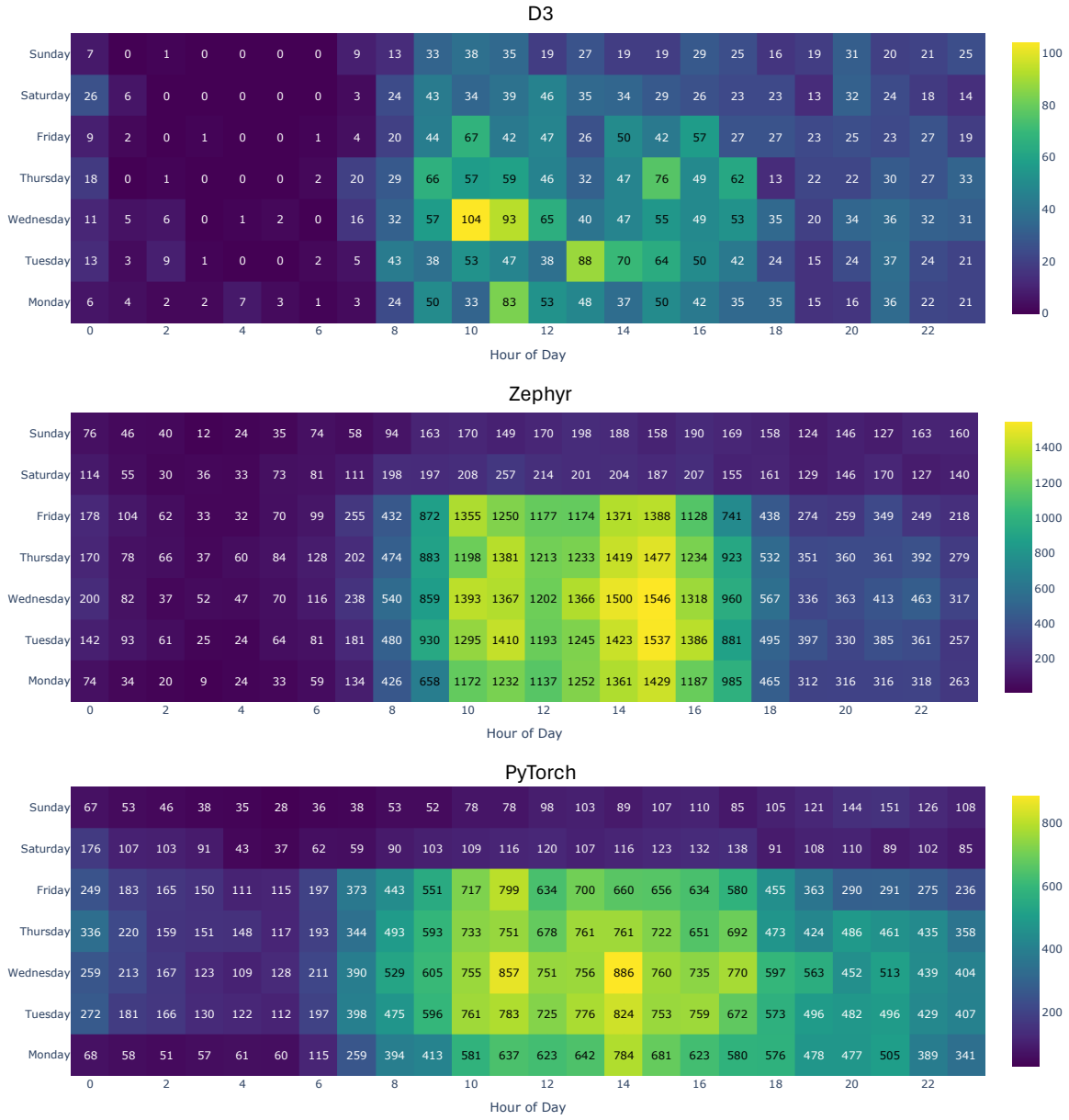

Figure 5.3: The commit activity heatmap of three projects.

5.4 Feedback from Stakeholders

The assessment results were sent to three recruited participants (**FP1-3**), and their feedback was collected through interviews. Note that the goal of collecting feedback was to investigate whether the assessment could provide valuable and helpful information for OSS practitioners and stakeholders, thus, it is not necessary to provide assessment results of all three projects to the recruited participants. We chose Zephyr in the feedback sessions since it is less well-known than PyTorch, helping to avoid potential bias, and it has more assessed themes than D3.

The recruited participants provided various feedback regarding the assessment of OSS ideologies. First, the participants were surprised that open source development involves various aspects, including both technical and social aspects within the community, and the assessment helped them be aware of them. For example, some participants shared:

> "*Open source development, and the community, could be studied from many perspectives, including technical stuff, social things... We cannot limit ourselves on a few perspectives.*" (**FP1**)

The assessment provided a thorough analysis of various aspects of the open source development. The assessment framework covered a comprehensive set of aspects of open source development, as our recruited participants mentioned:

> "*The framework is very comprehensive, it covers every aspect, like not only the developers, but also covers user aspect, interactions between developers, so it is very comprehensive from my understanding.*" (**FP2**)

Moreover, all three participants were not very familiar with the project, i.e., some of them only heard the name of the project. Our assessment results provided a comprehensive

overview of the project, and helped them learn more about the project. For example, one participant shared:

> "*I feel the project is very good, according to the report, only one theme was missed, so the project [basically] touched every aspect... It's good to know how the project is going on, from all perspectives.*" (**FP2**)

Furthermore, the information presented in the assessment results was informative, and could potentially help OSS practitioners make decisions, such as motivating them to join open source projects. OSS practitioners usually search for information regarding open source projects to help them make decisions, and our assessment could serve that purpose perfectly, as indicated by the participants:

> "*If I want to participate in open source projects, the more context and information I have from the community, the easier I can get involved, like their values, how they welcome the newcomers, etc. I think this report is basically a thorough analysis of the repository. Instead of me going through the repository, I can just walk through the report, and see the parts that I'm interested in.*" (**FP1**)

> "*I usually check the Github repository, to check everything, it is intuitive. If we have the [assessment] report, we can have everything in a single glance.*"
>
> "*Particularly to this project first, as the assessment results suggest, the project is very good, they maintain everything, in that sense, it motivates me to join their contributor list. It also motivates me to join open source development in general.*"(**FP2**)

> "*The [assessment] results make me curious to learn more about open source, maybe will try to commit one day to see how open source works.*" (**FP3**)

One participant also mentioned the potential of the assessment framework when applied to multiple open source projects:

> "*If I have several projects, and have their assessment results that they [might] not cover every aspect, then I can compare.*" (**FP2**)

Moreover, the participants also shared feedback regarding some detailed information they would like to have, and the format of the assessment results, to improve the accessibility to the provided information. For example, a high-level informative summary could attract audiences to read more detailed assessment results.

Therefore, the feedback from stakeholders indicated the significant values of the assessment framework and the assessment results, as well as the various benefits provided by such assessments:

- The assessment framework captures various aspects of open source development, and helps stakeholders be aware of multiple aspects.
- The assessment results provide a comprehensive overview of the open source projects, and help stakeholders better understand open source development.
- The information provided in the assessment plays a positive role in facilitating decision-making in open source development.

Chapter 6

Discussion & Implications

The empirical theory of OSS ideology, the assessment framework on OSS ideology, and the three case studies to investigate the OSS ideologies within open source projects not only provided significant insights into the open source movement, but also yielded rich implications and shed light on future research opportunities. We discuss both theoretical and practical implications, as well as limitations, in the remainder of this chapter.

6.1 Theoretical Implications

The empirical theory of OSS ideology, featuring *comprehensive*, *contemporary*, and *empirically-grounded* characteristics, provided an ideological lens to examine open source development. First, the theory reflected the ideologies across micro-, meso-, and macro-levels, i.e., individuals, projects, organizations, and society. It enabled us to understand the dynamics within open source movement across multiple levels with different granularity. Second, the theory consisted of 42 themes, organized under six broad categories. It covered various aspects of open source movements, and connected the fragmented knowledge from the literature under a

cohesive framework. It also established a common foundation to facilitate multi-disciplinary research endeavors in open source movement. Third, the assessment framework we developed was one implementation of the ideological lens provided by the theory, and it offered a more comprehensive investigation and in-depth understanding of the open source projects. Moreover, the open source movement, as an ongoing social movement, is still evolving. We identified various tensions and ongoing discussions within the open source community, which could be some potential directions that open source movement evolves. We could track those potential directions, keep the understanding of OSS ideology up-to-date, and gain insights into the evolution of the open source movement.

The case studies indicated the existence of the heterogeneity of OSS ideologies within open source projects, but further investigation would be necessary to have a more comprehensive understanding. First, different themes within the theory could influence particular dynamics and outcomes of open source development. Some of them tended to be straightforward to understand their influence, such as *Built-in Product & Quality Orientations*, while some have been explored in the extant literature, for example, *Individual Motivations* were one of the topics that were extensively explored [112, 11, 46, 94]. However, some themes, particularly newly identified themes, have not yet been studied comprehensively, such as the negative impacts of *Interaction with Commercial Software Development*, and *Innovation* in open source projects. Second, different themes might be associated with each other, i.e., the changes in one theme might affect another theme, even though under different categories. Investigating such associations could provide more insights into the dynamics of open source development from ideological perspectives, and potential could improve open source development.

Based on the research in this dissertation, we further list a few future directions toward a more comprehensive understanding of open source ideology:

- How would those themes within the OSS ideology affect the dynamics and outcomes

of the open source development?

- Which themes would associate with or affect other themes within the OSS ideology?
- How could the knowledge from other disciplines, such as organizational science, social science, and cognitive science, help gain more insights into open source development with such an ideological lens?
- What are the current tensions within the open source community, and how would those tensions influence the evolution of the open source movement?

6.2 Practical Implications

The empirical theory of OSS ideology could be a guideline and a checklist for open source practitioners to reflect on their practice in open source development, e.g., identifying potentially overlooked themes, resolving conflicts from the ideological perspective, attracting contributors who share a similar perception of OSS ideology, and so on. Based on the case studies, some themes tended to be fundamental across different open source projects, which could significantly distinguish open source development from other software development paradigms, such as fauxpen source [93]. Those fundamental themes could be the priorities for OSS practitioners when they identify open source projects to join, and examine their practice in open source development.

The assessment framework provided a practical tool to investigate the ideologies within particular open source projects. The project owners and maintainers could improve their practice according to the assessment results; open source contributors could find open source projects that match their perceptions of OSS ideology to join and contribute. The assessment framework could reduce the ideological misfit [26], and facilitate more efficient open source development. The feedback from OSS practitioners also indicated that the assessment could

provide a comprehensive and informative overview of open source projects, and potentially benefit their involvement in open source development.

We refined the assessment framework with the pilot case study, and many assessment workflows were improved based on the results. However, further refinement of the framework was still needed to provide a more accurate and comprehensive assessment of OSS ideology. First, there were ten themes that we have not yet operationalized and included in the assessment framework with the two design principles. Some methods might be explored and developed to assess those ten themes in the future, and the assessment framework could be a direct and complete mapping from the empirical theory of OSS ideology, which provides a comprehensive assessment of OSS ideology. Second, with new data and new methods introduced for the themes in the current assessment framework, we would be able to assess the themes utilizing multiple sources of evidence. Such improvement could significantly enhance the assessment results, making them more comprehensive and thorough. Third, the current design of the assessment framework includes automated or semi-automated assessment processes, by utilizing computer scripts and LLM. Automating the assessment could significantly improve the usability of the assessment framework, i.e., more OSS practitioners could conduct their assessments on the open source projects they are interested in, and make decisions accordingly. Moreover, the current assessment results were mostly descriptive, rather than quantitative. It would be more intuitive for OSS practitioners to quantify the results, making it easier to compare across different projects.

Therefore, there are rich research opportunities to further improve the practical aspects of both the empirical theory and assessment framework of OSS ideology:

- Providing detailed guidelines based on OSS ideology for OSS practitioners to examine the open source development.
- Further refining the workflows in the assessment framework to provide more accurate

and comprehensive assessment results.

- Automating the assessment framework to improve its usability for OSS practitioners.
- Exploring the possibility of quantifying the assessment results to facilitate cross-project comparison.

6.3 Limitations

Our studies in this dissertation are subjected to some limitations:

Regarding the empirical theory of OSS ideology, first, the concept of ideology is abstract and vague, sometimes ambiguous. We chose not to mention the term "ideology" during the interview, to avoid misleading the participants. However, they might not be able to explicitly express their perceptions of OSS ideology, and the collected data might not reflect their understanding of OSS ideology. Then, we compiled a diverse sample, including both grassroots OSS practitioners and OSI board members, to reflect the population of OSS community, but it was still impossible to claim that the sample was exhaustive enough to represent the whole OSS community. New concepts and themes might emerge when more sampled participants are in the study. Third, the empirical theory developed in our study only served as a contemporary understanding of OSS ideology, and it became "outdated" immediately after its establishment, like any other empirical theories related to social phenomena. With the evolution of OSS movement, further inquiries on OSS ideology are needed to keep the empirical understanding up-to-date.

Regarding the assessment framework and case studies on selected open source projects, the first limitation was the scope of the assessment framework only reflected OSS ideology at the project level. Since the OSS ideology is across multiple levels, we might expand the scope of the assessment framework in the future to reflect the multi-level nature of OSS ideology.

Second, we collected data from public data sources, some particular information might not be disclosed through those data sources, potentially limiting access to all data related to the open source projects. New findings and insights might emerge when collecting more related data within the selected projects. Third, we defined workflow for each assessed theme within the framework, we intended to use multiple sources of evidence, but for some themes, we were unable to develop other assessment methods with different data sources, which potentially became a threat to construct validity. It could be improved with new data sources and new assessment methods in the future refinement of the assessment framework. Fourth, the three selected open source projects were high-profile projects within the open source community, but it was still impossible for them to represent all open source projects, thus, the findings and results might not be generalized enough for other open source projects.

Chapter 7

Conclusions

Open source development is not only a software development paradigm that is widely adopted nowadays, but also a social movement that has profound impacts beyond technical realms. Ideology, as the "heart and soul", plays a critical role in motivating such social movements, and so does OSS ideology in the open source movement. Understanding open source ideology could help us gain more insights into open source development, which benefits both practitioners and researchers in open source development.

In this dissertation, we developed an empirical understanding of open source ideologies from both theoretical and practical perspectives. First, we developed an empirical theory of OSS ideology. Following grounded theory methodology, we recruited 22 OSS practitioners for semi-structured interviews to collect their narratives regarding open source development, and collected 41 videos of OSI board members' public interviews/speeches as a secondary data source. The collected data were analyzed through the coding process, and the empirical theory emerged from the collected. The empirical theory consisted of 42 themes under six broad categories, i.e., *Norms/Values*, *Membership*, *Goals*, *Activities*, *Resources*, and *Positions/Group Relations*. Such theory featured *comprehensive*, *contemporary*, and *empirically-*

grounded characteristics, reflecting our understanding of the open source movement from the ideological perspective.

Then, based on the emerging theory of OSS ideology, we developed a *practical* and *extensive* assessment framework to examine the ideologies within open source projects. The assessment framework utilizes data from public sources, i.e., project repository, documentation, and news articles, and provides assessments on OSS ideology at the project level. Moreover, we conducted three case studies by applying the assessment framework to three selected open source projects. The results demonstrated how open source projects uphold OSS ideology in practice, i.e., they indicated the existence of the heterogeneity of OSS ideology in open source projects, and its influences on the different dynamics and outcomes in open source development.

The research in this dissertation set a foundation to understand and investigate open source development from an ideological perspective. The empirical theory of OSS ideology reflected the collective perception within the whole open source community. It connected the fragmented knowledge, and provided an ideological lens for researchers to examine open source development, even facilitating interdisciplinary research inquiries. The assessment framework provided a practical way to utilize such an ideological lens for both OSS researchers and practitioners, i.e., OSS researchers would gain in-depth insights from an ideological perspective; OSS practitioners would find open source projects that match their perception, and improve the practice of open source development with the guidance of OSS ideology. Furthermore, we discussed a few future directions to continue the exploration toward the empirical understanding of open source ideology and open source movement.

Bibliography

[1] G. Abramo, C. A. D'Angelo, and F. Di Costa. The effect of multidisciplinary collaborations on research diversification. *Scientometrics*, 116:423–433, 2018.

[2] M. Alvesson and H. Willmott. Identity regulation as organizational control: Producing the appropriate individual. *Journal of management studies*, 39(5):619–644, 2002.

[3] B. E. Ashforth, S. H. Harrison, and K. G. Corley. Identification in organizations: An examination of four fundamental questions. *Journal of Management*, 34(3):325–374, 2008.

[4] R. P. Bagozzi and U. M. Dholakia. Open source software user communities: A study of participation in linux user groups. *Management Science*, 52(7):1099–1115, 2006.

[5] S. Ballmer. Ballmer: 'linux is a cancer'. *The Register*, June 2001. Accessed: 2024-04-17.

[6] H. Baytiyeh and J. Pfaffman. Open source software: A community of altruists. *Computers in Human Behavior*, 26(6):1345–1354, 2010.

[7] S. Beck, C. Bergenholtz, M. Bogers, T.-M. Brasseur, M. L. Conradsen, D. Di Marco, A. P. Distel, L. Dobusch, D. Dörler, A. Effert, et al. The open innovation in science research field: a collaborative conceptualisation approach. *Industry and Innovation*, 29(2):136–185, 2022.

[8] R. Bendix. *Work and Authority in Industry: Ideologies of Management in the Course of Industrialization*. Harper & Row, 1956.

[9] Y. S. Bermiss and R. McDonald. Ideological misfit? political affiliation and employee departure in the private-equity industry. *Academy of Management Journal*, 61(6):2182–2209, 2018.

[10] M. Bianchin. Explaining ideology: Mechanisms and metaphysics. *Philosophy of the Social Sciences*, 50(4):313–337, 2020.

[11] J. Bitzer, W. Schrettl, and P. J. Schröder. Intrinsic motivation in open source software development. *Journal of Comparative Economics*, 35(1):160–169, 2007.

[12] A. Bosu and K. Z. Sultana. Diversity and inclusion in open source software (OSS) projects: Where do we stand? In *2019 ACM/IEEE International Symposium on*

Empirical Software Engineering and Measurement, ESEM 2019, Porto de Galinhas, Recife, Brazil, September 19-20, 2019, pages 1–11. IEEE, 2019.

[13] P. Bourdieu and L. Boltanski. La production de l'idéologie dominante. *Actes de la Recherche en Sciences Sociales*, 2(2):3–73, 1976.

[14] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli. Sentiment polarity detection for software development. *Empir. Softw. Eng.*, 23(3):1352–1382, 2018.

[15] E. D. Canedo, R. Bonifácio, M. V. Okimoto, A. Serebrenik, G. Pinto, and E. Monteiro. Work practices and perceptions from women core developers in OSS communities. In M. T. Baldassarre, F. Lanubile, M. Kalinowski, and F. Sarro, editors, *ESEM '20: ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Bari, Italy, October 5-7, 2020*, pages 26:1–26:11. ACM, 2020.

[16] M. Castells. *The Internet galaxy: Reflections on the Internet, business, and society.* Oxford University Press, USA, 2002.

[17] K. Charmaz. *Constructing Grounded Theory: A Practical Guide through Qualitative Analysis.* SAGE, 2006.

[18] S. Chen. The relation between ideology and decision-making. *The Journal of Global Business Management*, 2(3):140–50, 2006.

[19] I. Chengalur-Smith, A. Sidorova, and S. L. Daniel. Sustainability of free/libre open source projects: A longitudinal study. *Journal of the Association for Information Systems*, 11(11):657–683, 2010.

[20] J. Coelho and M. T. Valente. Why modern open source projects fail. In E. Bodden, W. Schäfer, A. van Deursen, and A. Zisman, editors, *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*, pages 186–196. ACM, 2017.

[21] J. A. Colazo and Y. Fang. Impact of license choice on open source software development activity. *J. Assoc. Inf. Sci. Technol.*, 60(5):997–1011, 2009.

[22] J. Corbin and A. Strauss. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory.* SAGE, 2008.

[23] K. Crowston and B. Scozzi. Open source software projects as virtual organisations: competency rallying for software development. *IEE Proc. Softw.*, 149(1):3–17, 2002.

[24] 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.*, 44(2):7:1–7:35, 2012.

[25] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In S. E. Poltrock, C. Simone, J. Grudin, G. Mark, and J. Riedl, editors, *CSCW '12 Computer Supported Cooperative Work, Seattle, WA, USA, February 11-15, 2012*, pages 1277–1286. ACM, 2012.

[26] S. L. Daniel, L. M. Maruping, M. Cataldo, and J. D. Herbsleb. The impact of ideology misfit on open source software communities and companies. *MIS Quarterly*, 42(4):1069–1096, 2018.

[27] A. Davies, S. Manning, and J. Söderlund. When neighboring disciplines fail to learn from each other: The case of innovation and project management research. *Research Policy*, 47(5):965–979, 2018.

[28] F. P. Deek and J. A. McHugh. *Open source: Technology and Policy*. Cambridge University Press, 2007.

[29] M. Diani. The concept of social movement. *The Sociological Review*, 40(1):1–25, 1992.

[30] M. Diani. Social movement networks virtual and real. *Information, Communication & Society*, 3(3):386–401, 2000.

[31] P. L. Elkin. *Terminology and Terminological Systems*. Springer Science & Business Media, 2012.

[32] G. L. Evans. A novice researcher's first walk through the maze of grounded theory: Rationalization for classical grounded theory. *Grounded Theory Review*, 12(1), 2013.

[33] H. Fang, D. Klug, H. Lamba, J. D. Herbsleb, and B. Vasilescu. Need for tweet: How open source developers talk about their github work on twitter. In S. Kim, G. Gousios, S. Nadi, and J. Hejderup, editors, *MSR '20: 17th International Conference on Mining Software Repositories, Seoul, Republic of Korea, 29-30 June, 2020*, pages 322–326. ACM, 2020.

[34] H. Fang, H. Lamba, J. D. Herbsleb, and B. Vasilescu. "this is damn slick!" estimating the impact of tweets on open source project popularity and new contributors. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pages 2116–2129. ACM, 2022.

[35] M. Feldman. Organizational routines as a source of continuous change. *Organization Science*, 11(6):611–629, 2000.

[36] M. Feldman and B. Pentland. Reconceptualizing organizational routines as a source of flexibility and change. *Administrative Science Quarterly*, 48(1):94–118, 2003.

[37] J. Feller and B. Fitzgerald. A framework analysis of the open source software development paradigm. In S. Ang, H. Krcmar, W. J. Orlikowski, P. Weill, and J. I. DeGross, editors, *Proceedings of the Twenty-First International Conference on Information Systems, ICIS 2000, Brisbane, Australia, December 10-13, 2000*, pages 58–69. Association for Information Systems, 2000.

[38] J. Feller, B. Fitzgerald, W. Scacchi, and A. Sillitti. *Open Source Development, Adoption and Innovation: IFIP Working Group 2.13 on Open Source Software*. Springer, 2007.

[39] I. Ferreira, J. Cheng, and B. Adams. The "shut the f**k up" phenomenon: Characterizing incivility in open source code review discussions. *Proc. ACM Hum. Comput. Interact.*, 5(CSCW2):1–35, 2021.

[40] R. T. Fielding. Shared leadership in the apache project. *Commun. ACM*, 42(4):42–43, 1999.

[41] B. Fitzgerald. The transformation of open source software. *MIS Quarterly*, 30(3):587–598, 2006.

[42] D. Ford, M. Behroozi, A. Serebrenik, and C. Parnin. Beyond the code itself: how programmers *really* look at pull requests. In R. Kazman and L. Pasquale, editors, *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Society, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, pages 51–60. ACM, 2019.

[43] M. Foucault. The history of sexuality: An introduction, volume i. *Trans. Robert Hurley. New York: Vintage*, 95, 1990.

[44] 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*, 89:128–145, 2014.

[45] R. S. Geiger, D. Howard, and L. Irani. The labor of maintaining and scaling free and open-source software projects. *Proc. ACM Hum. Comput. Interact.*, 5(CSCW1):1–28, 2021.

[46] M. A. Gerosa, I. Wiese, B. Trinkenreich, G. Link, G. Robles, C. Treude, I. Steinmacher, and A. Sarma. The shifting sands of motivation: Revisiting what drives contributors in open source. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*, pages 1046–1058. IEEE, 2021.

[47] J. L. Gibbs, N. A. Rozaidi, and J. Eisenberg. Overcoming the "ideology of openness": Probing the affordances of social media for organizational knowledge sharing. *Journal of Computer-Mediated Communication*, 19(1):102–120, 2013.

[48] G. Gousios, M. D. Storey, and A. Bacchelli. Work practices and challenges in pull-based development: the contributor's perspective. In L. K. Dillon, W. Visser, and L. A. Williams, editors, *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, pages 285–296. ACM, 2016.

[49] B. D. Haig. Grounded theory as scientific method. *Philosophy of Education*, 28(1):1–11, 1995.

[50] Y. Han, Z. Wang, Y. Feng, Z. Zhao, and Y. Wang. Cross-status communication and project outcomes in oss development–a language style matching perspective. *Empirical Software Engineering*, 28(3):78, 2023.

[51] J. F. Hartley. Ideology and organizational behavior. *International Studies of Management & Organization*, 13(3):7–34, 1983.

[52] J. Hergueux and S. Kessler. Follow the leader: Technical and inspirational leadership in open source software. In S. D. J. Barbosa, C. Lampe, C. Appert, D. A. Shamma, S. M. Drucker, J. R. Williamson, and K. Yatani, editors, *CHI '22: CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April 2022 - 5 May 2022*, pages 303:1–303:15. ACM, 2022.

[53] C. E. Hill, S. Knox, B. J. Thompson, E. N. Williams, S. A. Hess, and N. Ladany. Consensual qualitative research: An update. *Journal of counseling psychology*, 52(2):196, 2005.

[54] M. Hinchey, B. Fitzgerald, F. Krehbiel, B. Donnellan, and T. Margaria. *Innovation Potential of Software Technologies in the context of Horizon 2020*. European Commission, 2016.

[55] C. Hinings, L. Thibault, T. Slack, and L. Kikulis. Values and organizational structure. *Human relations*, 49(7):885–916, 1996.

[56] S. E. Hobfoll, J. Halbesleben, J.-P. Neveu, and M. Westman. Conservation of resources in the organizational context: The reality of resources and their consequences. *Annual Review of Organizational Psychology and Organizational Behavior*, 5:103–128, 2018.

[57] E. K. Huizingh. Open innovation: State of the art and future perspectives. *Technovation*, 31(1):2–9, 2011.

[58] Z. Iskoujina and J. Roberts. Knowledge sharing in open source software communities: motivations and management. *J. Knowl. Manag.*, 19(4):791–813, 2015.

[59] C. Jensen and W. Scacchi. Collaboration, leadership, control, and conflict negotiation and the netbeans.org open source software development community. In *38th Hawaii International Conference on System Sciences (HICSS-38 2005), CD-ROM / Abstracts Proceedings, 3-6 January 2005, Big Island, HI, USA*. IEEE Computer Society, 2005.

[60] C. Jensen and W. Scacchi. Role migration and advancement processes in OSSD projects: A comparative case study. In *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007*, pages 364–374. IEEE Computer Society, 2007.

[61] C. Jergensen, A. Sarma, and P. Wagstrom. The onion patch: migration in open source ecosystems. In T. Gyimóthy and A. Zeller, editors, *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13th European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9, 2011*, pages 70–80. ACM, 2011.

[62] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38, 2023.

[63] B. Kabanoff. Equity, equality, power, and conflict. *Academy of management Review*, 16(2):416–441, 1991.

[64] M. Kejriwal, J. Ding, R. Shao, A. Kumar, and P. Szekely. Flagit: A system for minimally supervised human trafficking indicator mining. *Advances in Neural Information Processing Systems*, 2017.

[65] E. Kennedy and T. Marshall. A Philosophe in The Age of Revolution : Destutt de Tracy and The Origins of 'Ideology'. *Les Études philosophiques*, (4):455–459, dec 1982.

[66] L. King. Ideology, strategy and conflict in a social movement organization: The sierra club immigration wars. *Mobilization: An International Quarterly*, 13(1):45–61, 2008.

[67] D. Klug, C. Bogart, and J. D. Herbsleb. "they can only ever guide": How an open source software community uses roadmaps to coordinate effort. *Proc. ACM Hum. Comput. Interact.*, 5(CSCW1):1–28, 2021.

[68] R. Krishnaveni and R. Sujatha. Communities of practice: An influencing factor for effective knowledge transfer in organizations. *IUP Journal of Knowledge Management*, 10(1):26–40, 01 2012.

[69] J. Lerner and J. Tirole. Some simple economics of open source. *The Journal of Industrial Economics*, 50(2):197–234, 2002.

[70] J. Leveille. *Searching for Marx in the Occupy Movement*. Rowan & Littlefield, 2017.

[71] S. S. Levine and M. J. Prietula. Open collaboration for innovation: Principles and performance. *Organization Science*, 25(5):1414–1433, 2014.

[72] R. Li, P. Pandurangan, H. Frluckaj, and L. Dabbish. Code of conduct conversations in open source software projects on github. *Proc. ACM Hum. Comput. Interact.*, 5(CSCW1):1–31, 2021.

[73] J. Ljungberg. Open source movements as a model for organising. *Eur. J. Inf. Syst.*, 9(4):208–216, 2000.

[74] M. Maclean, C. Harvey, J. A. Sillince, and B. D. Golant. Living up to the past? ideological sensemaking in organizational transition. *Organization*, 21(4):543–567, 2014.

[75] T. W. Malone and M. Klein. Harnessing collective intelligence to address global climate change. *Innovations: Technology, Governance, Globalization*, 2(3):15–26, 2007.

[76] C. Marx and F. Engels. *The German Ideology*. 1845.

[77] A. M. McCright and R. E. Dunlap. The nature and social bases of progressive social movement ideology: Examining public opinion toward social movements. *Sociological Quarterly*, 49(4):825–848, 2008.

[78] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3):309–346, 2002.

[79] J. Y. Moon and L. S. Sproull. Essence of distributed work: The case of the linux kernel. *First Monday*, 5(11), 2000.

[80] M. J. Muller and S. Kogan. Grounded theory method in hci and cscw. *Cambridge: IBM Center for Social Software*, 28(2):1–46, 2010.

[81] M. D. Mumford. *Pathways to Outstanding Leadership: A Comparative Analysis of Charismatic, Ideological, and Pragmatic Leaders*. Lawrence Erlbaum Associates Publishers, 2006.

[82] F. N. A. A. Omran and C. Treude. Choosing an NLP library for analyzing software documentation: a systematic literature review and a series of experiments. In J. M. González-Barahona, A. Hindle, and L. Tan, editors, *Proceedings of the 14th International Conference on Mining Software Repositories, MSR 2017, Buenos Aires, Argentina, May 20-28, 2017*, pages 187–197. IEEE Computer Society.

[83] OpenAI. Prompt engineering guide. `https://platform.openai.com/docs/guides/prompt-engineering`, 2023. Accessed: 2023-04-10.

[84] C. Overney, J. Meinicke, C. Kästner, and B. Vasilescu. How to not get rich: an empirical study of donations in open source. In G. Rothermel and D. Bae, editors, *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pages 1209–1221. ACM, 2020.

[85] H. Perera, W. Hussain, J. Whittle, A. Nurwidyantoro, D. Mougouei, R. A. Shams, and G. Oliver. A study on the prevalence of human values in software engineering publications, 2015 - 2018. In G. Rothermel and D. Bae, editors, *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pages 409–420. ACM, 2020.

[86] PyTorch. Governance model of pytorch. `https://github.com/pytorch/pytorch/blob/main/docs/source/community/governance.rst`, 2023. Accessed: 2023-04-10.

[87] H. S. Qiu, B. Vasilescu, C. Kästner, C. Egelman, C. Jaspan, and E. Murphy-Hill. Detecting interpersonal conflict in issues and code review: Cross pollinating open- and closed-source approaches. In *44th IEEE/ACM International Conference on Software Engineering: Software Engineering in Society ICSE (SEIS) 2022, Pittsburgh, PA, USA, May 22-24, 2022*, pages 41–55. IEEE/ACM, 2022.

[88] C. Raasch, V. Lee, S. Spaeth, and C. Herstatt. The rise and fall of interdisciplinary research: The case of open source innovation. *Research Policy*, 42(5):1138–1151, 2013.

[89] A. K. Rai. " open source" and private ordering: A commentary on dusollier. *Chicago-Kent Law Review*, 82(3):1439, 2007.

[90] P. Railton et al. *Facts, values, and norms: Essays toward a morality of consequence*. Cambridge University Press, 2003.

[91] E. S. Raymond. The cathedral and the bazaar. *First Monday*, 3(3), 1998.

[92] G. W. Remmling. *The Sociology of Karl Mannheim: With a Bibliographical Guide to the Sociology of Knowledge, Ideological Analysis, and Social Planning*. Routledge, 2020.

[93] D. Riehle. The future of the open source definition. *Computer*, 56(12):95–99, 2023.

[94] J. A. Roberts, I. Hann, and S. Slaughter. Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects. *Manag. Sci.*, 52(7):984–999, 2006.

[95] S. Sarker, F. Y. Lau, and S. Sahay. Using an adapted grounded theory approach for inductive theory building about virtual team development. *Data Base*, 32(1):38–56, 2001.

[96] Z. Schillaci. Llm adoption trends and associated risks. *Large*, page 121, 2024.

[97] S. H. Schwartz et al. An overview of the schwartz theory of basic values. *Online Readings in Psychology and Culture*, 2, 2012.

[98] H. Sharp, N. Baddoo, S. Beecham, T. Hall, and H. Robinson. Models of motivation in software engineering. *Information and Software Technology*, 51(1):219–233, 2009.

[99] R. M. Stallman. Viewpoint - why "open source" misses the point of free software. *Commun. ACM*, 52(6):31–33, 2009.

[100] I. Steinmacher, S. Balali, B. Trinkenreich, M. Guizani, D. Izquierdo-Cortazar, G. G. C. Zambrano, M. A. Gerosa, and A. Sarma. Being a mentor in open source projects. *J. Internet Serv. Appl.*, 12(1):7, 2021.

[101] I. Steinmacher, M. A. Gerosa, T. U. Conte, and D. F. Redmiles. Overcoming social barriers when contributing to open source software projects. *Comput. Support. Cooperative Work.*, 28(1-2):247–290, 2019.

[102] K. J. Stewart and S. Gosain. The impact of ideology on effectiveness in open source software development teams. *MIS Quarterly*, 30(2):291–314, 2006.

[103] X. Tan, M. Zhou, and B. Fitzgerald. Scaling open source communities: an empirical study of the linux kernel. In G. Rothermel and D. Bae, editors, *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pages 1222–1234. ACM, 2020.

[104] J. B. Thompson. *Ideology and Modern Culture*. Polity Press, 1990.

[105] W. E. Thompson, J. V. Hickey, and M. L. Thompson. *Society in Focus: An Introduction to Sociology*. Rowman & Littlefield, 2016.

[106] B. Trinkenreich, M. Guizani, I. Wiese, A. Sarma, and I. Steinmacher. Hidden figures: Roles and pathways of successful OSS contributors. *Proc. ACM Hum. Comput. Interact.*, 4(CSCW2):180:1–180:22, 2020.

[107] J. Tsay, L. Dabbish, and J. D. Herbsleb. Influence of social and technical factors for evaluating contribution in github. In *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, pages 356–366. ACM, 2014.

[108] S. Valocchi. The emergence of the integrationist ideology in the civil rights movement. *Social Problems*, 43(1):116–130, 1996.

[109] T. A. van Dijk. *Ideology: A multidisciplinary Approach*. SAGE, 1998.

[110] J. Van Laer and P. Van Aelst. Internet and social movement action repertoires: Opportunities and limitations. *Information, Communication & Society*, 13(8):1146–1171, 2010.

[111] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov. Gender and tenure diversity in github teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*, pages 3789–3798. ACM, 2015.

[112] G. von Krogh, S. Haefliger, S. Spaeth, and M. W. Wallin. Carrots and rainbows: Motivation and social practice in open source software development. *MIS Quarterly*, 36(2):649–676, 2012.

[113] P. A. Wagstrom. *Vertical Interaction in Open Software Engineering Communities*. PhD thesis, Pittsburgh, PA, 2009.

[114] T. Warren. Microsoft admits it was wrong about open source. *The Verge*, May 2020. Accessed: 2024-04-17.

[115] R. M. Weiss and L. E. Miller. The concept of ideology in organizational analysis: The sociology of knowledge or the social psychology of beliefs? *Academy of Management Review*, 12(1):104–116, 1987.

[116] R. K. Yin. *Case Study Research: Design and Methods*. SAGE, 2014.

[117] Y. Yue, X. Yu, X. You, Y. Wang, and D. F. Redmiles. Ideology in open source development. In *14th IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE@ICSE 2021, Madrid, Spain, May 20-21, 2021*, pages 71–80. IEEE, 2021.

[118] A. Zagalsky, D. M. Germán, M. D. Storey, C. G. Teshima, and G. Poo-Caamaño. How the R community creates and curates knowledge: an extended study of stack overflow and mailing lists. *Empirical Software Engineering*, 23(2):953–986, 2018.

[119] M. Zald. Ideologically structured action: An enlarged agenda for social movement research. *Mobilization: An International Quarterly*, 5(1):1–16, 2000.

[120] M. Zastrow. Open science takes on the coronavirus pandemic. *Nature*, 581(7806):109–111, 2020.

[121] Zephyr Project. Licensing guidelines. `https://docs.zephyrproject.org/latest/contribute/guidelines.html#licensing`, 2023. Accessed: 2023-04-11.

[122] X. Zhang, T. Wang, Y. Yu, Q. Zeng, Z. Li, and H. Wang. Who, what, why and how? towards the monetary incentive in crowd collaboration: A case study of github's sponsor mechanism. In S. D. J. Barbosa, C. Lampe, C. Appert, D. A. Shamma, S. M. Drucker, J. R. Williamson, and K. Yatani, editors, *CHI '22: CHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 29 April 2022 - 5 May 2022*, pages 206:1–206:18. ACM, 2022.

[123] Y. Zhang, H. Liu, X. Tan, M. Zhou, Z. Jin, and J. Zhu. Turnover of companies in openstack: Prevalence and rationale. *ACM Trans. Softw. Eng. Methodol.*, 31(4):75:1–75:24, 2022.

[124] Y. Zhang, K. Stol, H. Liu, and M. Zhou. Corporate dominance in open source ecosystems: a case study of openstack. In A. Roychoudhury, C. Cadar, and M. Kim, editors, *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pages 1048–1060. ACM, 2022.

[125] Y. Zhang, M. Zhou, K.-J. Stol, J. Wu, and Z. Jin. How do companies collaborate in open source ecosystems?: an empirical study of openstack. In G. Rothermel and D. Bae, editors, *ICSE '20: 42nd International Conference on Software Engineering, Seoul, South Korea, 27 June - 19 July, 2020*, pages 1196–1208. ACM, 2020.

[126] J. Zhou, S. Wang, Y. Kamei, A. E. Hassan, and N. Ubayashi. Studying donations and their expenses in open source projects: a case study of github projects collecting donations through open collectives. *Empirical Software Engineering*, 27(1):24:1–24:38, 2022.

[127] M. Zhou. *Onboarding and Retaining of Contributors in FLOSS Ecosystem*, pages 107–117. Springer Singapore, Singapore, 2019.

[128] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Appendix A

Interview Protocol

Interview Subject: developers who participate in open source projects.

Length of Interview: 30 minutes.

Interview Strategy:

- Subject has a brief introduction about background, e.g., country, job/occupation, years of experience in software development and open source projects, etc.
- Based on the background information, ask subject to share his/her experience in contributing to open source projects.
- Ask subject follow-up questions to elaborate more details.

A.1 Interview Script

A.1.1 Briefing and Verbal Consent

Hello, I'm Yang Yue from the University of California, Irvine, Prof. David Redmiles is the researcher in charge of this study. We appreciate your voluntary participation in our study. We are interested in hearing about your thoughts and experiences in open source development.

As a reminder, the interview will be recorded for internal use only, and we will take notes during the interview. Your identity will remain anonymous in our study, other than your email address, no other personal information will be collected. Participating in this study is optional, and you cal tell me if you want to stop being in the study at any time.

Before we start, do you have any questions about the study?

(If no further questions from the subject)

Great, I'll start the interview recording now.

A.1.2 Interview Questions

(The questions listed in this section are potential questions that might be asked during interview, interviewer might also ask some other questions based on subject's answer.)

Background Information

- Which country do you live in?
- What is your current occupation/job title?

- How many years of experience in software development?
- How long have you been participating in open source projects?

Perceptions/Experiences about Open Source

- What is your understanding of open source?
- When and where did you first learn the idea of open source? Could you explain what it means to you?
- What makes you start contributing in open source projects?
- What is the process of contributing to open source projects?
- Who do you think could contribute to open source projects?

Debriefing

Thank you again for participating in our study. If you have any questions or think of any other comments that you forgot to mention, please feel free to email me. And we might contact later for more information.

Appendix B

Assessment Framework for OSS Ideologies

In this section, we introduce the workflows for assessing 32 themes in the assessment framework for OSS ideologies.

B.1 Category: Values/Norms

B.1.1 Appreciation & Recognition

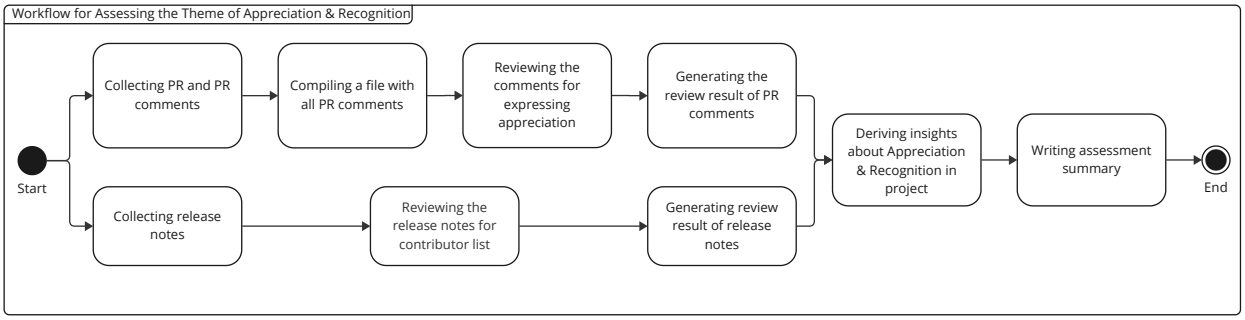

Figure B.1: The workflow for assessing Appreciation & Recognition.

Fig. B.1 shows the workflow for assessing *Appreciation & Recognition*. Two assessment methods are included in this workflow. First, we collect all the pull-requests and their comments, and compile all the PR comments into a file. Then we review the PR comments, particularly the comments that express appreciation towards the contributors, e.g., "Thank you for submitting your PR!" Based on those PR comments, we generate the review results, such as whether it is standard to express appreciation in PR comments, and the frequency of such comments. The other assessment method focuses on release notes, we collect all release notes, and review the content related to recognizing the contributors. For example, some projects would list all contributors who have contributed to the releases. Then, we generate the review results of release notes, such as whether there is a contributor list, and the frequency of having such a list in the release notes. With the review results of two assessment methods, we derive the insights about *Appreciation & Recognition*, and write the assessment summary.

B.1.2 Autonomy

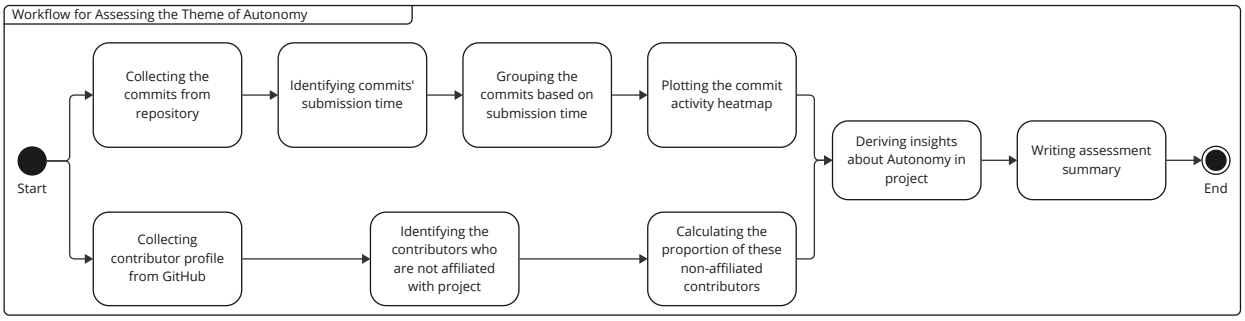

Figure B.2: The workflow for assessing Autonomy

Fig. B.2 shows the workflow for assessing *Autonomy*. Two assessment methods are included, i.e., one focuses on commit activity, while the other focuses on contributors' affiliation. First, we collect commit data from the project repository, and extract the submission time of the commits. Then we group the commit based on submission time, i.e., time of the day, and

week of the day, and plot the data in a commit activity heatmap. The heatmap shows the number of commits submitted on particular days and times, which indicates the distribution of the commits submitted within the regular working time and non-working time. With the other assessment method, we collect contributors from the project repository on GitHub. Based on the collected contributor data, we further identify contributors' affiliation, i.e., whether the contributors are listed as project members or not, and then provide the results of affiliated and non-affiliated contributors. Based on the results of the two assessment methods, we derive the insights about *Autonomy*, and write the assessment summary.

B.1.3 Comparative Advantages

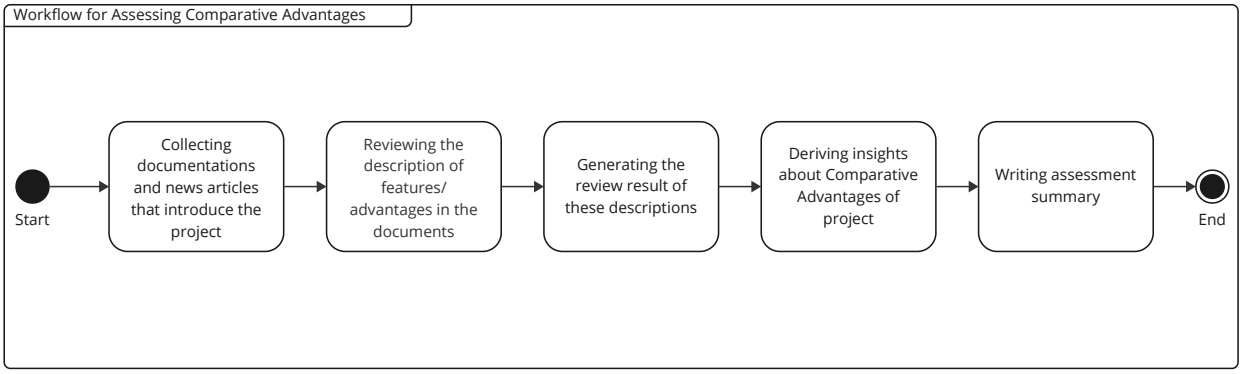

Figure B.3: The workflow for assessing Comparative Advantages

Fig. B.3 shows the workflow for assessing *Comparative Advantages*. First, we collect documents and online news articles that introduce the project. Then we review the content that describes the features or advantages of the project, and generate the review results. Based on the review results, we derive the insights about *Comparative Advantages*, and write the assessment summary.

B.1.4 Constantly Changing & Evolving

Fig. B.4 shows the workflow for assessing *Constantly Changing & Evolving*. There are two assessment methods with the collected commit data from the project repository. First, we extract the contributor of each commit, and identify whether the commit was the first commit of this contributor, which determines whether the contributor was a newcomer at that commit. Then we compile the list of newcomers, and plot the dynamics of newcomers in each month. The other assessment method also analyzes the commit data. We extract the submission date of the commits, and group the commits into months. Then we plot the dynamics of commit frequency in each month. Based on the results of two assessment methods, we derive insights about *Constantly Changing & Evolving*, and write the assessment summary.

B.1.5 Ethics

Fig. B.5 shows the workflow for assessing *Ethics*. Two assessment methods are included. With the first assessment method, we collect documents about the project community, and review the content about the community guidelines, such as ethical values emphasized in the community. Then we identify those values and the corresponding enforcement process from the documents, and generate the review results. The second assessment method utilizes online media outlets. We search online for the news about the project, particularly public scandals within the project. Then we review those news articles, and generate review results. Based on the results of two assessment methods, we derive insights about *Ethics*, and write the assessment summary.

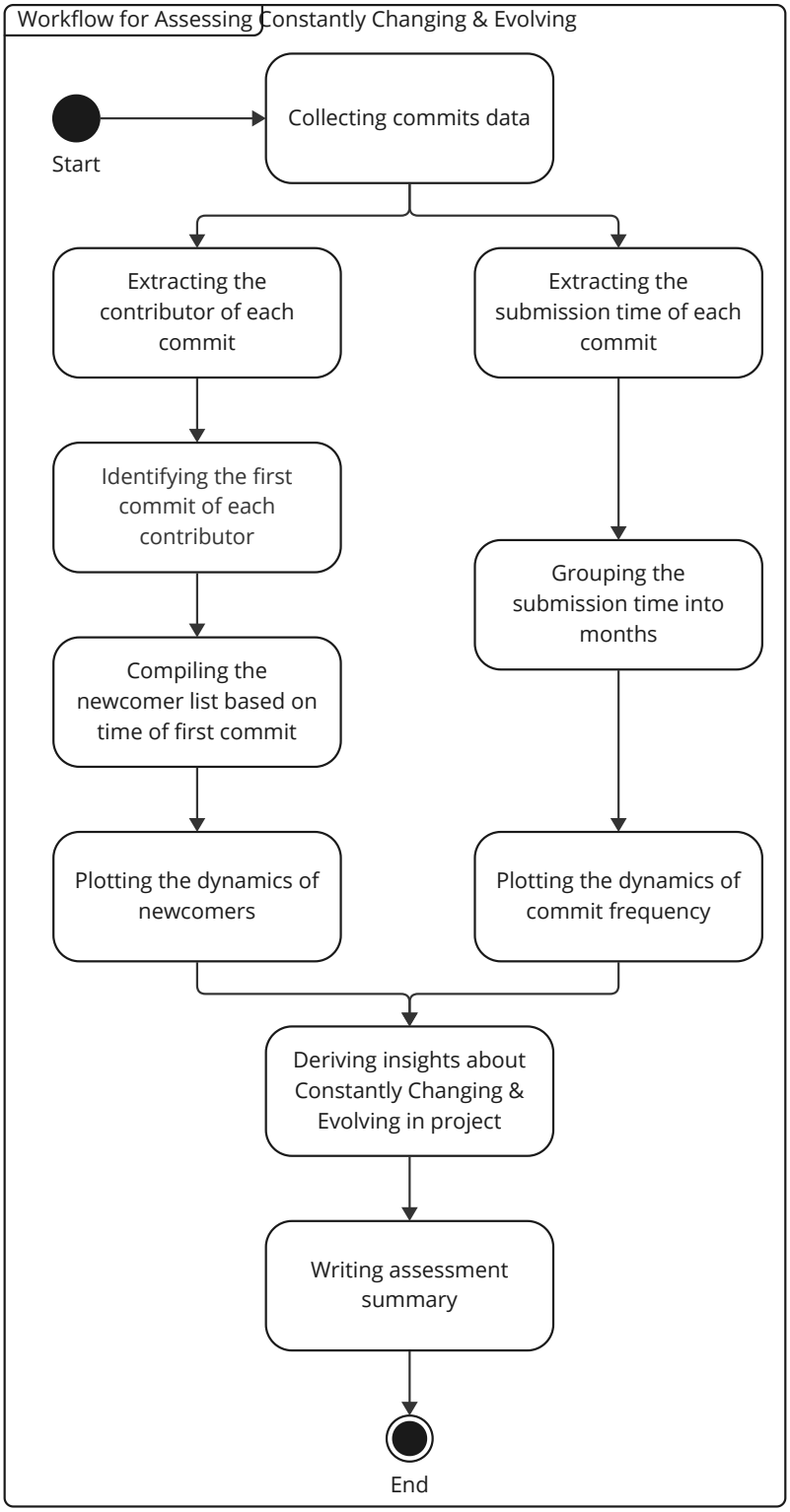

Figure B.4: The workflow for assessing Constantly Changing & Evolving

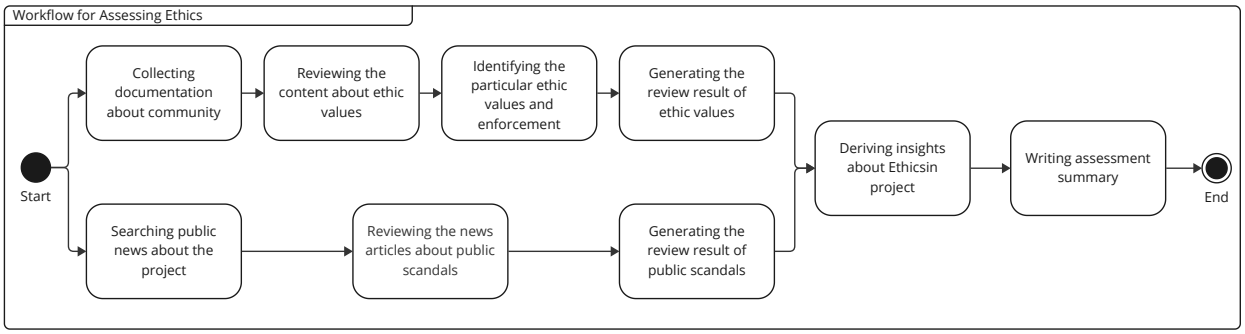

Figure B.5: The workflow for assessing Ethics

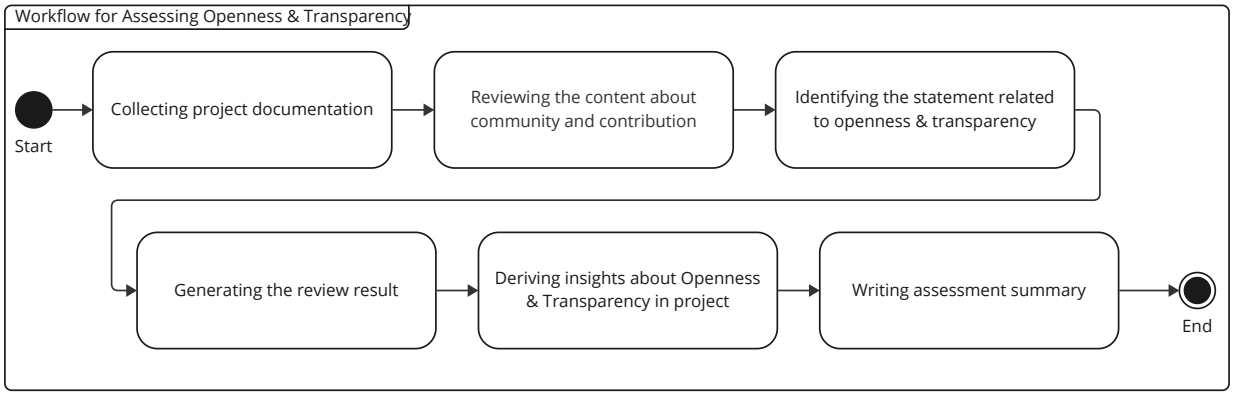

Figure B.6: The workflow for assessing Openness & Transparency

B.1.6 Openness & Transparency

Fig. B.6 shows the workflow for assessing *Openness & Transparency*. First, we collect the documents, and review the content about community and contribution guidelines. Then we identify the statements related to openness or transparency in the collected documents, and generate the review results. Based on the results, we derive the insights about *Openness & Transparency*, and write the assessment summary.

B.1.7 Meritocracy

Fig. B.7 shows the workflow for assessing *Meritocracy*. We first identify whether there are different roles in the project, such as maintainers, and contributors. If such roles exist,

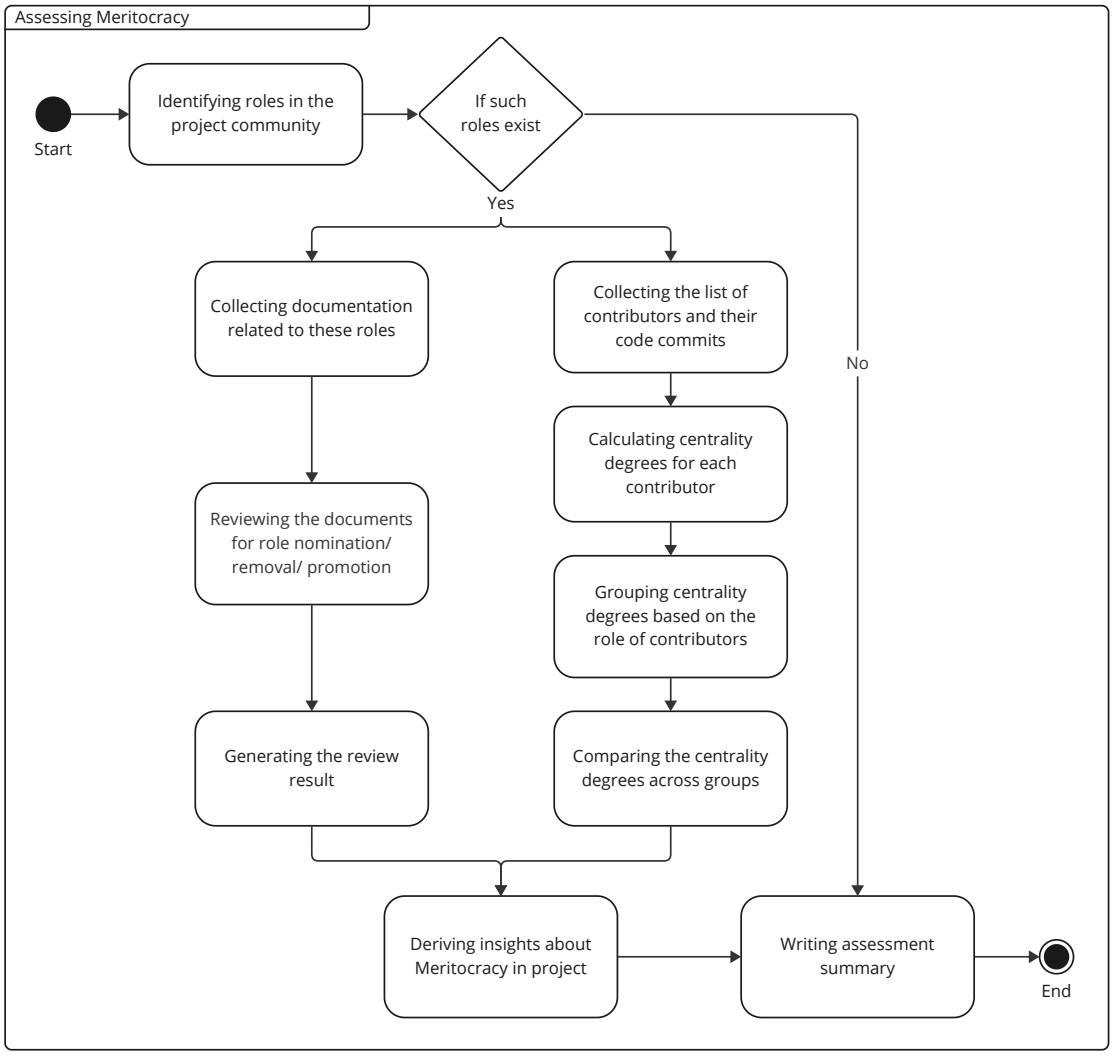

Figure B.7: The workflow for assessing Meritocracy.

we assess with two methods. We collect the documents related to those roles, review the content regarding role nomination/removal/promotion, and generate the review results. We also collect the list of contributors and their code commits from project repository. Then we calculate the centrality degree of each contributor[1], and group them based on the roles. We further compare the difference of the centrality degrees across different role groups. Based on the results of two assessment methods, we derive insights about *Meritocracy*, and write the assessment summary.

[1]see [61] for more details on centrality degree

B.1.8 Reciprocity at Multiple Levels

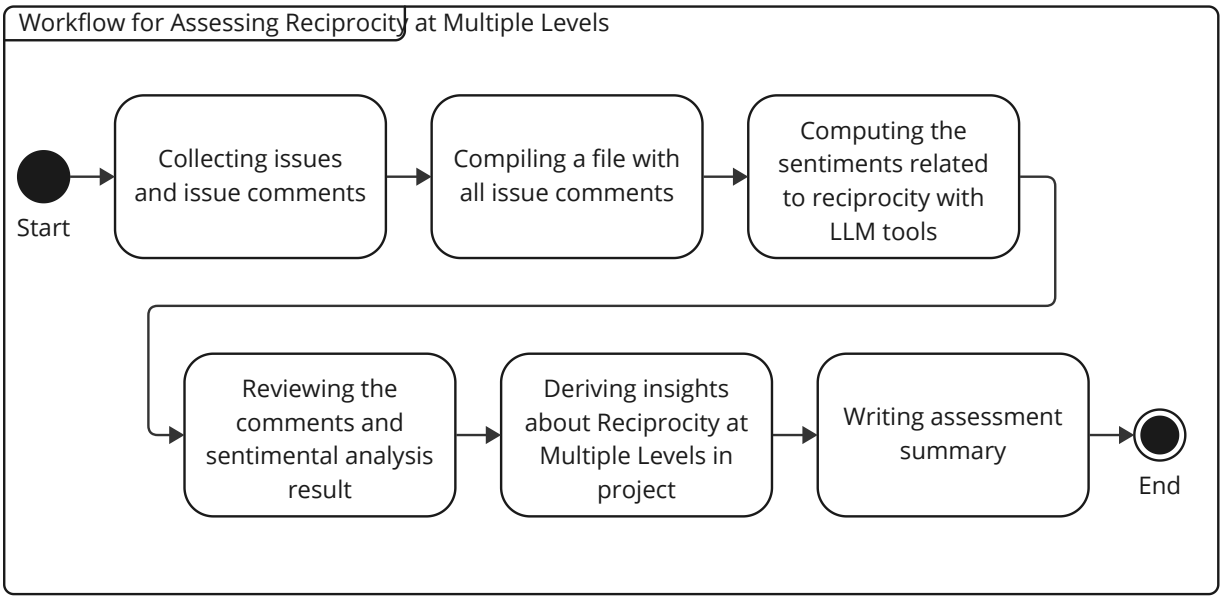

Figure B.8: The workflow for assessing Reciprocity at Multiple Levels.

Fig. B.8 shows the workflow for assessing *Reciprocity at Multiple Levels*. First, we collect issues and issue comments from the project repository, and compile a file with all issue comments. Then we use LLM to analyze the reciprocity in the collected data, an example prompt for GPT models is "Check the sentiment related to reciprocity in the given text: <issue comments>". We review the response generated by LLM, derive insights about *Reciprocity at Multiple Levels*, and write the assessment summary.

B.1.9 Trust

Fig. B.9 shows the workflow for assessing *Trust*. First, we collect issue/PR comments, and compile a file. Then we use LLM to estimate the trust within the collected comments, an example prompt for GPT models is "Estimate the trust among community members based on the given text: <comments>". We review the responses generated by LLM, derive insights about *Trust*, and write the assessment summary.

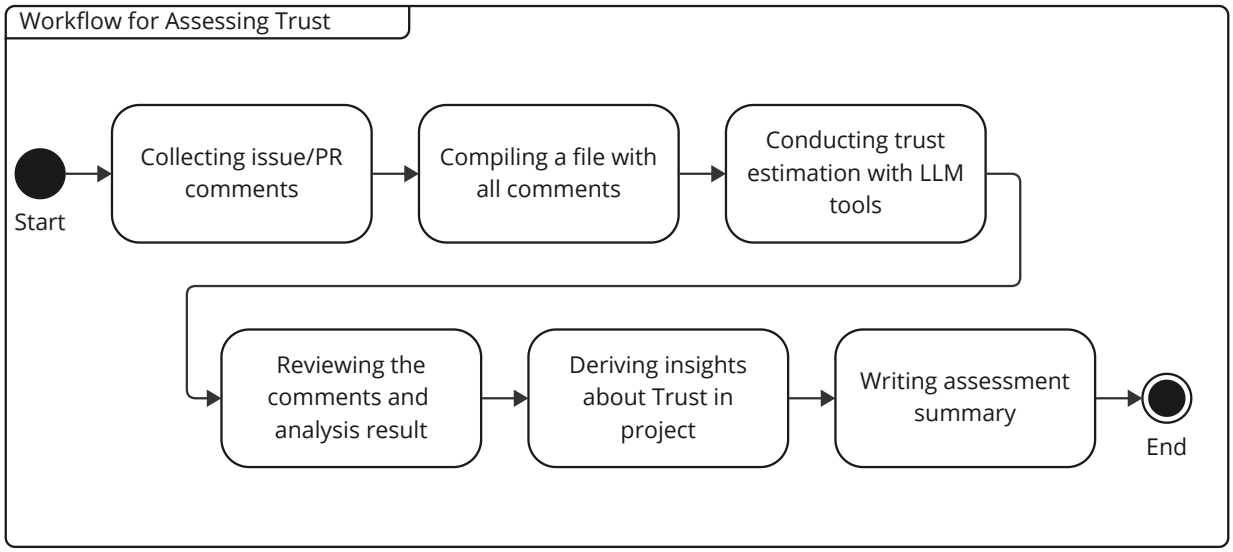

Figure B.9: The workflow for assessing Trust.

B.1.10 Universal Accessibility & Availability

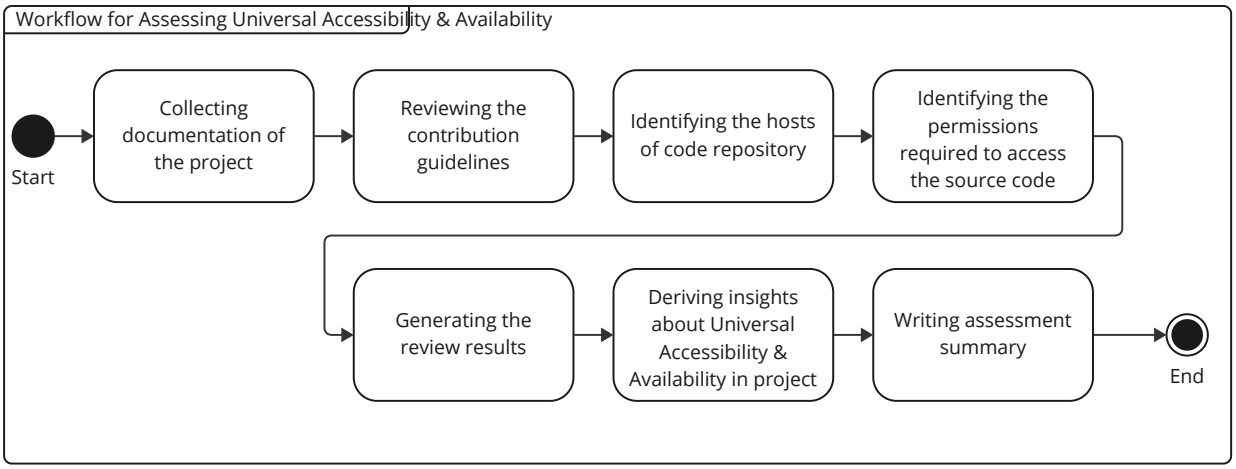

Figure B.10: The workflow for assessing Universal Accessibility & Availability

Fig. B.10 shows the workflow for assessing *Universal Accessibility & Availability*. We first collect the documents, and review the content about contribution guidelines. Then we identify the host of the code repository, and the permissions required to access the source code. Based on the review results, we derive insights about *Universal Accessibility & Availability*, and write the assessment summary.

B.2 Category: Goals

B.2.1 Broad Impacts

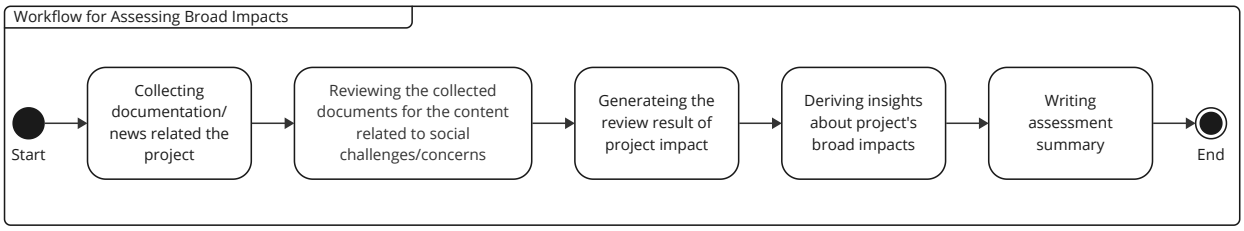

Figure B.11: The workflow for assessing Broad Impacts.

Fig. B.11 shows the workflow for assessing *Broad Impacts*. First, we collect documents and new articles related to the project. Then we review the content that describes the social challenges or concerns that the project aims to resolve, and generate review results. Based on the review results, we derive insights about *Broad Impacts*, and write the assessment summary.

B.2.2 Built-in Product & Quality Orientations

Fig. B.12 shows the workflow for assessing *Built-in Product & Quality Orientations*. Two assessment methods are included in this workflow. First, we collect all issues and PRs from the project repository, and identify issue tags that related to bug and bug fix, e.g., "bug report" and "fix patch". If such tags exist, we extract issues and PRs with those identified tags, group them into months based on their submission time, and plot the dynamics of bug report and fix. In the other assessment method, we collect the documents on testing, review the content about testing suite and testing coverage, and generate the review results. Based on the results of the two assessment methods, we derive insights about *Built-in Product & Quality Orientations*, and write the assessment summary.

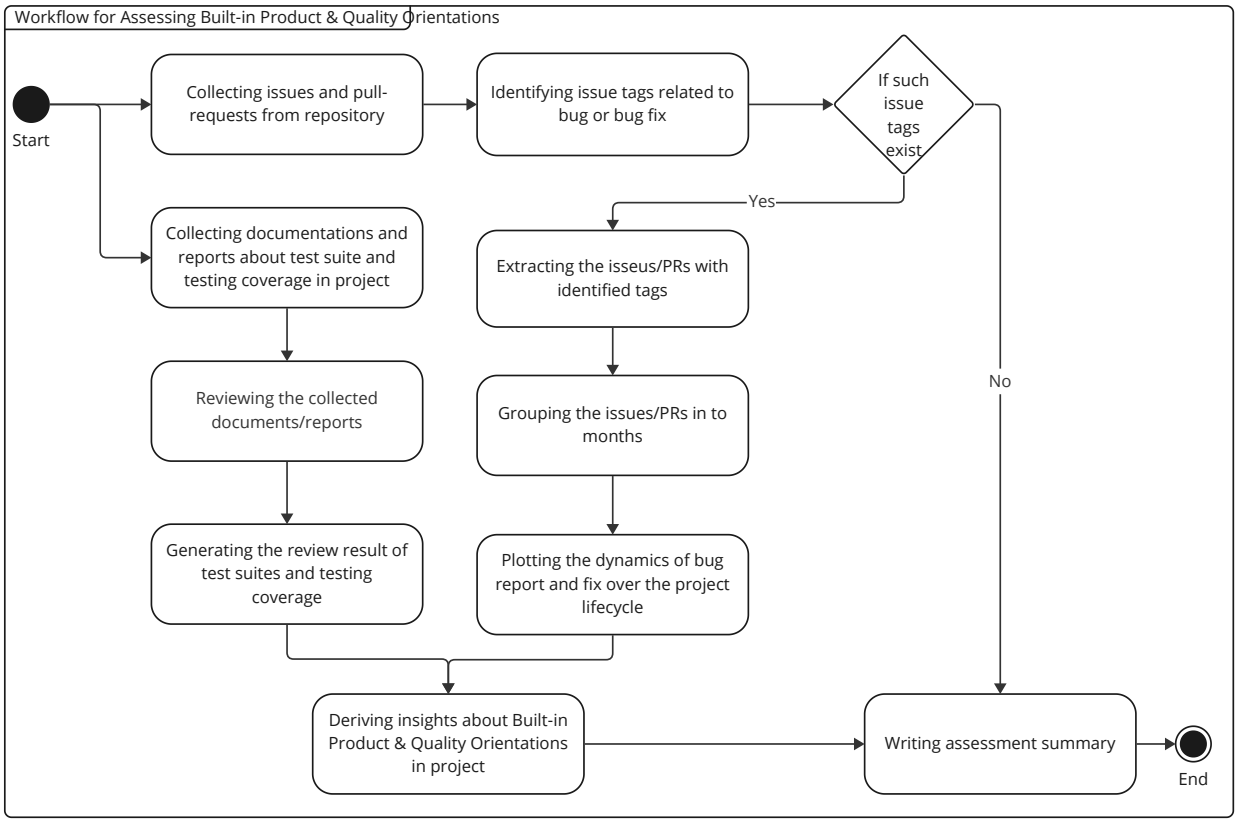

Figure B.12: The workflow for assessing Built-in Product & Quality Orientations.

B.2.3 Outreach

Fig. B.13 shows the workflow for assessing *Outreach*. First, we collect documents about social media, and meetup events, and identify whether the project utilizes any social media platforms, or has any public meetup events. If any platforms or events are identify, we compile a list of those social media platforms and meetup events, then review the content related to them, such as schedules of the meetup events, and frequency of posting on social media, and generate the review results. Based on review results, we derive insights about *Outreach*, and write the assessment summary.

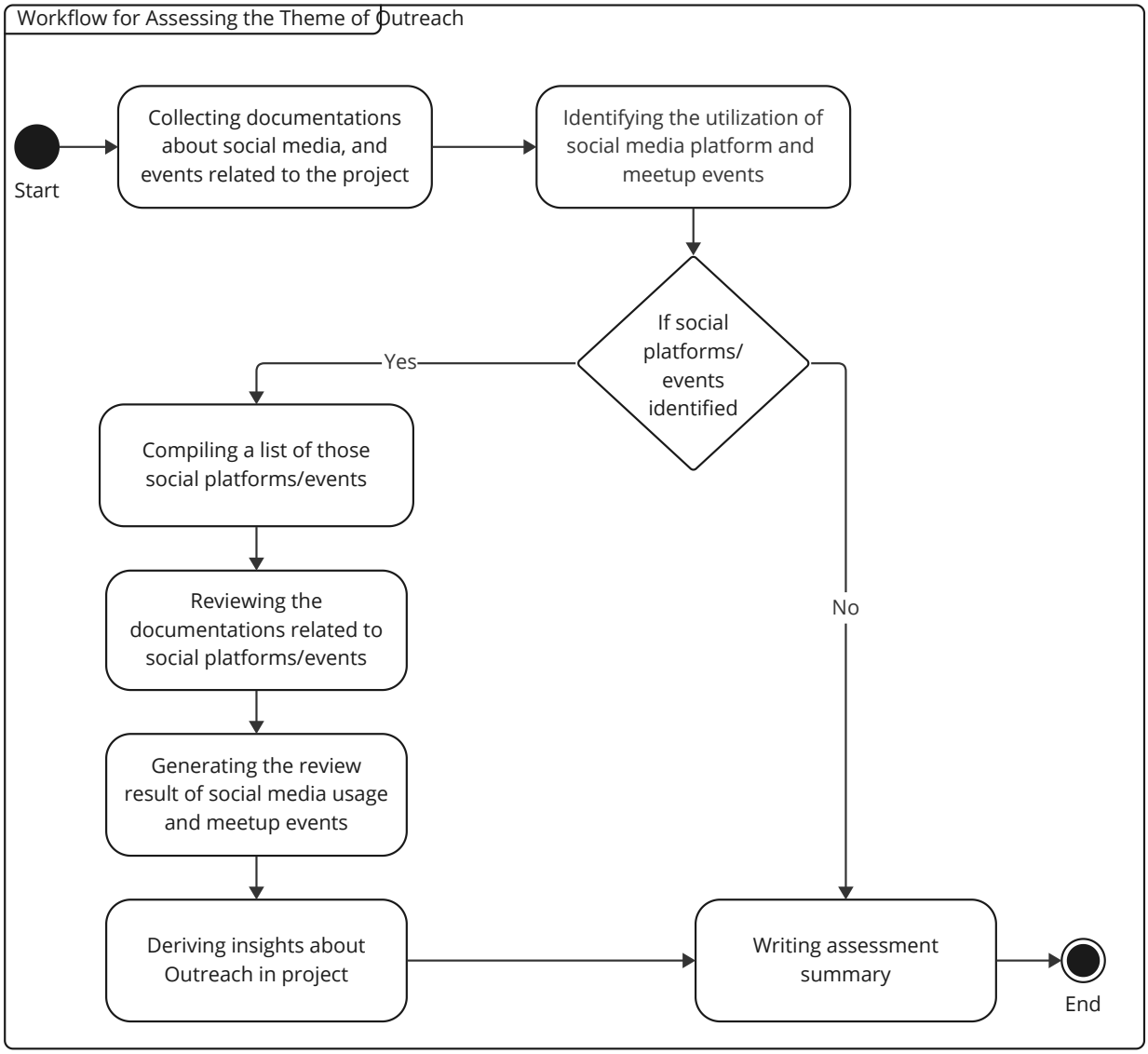

Figure B.13: The workflow for assessing Outreach

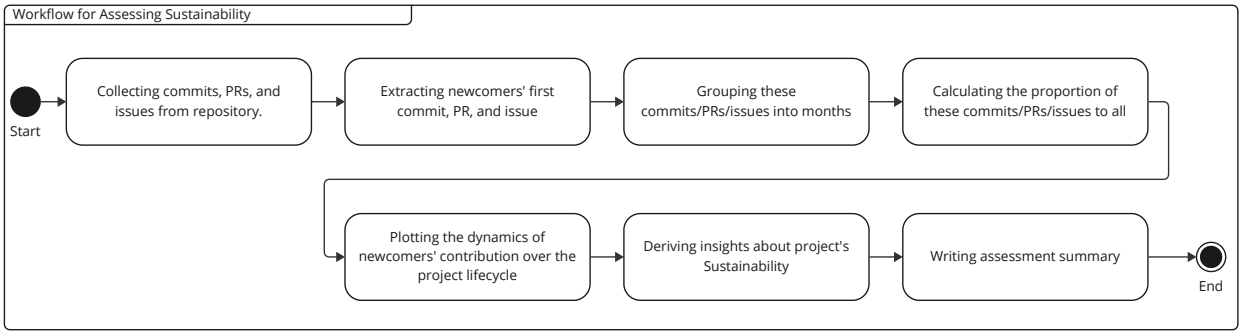

Figure B.14: The workflow for assessing Sustainability.

B.2.4 Sustainability

Fig. B.14 shows the workflow for assessing *Sustainability*. First, we collect commits, issues, and PRs from the project repository, and identify the ones submitted by newcomers. Then we group those collected commits/issues/PRs into months, and calculate the proportion of commits/issues/PRs submitted by newcomers. Next, we plot the dynamics of newcomers' contribution with commits/issues/PRs every month. Note that various charts could be generated, for commits, issues, and PRs, respectively. Based on the results, we derive insights about *Sustainability*, and write the assessment summary.

B.2.5 Innovation

Fig. B.15 shows the workflow for assessing *Innovation*. Two assessment methods are included. First, we collect issues and issue comments from the project repository, and identify whether there are tags related to new features/functions, e.g., "new feature request". If such tags exist, we extract all issues with those tags, group them into months, and calculate the proportion of them. Then we plot the dynamics of those issues. In the other assessment method, we compile all comments for those issues, and utilize LLM to analyze the sentiments in those comments, an example prompt for GPT-3.5 is "Given the following text, analyze the sentiment towards the new functions or functions. The text is: <comment text>". Based on the charts and the responses generated by LLM, we derive insights about *Innovation*, and write the assessment summary.

B.2.6 Ubiquitous Penetration

Fig. B.16 shows the workflow for assessing *Ubiquitous Penetration*. First, we collect documents and news articles about the use case of the project. Then we review the content for

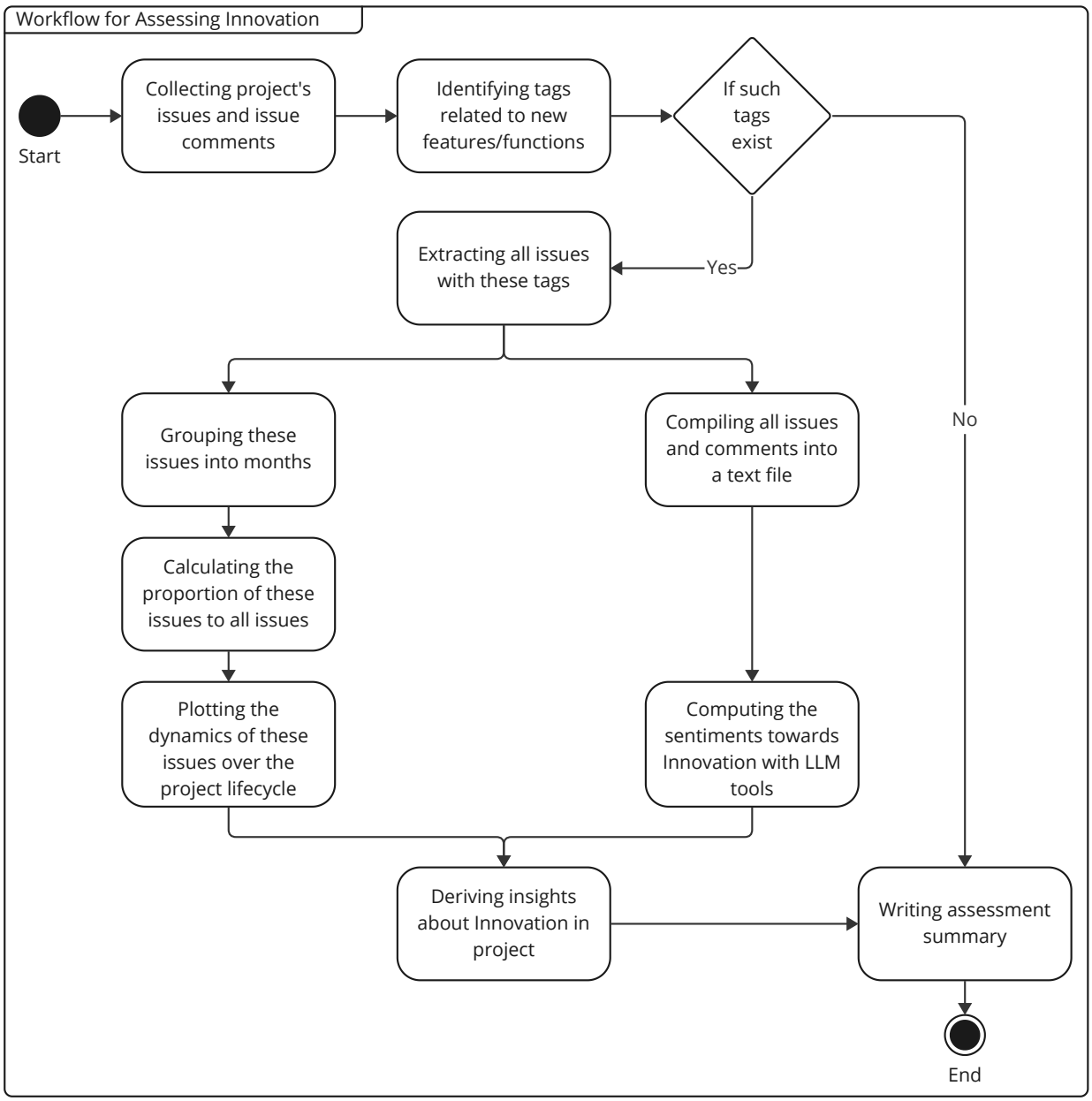

Figure B.15: The workflow for assessing Innovation.

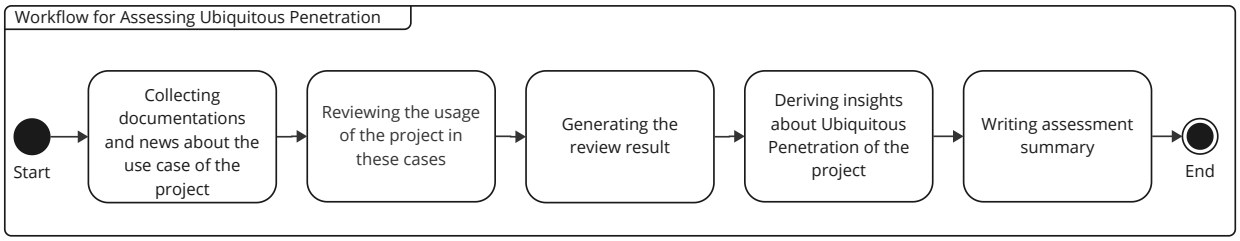

Figure B.16: The workflow for assessing Ubiquitous Penetration.

how the project is used in practice, and generate the review results. Based on review results, we derive insights about *Ubiquitous Penetration*, and write the assessment summary.

B.3 Category: Activities

B.3.1 Copyright, Licensing, & Legal Implications

Fig. B.17 shows the workflow for assessing *Copyright, Licensing, & Legal Implications*. Two assessment methods are included. First, we search online for patent and legal conflicts related to the project, and review the collected content. Then we identify the legal practice of the project with licensing, and generate review results. In the other assessment method, we collect documents from project documentation, and review the content related to licensing. Then we analyze the type of license in the project, i.e., the type of license, permission and rights granted, etc. We further identify whether additional legal documents, such as CLA, are required for contribution. Then we generate the review results on the collected project documents. Based on the review results of the two assessment methods, we derive insights about *Copyright, Licensing, & Legal Implications*, and write the assessment summary.

B.3.2 Governance & Decision Making

Fig. B.18 shows the workflow for assessing *Governance & Decision Making*. First, we collect the documents and review the content related to governance in the project. THen we identify the governance structure in the project, as well as the decision-making process with the governance structure, and generate review results. Based on the review results, we derive insights about *Governance & Decision Making*, and write the assessment summary.

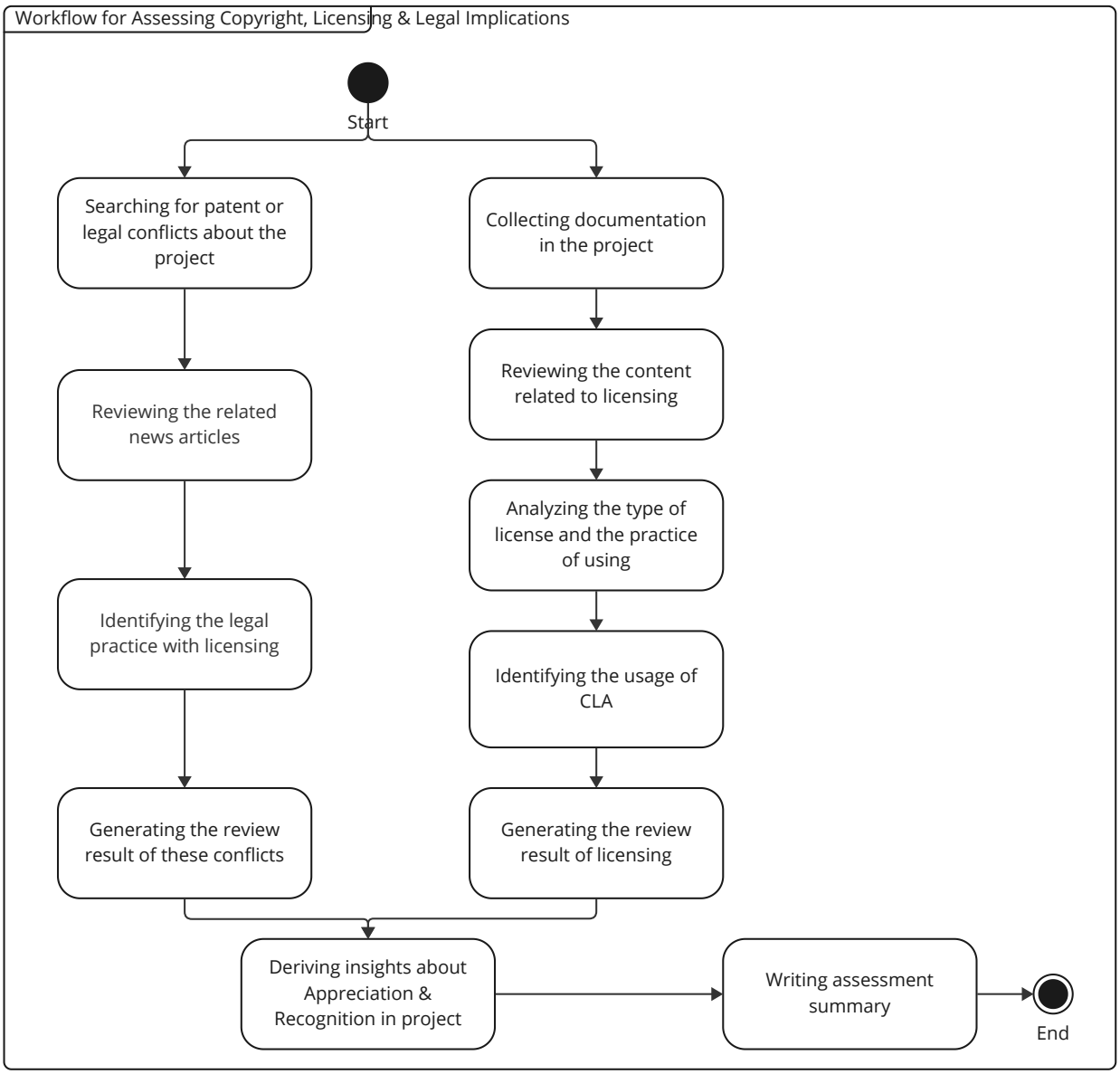

Figure B.17: The workflow for assessing Copyright, Licensing, & Legal Implications.

B.3.3 Personal Development

Fig. B.19 shows the workflow for assessing *Personal Development*. First, we collect commit data from the project repository. Then we calculate the centrality degrees of contributors after each of their commit[2], and have their traces of centrality degrees in the project. We plot the dynamics of the centrality degree traces of all contributors. Based on the plot-

[2]see [61] for more details on centrality degree

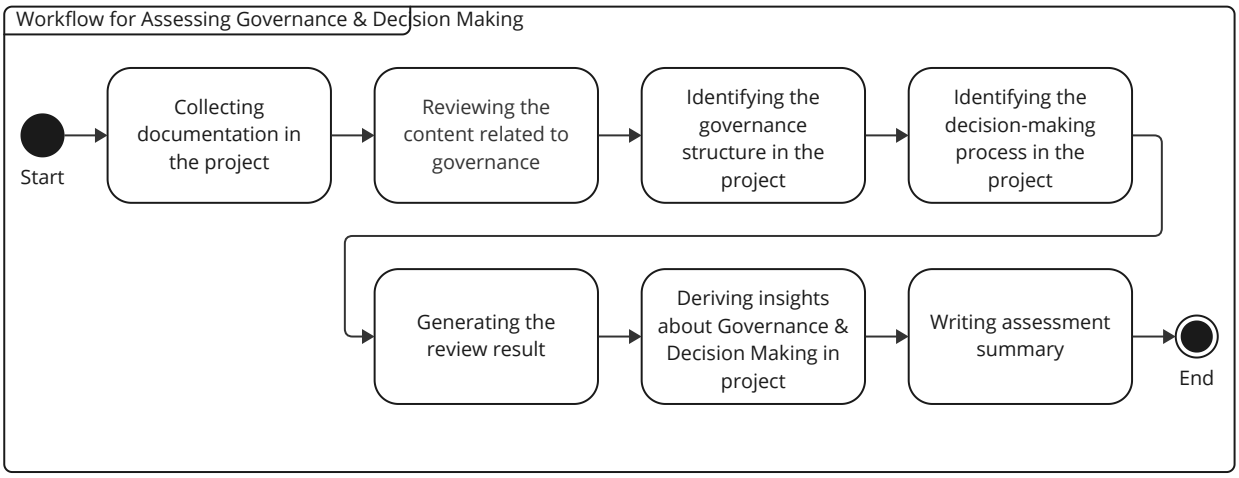

Figure B.18: The workflow for assessing Governance & Decision Making.

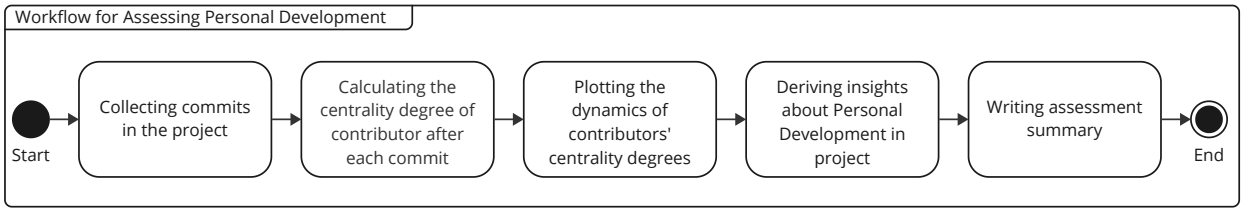

Figure B.19: The workflow for assessing Personal Development.

ted centrality degree trace, we derive insights about *Personal Development*, and write the assessment summary.

B.3.4 Social Production

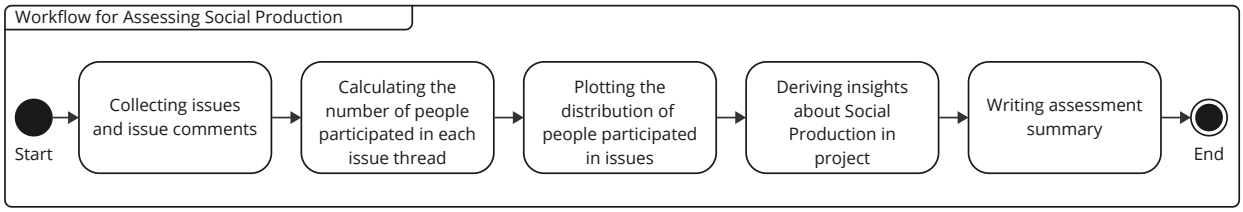

Figure B.20: The workflow for assessing Social Production.

Fig. B.20 shows the workflow for assessing *Social Production*. First we collect all issues and issue comments. Then we count the number of people participated in the discussion in each issue thread, and plot the distribution of people participated in issues. Based on analyzed

results, we derive insights about *Social Production*, and write the assessment summary.

B.3.5 Work Organization & Practices

Fig. B.21 shows the workflow for assessing *Work Organization & Practices*. Two assessment methods are included in the workflow. First, we collect all commits in the project repository, and extract the timezone information of the contributors who submitted the commits. Then we compute and plot the distribution of the time zones. In the other assessment method, we collect the documents, and review the content about communication during development process. Then we identify the communication channels used by contributors to collaborate on development; and the scheduled meetup events for project development. We generate review results of the communication channels and developer meetup events. Based on the results of the two assessment methods, we derive insights about *Work Organization & Practices*, and write the assessment summary.

B.4 Category: Membership

B.4.1 Individual Identity

Fig. B.22 shows the workflow for assessing *Individual Identity*. First, we collect all issue/PR comments, and compile a file with all comments. Then we utilize LLM to analyze the usage of pronouns in the comments, an example prompt for GPT-3.5 is "Provide statistical summary on the pronoun usage for the given text: <comment text>". Based on the responses generated by LLM, we derive insights about *Individual Identity*, and write the assessment summary.

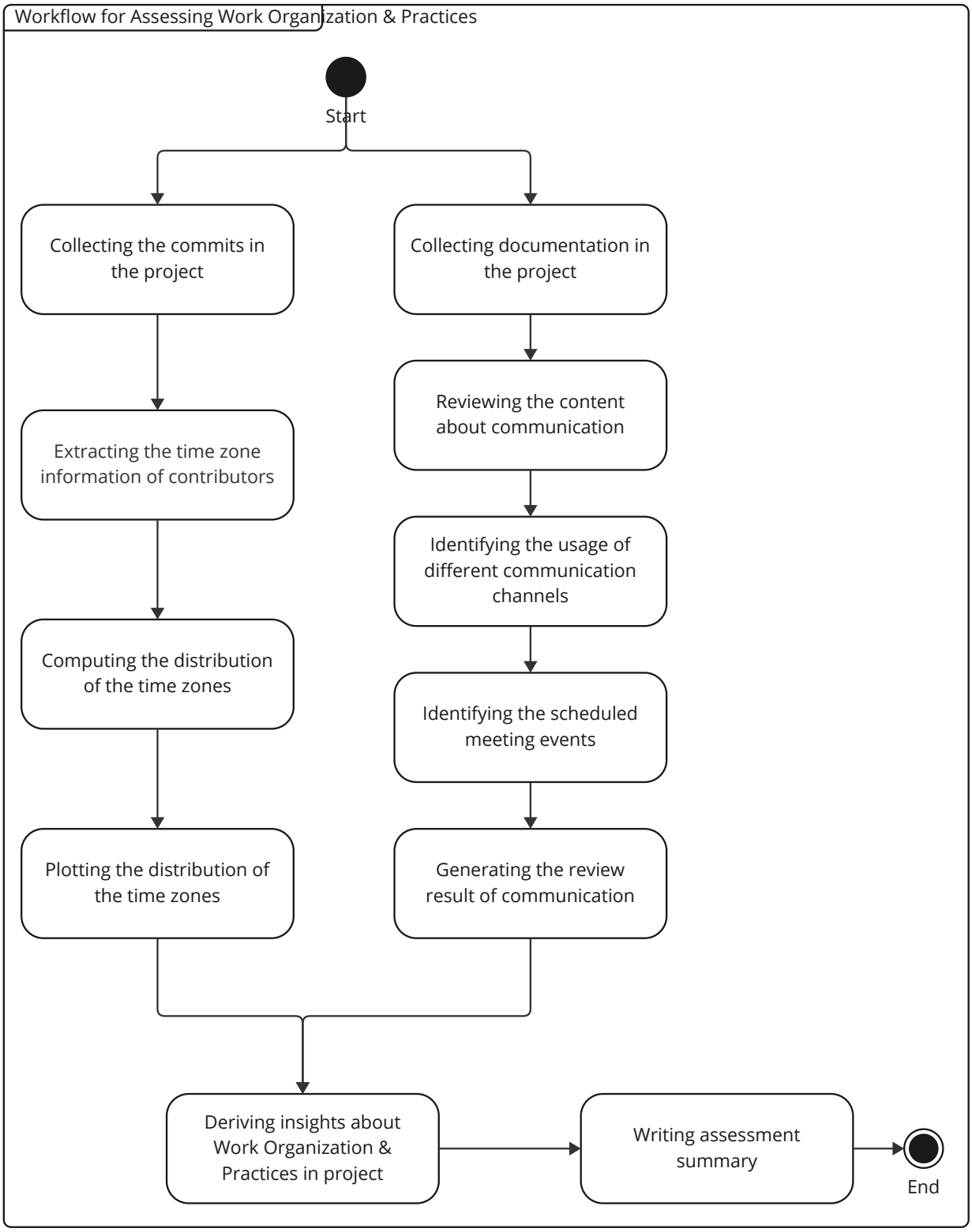

Figure B.21: The workflow for assessing Work Organization & Practices.

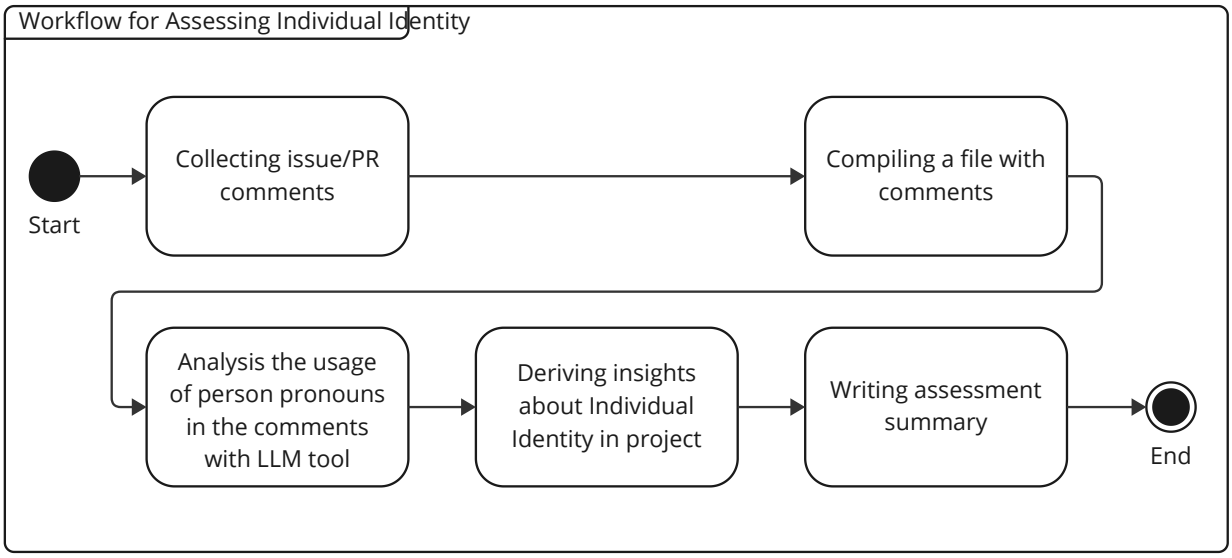

Figure B.22: The workflow for assessing Individual Identity.

B.4.2 Ideological Leadership

Fig. B.23 shows the workflow for assessing *Ideological Leadership*. Two assessment methods are included. First, we collect documents and contributor list, and identify whether there is leadership role in the project. If such role exist, we review the content related to the leadership role, and generate review results. In the other assessment method, we collect issue/PR comments, and compile a file with all comments. Then we adopt LLM to identify the content related to leadership, an example prompt for GPT-3.5 is "Identify the content indicating ideological leadership in the given text: <comment text>". Based on the review results and the responses generated by LLM, we derive insights about *Ideological Leadership*, and write the assessment summary.

B.4.3 Interpersonal Relationships

Fig. B.24 shows the workflow for assessing *Interpersonal Relationships*. First, we collect all issue/PR comments, and compile a text file. Then we adopt LLM to analyze the sen-

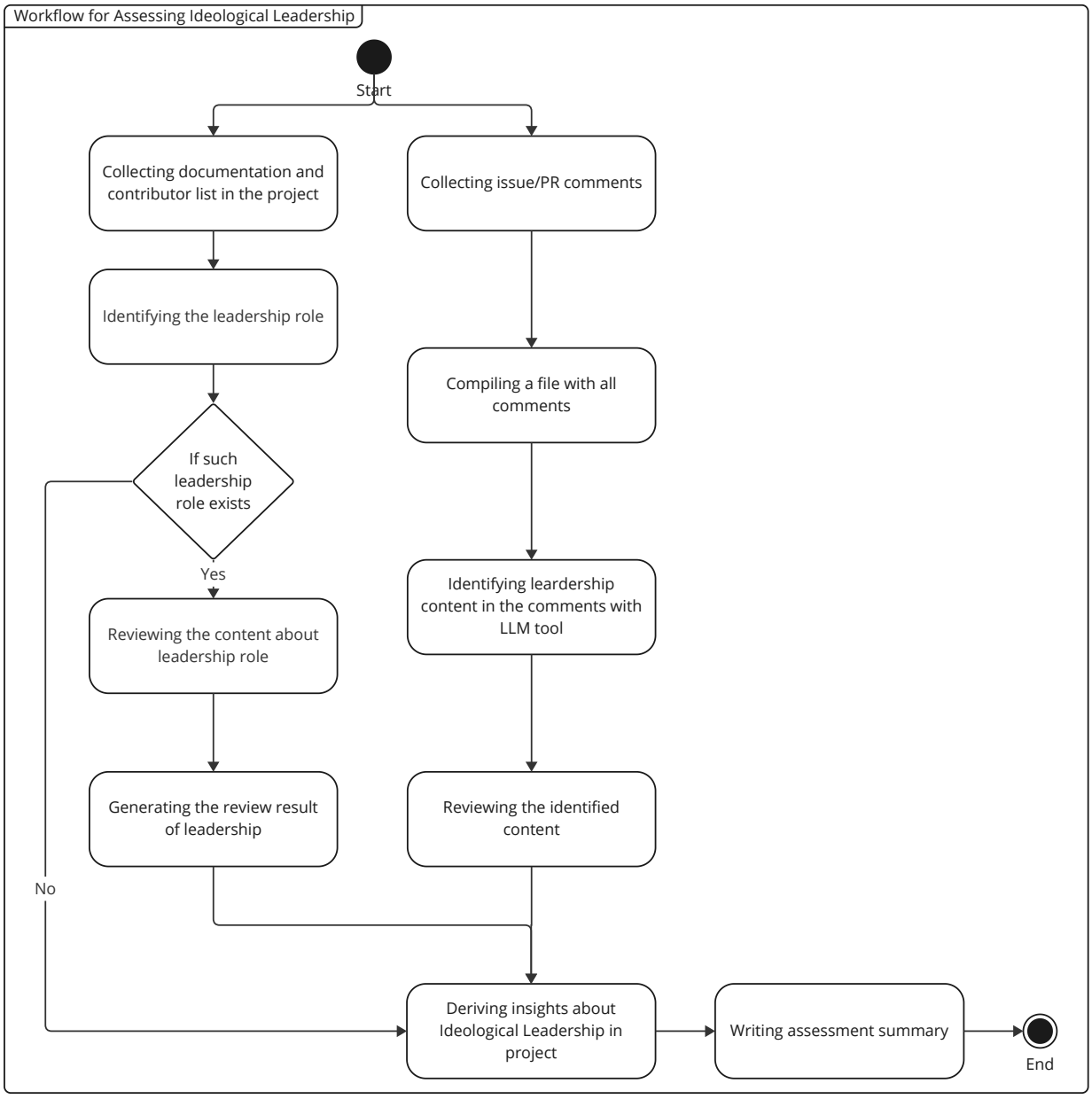

Figure B.23: The workflow for assessing Ideological Leadership.

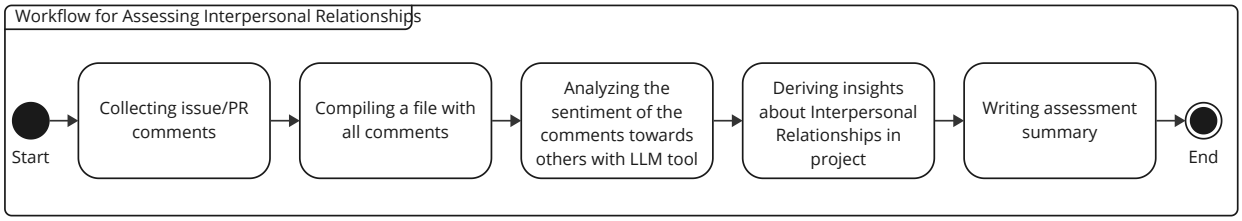

Figure B.24: The workflow for assessing Interpersonal Relationships.

timent towards others in the comments, an example prompt for GPT-3.5 is "Analyze the interpersonal sentiment in the given text: <comment text>". Based on the responses generated by LLM, we derive insights about *Interpersonal Relationships*, and write the assessment summary.

B.4.4 Member Hierarchy & Roles

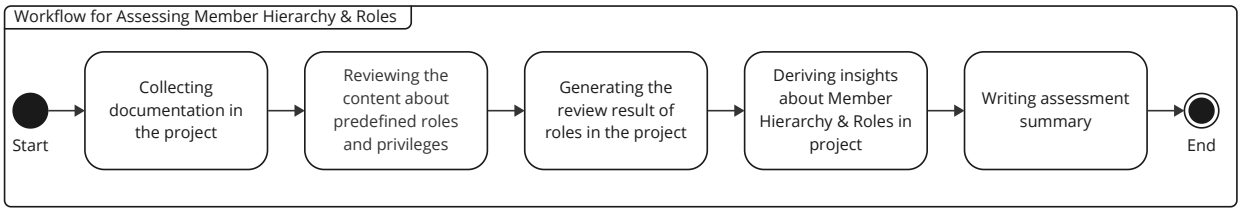

Figure B.25: The workflow for assessing Member Hierarchy & Roles.

Fig. B.25 shows the workflow for assessing *Member Hierarchy & Roles*. First, we collect documents in the project. Then we review the content related to predefined roles and their privileges, and generate review results. Based on the review results, we derive insights about *Member Hierarchy & Roles*, and write the assessment summary.

B.5 Category: Resources

B.5.1 Dealing with Barriers & Restrictions

Fig. B.26 shows the workflow for assessing *Dealing with Barriers & Restrictions*. We first collect documents in the project, and review the content regarding tutorials/guidelines for newcomers. With each of those documents, we analyze its depth and coverage in introducing the project, and generate review results. Based on those review results, we derive insights about *Dealing with Barriers & Restrictions*, and write the assessment summary.

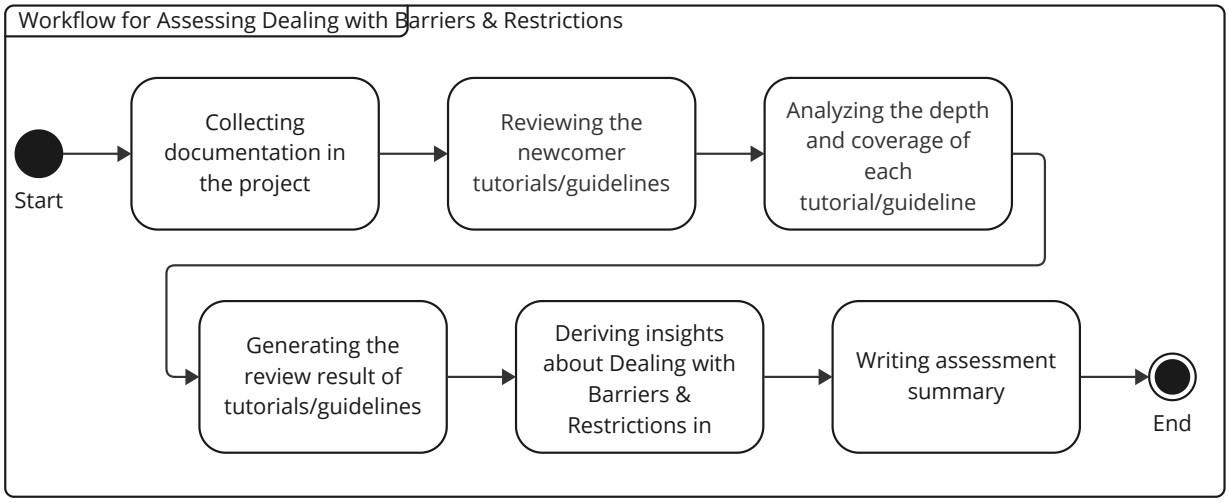

Figure B.26: The workflow for assessing Dealing with Barriers & Restrictions.

B.5.2 Incentives, Financing, & Funding

Fig. B.27 shows the workflow for assessing *Incentives, Financing, & Funding*. Two assessment methods focused on the project documentation are included. First, we review the content about supporting individual developers in the project, and identify whether the project offers employment opportunities for developers, as well as other types of incentives for individual developers. Then we generate review results on supporting individual developers. In the other assessment method, we review the content related to donation and sponsorship, identify whether there are any donation/sponsorship mechanisms for companies and organizations, and generate review results. Based on the review results of two assessment methods, we derive insights about *Incentives, Financing, & Funding*, and write the assessment summary.

B.5.3 Knowledge & Expertise

Fig. B.28 shows the workflow for assessing *Knowledge & Expertise*. First, we collect documents in the project, and identify and review the content related to tutorials. Then we

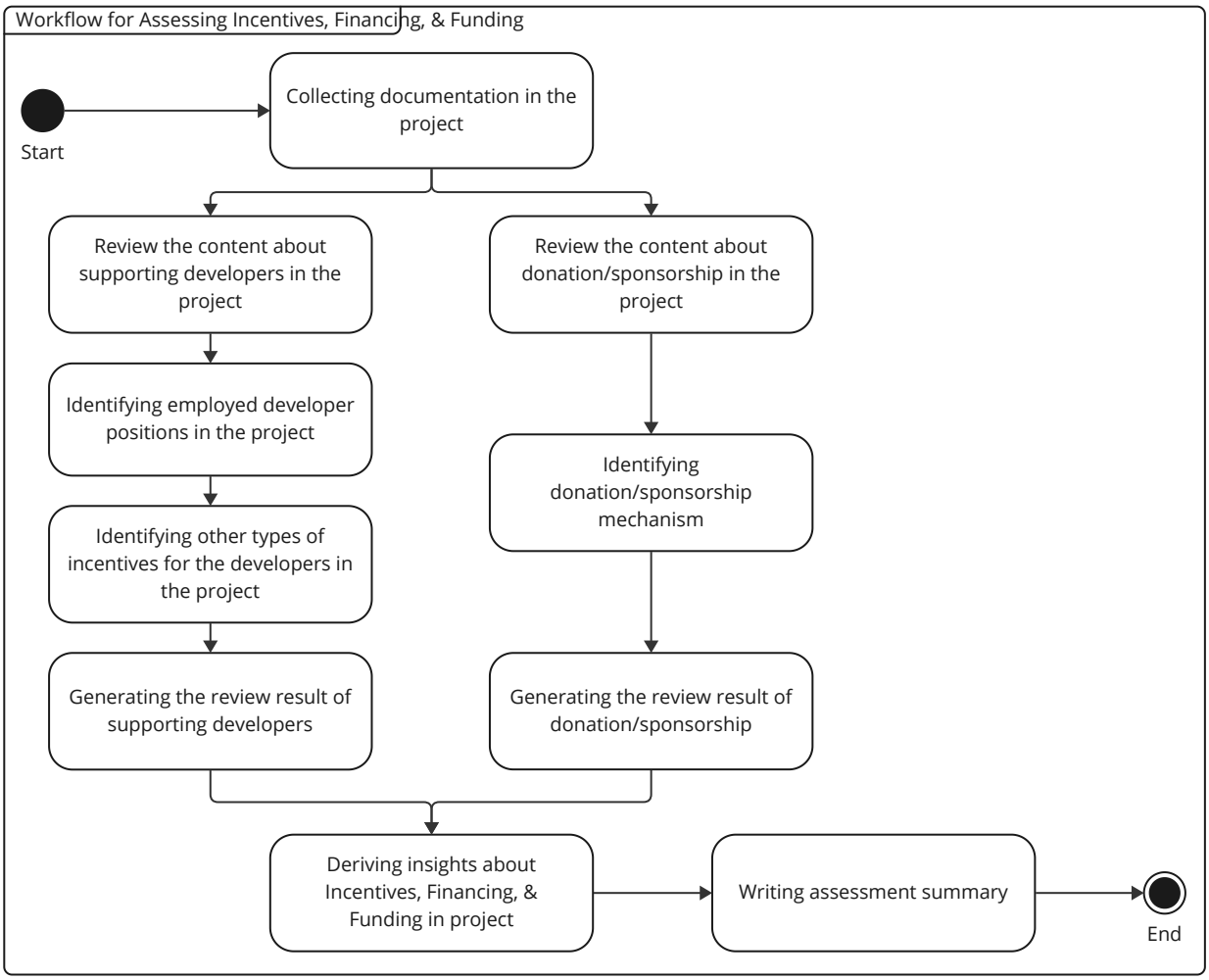

Figure B.27: The workflow for assessing Incentives, Financing, & Funding.

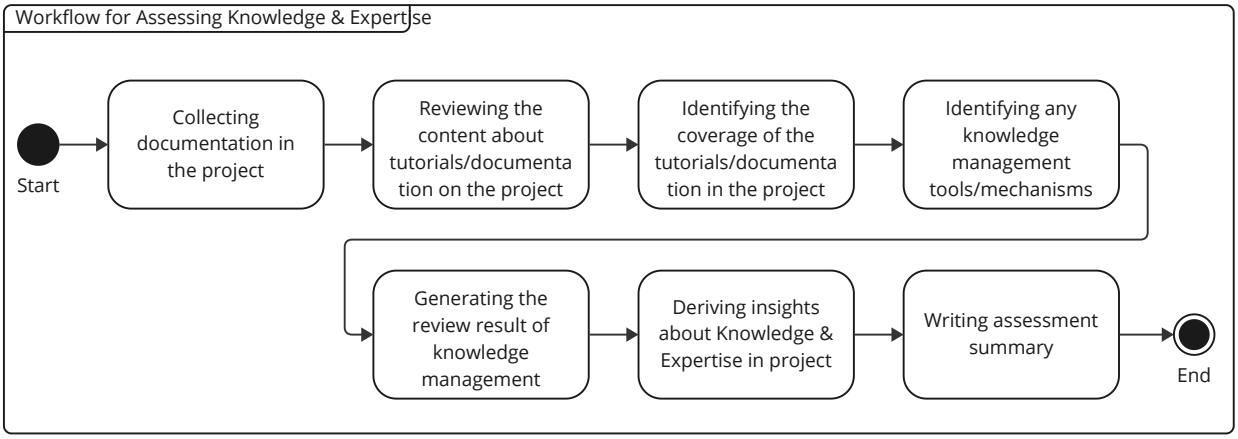

Figure B.28: The workflow for assessing Knowledge & Expertise.

identify the knowledge management tools/mechanisms mentioned in those tutorials. Next, we generate review results on those collected documents. Based on the review results, we derive insights about *Knowledge & Expertise*, and write the assessment summary.

B.5.4 Supportive Facilities & Mechanisms

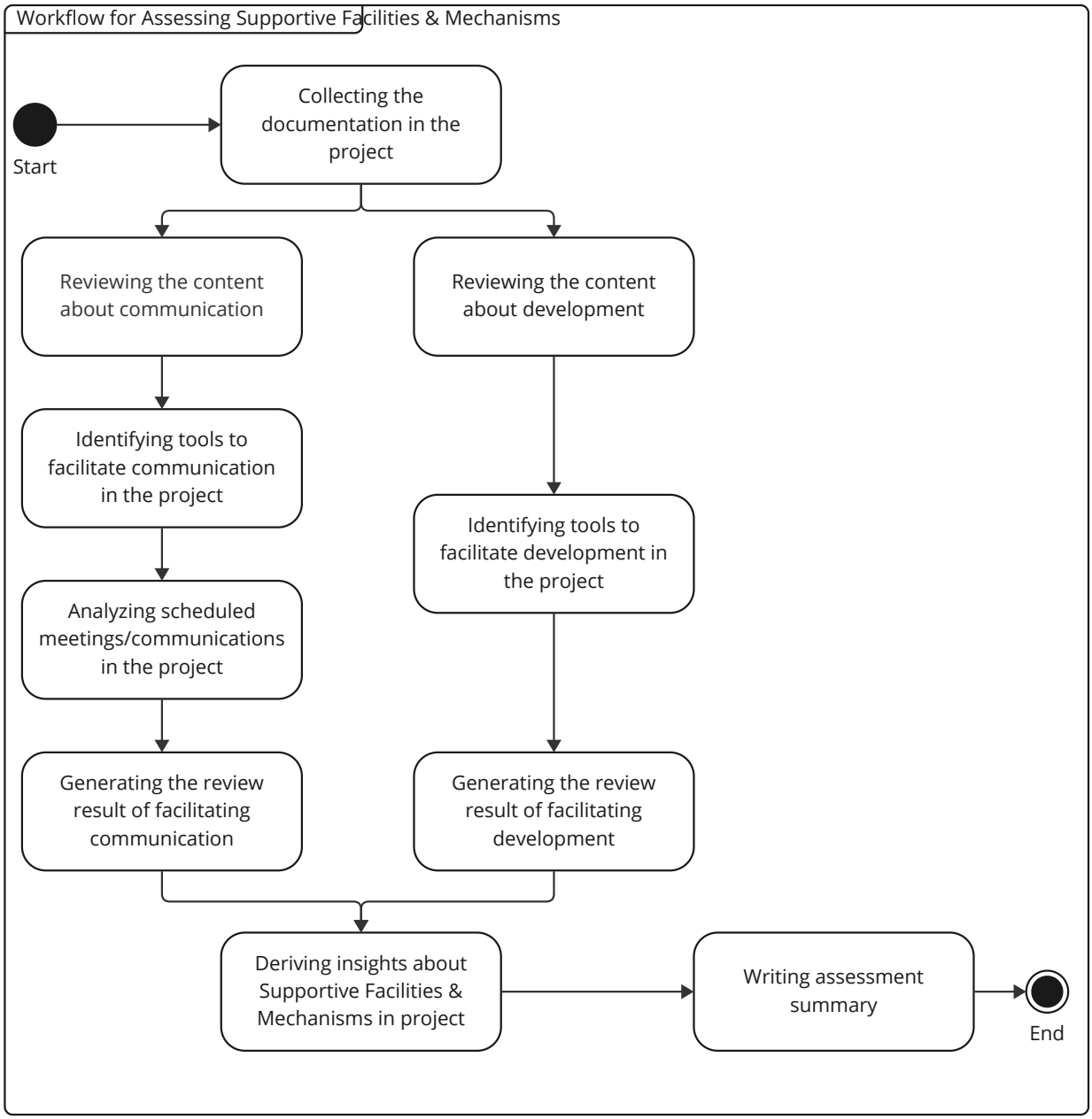

Figure B.29: The workflow for assessing Supportive Facilities & Mechanisms.

Fig. B.29 shows the workflow for assessing *Supportive Facilities & Mechanisms*. Two assessment methods are included, and both analyze the project documentation. First, we review the content about communication, and identify the communication tools adopted in the project development; we also analyze whether there are scheduled meetings or communications in the project. Then we generate the review results of facilitating communication in the project. In the other method, we review the content related to development, identity tools/toolchains adopted to facilitate the project development, and generate review results. Based on the review results of the two assessment methods, we derive insights about *Supportive Facilities & Mechanisms*, and write the assessment summary.

B.6 Category: Position/Group Relations

B.6.1 Interaction with Emerging Technologies

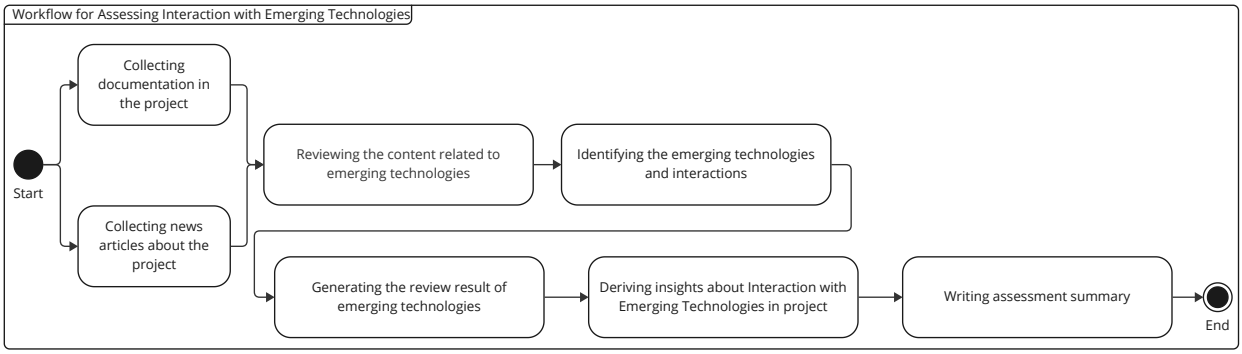

Figure B.30: The workflow for assessing Interaction with Emerging Technologies.

Fig. B.30 shows the workflow for assessing *Interaction with Emerging Technologies*. First, we collect documentation from the project, and online news articles about the project. We review the content related to emerging technologies in the collected documents, i.e., the particular technologies and their relations with the project, and generate review results. Based on the review results, we derive insights about *Interactions with Emerging Technologies*, and

write the assessment summary.

B.6.2 Interaction with Commercial Software Development

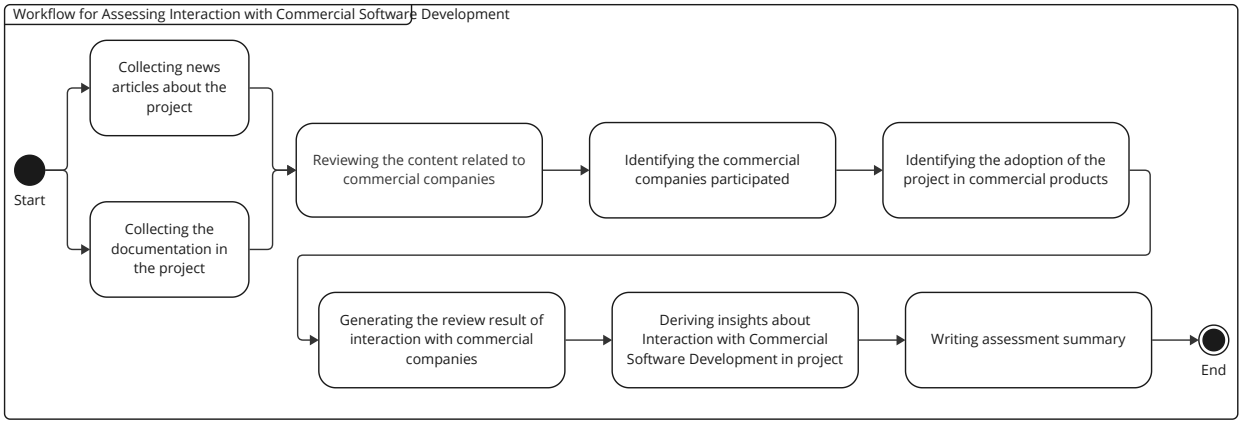

Figure B.31: The workflow for assessing Interaction with Commercial Software Development

Fig. B.31 shows the workflow for assessing *Interaction with Commercial Software Development*. First, we collect documentation from the project and online news articles about the project. We review the content related to commercial companies, and identify the companies that participate in the project development. Then, we identify the adoption of the project in commercial products. We generate review results based on the identified content. Based on the review results, we derive insights about *Interaction with Commercial Software Development*, and write the assessment summary.

B.6.3 Market & Users

Fig. B.32 shows the workflow for assessing *Market & Users*. First, we collect documentation from the project and online news articles about the market and users. Then we identify the project's targeted market, and identify the targeted users. We generate review results with the identified content. Based on the review results, we derive insights about *Market & Users*, and write the assessment summary.

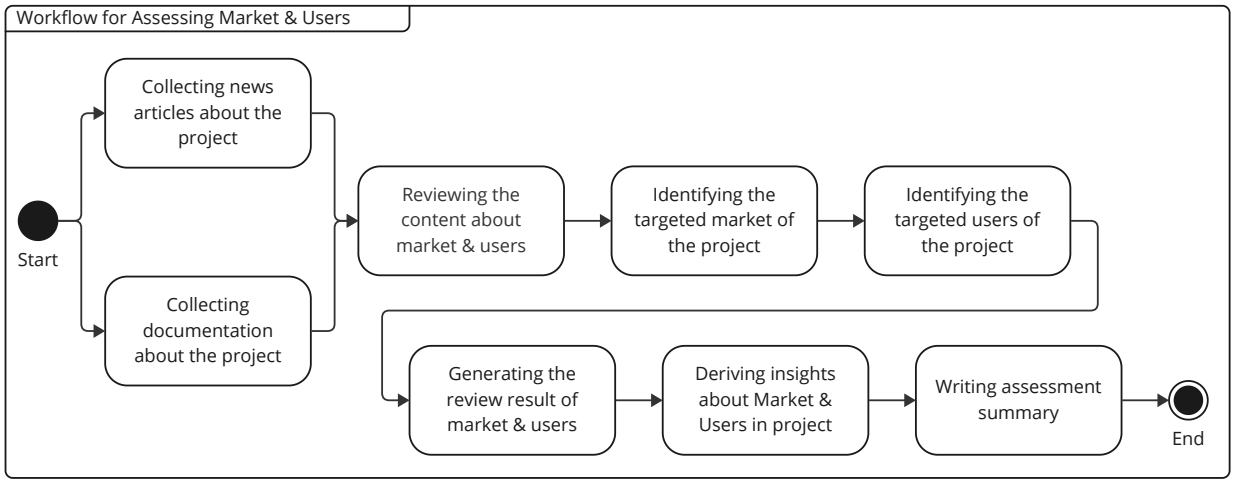

Figure B.32: The workflow for assessing Market & Users.

Appendix C

Report of Assessment on Zephyr

This chapter reports the detailed assessment on Zephyr. Before reading the assessment results, we would like to issue a disclaimer: The assessment results were based on public data sources, some of them might not represent the official opinion of the projects. The assessment we conducted was a snapshot of the OSS ideologies in the project, the assessment results for particular projects are limited to this snapshot, and could not reflect the future changes within the projects. The data and assessments were indicators that reflected the themes, they might not reflect all practices related to the assessed themes.

C.1 Category: Values/Norms

C.1.1 Theme: Appreciation & Recognition

The project would sometimes express appreciation and recognize contributors' contributions, and give them credits in the release notes or pull request comments, but not all the time. For example, Fig. C.1 is an example of release notes, each contributor who contributed

to this release was listed with the items they contributed, which gave contributors proper recognition. In some pull requests, as shown in Fig. C.2, sometimes the reviewer would explicitly thank contributors for their work in the comments. However, neither the recognition in release notes nor the appreciation in pull request comments occurred every time, they often depended on the discretion of the members or reviewers involved.

C.1.2 Theme: Autonomy

First, 58 contributors were affiliated with the project organization, while 339 contributors were not, and some of the non-affiliated contributors might be involved in the development without bonding to employment, which indicates self-determined contributions. Second, according to the submission time of the code commits shown in Fig. C.3, the majority of the code commits were submitted during working hours on workdays, while a relatively small number of the code commits were submitted outside of the working time, which reflected self-determined contribution from some contributors. Thus, autonomy exists in the project community, but to a limited extent.

C.1.3 Theme: Comparative Advantages

In the project documentation, there is a section[1] that lists some distinguishing features of the project, as the comparative advantages compared with other similar projects, including *Extensive suite of Kernel services*, *Multiple Scheduling Algorithms*, *Cross Architecture*, *Memory Protection* and so on. For example, the feature *Cross Architecture* reflected the project "supports a wide variety of supported boards with different CPU architectures and developers tools. Contributions have added support for an increasing number of SoCs, platforms, and drivers."

[1] `https://docs.zephyrproject.org/latest/introduction/index.html`

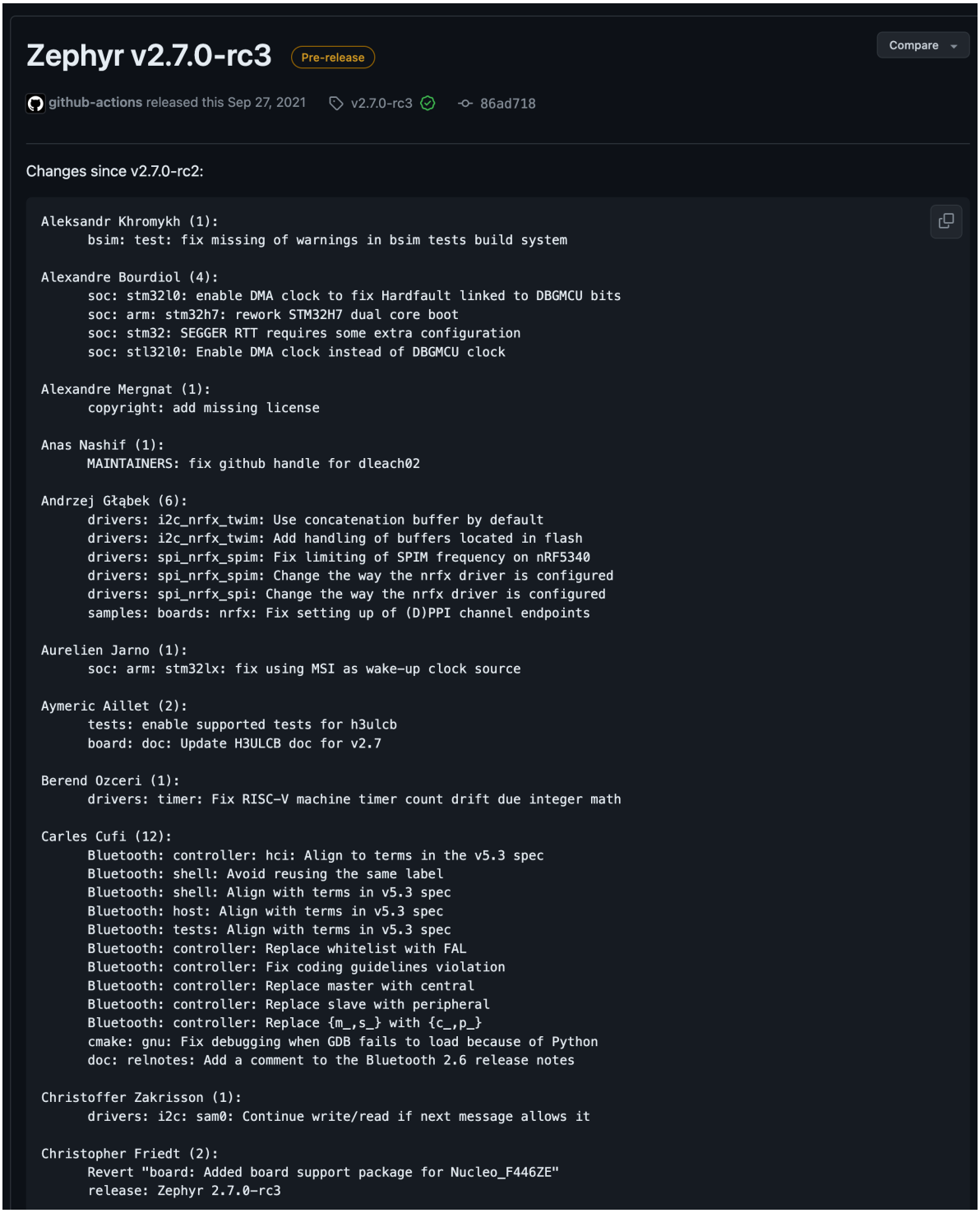

Figure C.1: Screenshot of an example release notes.

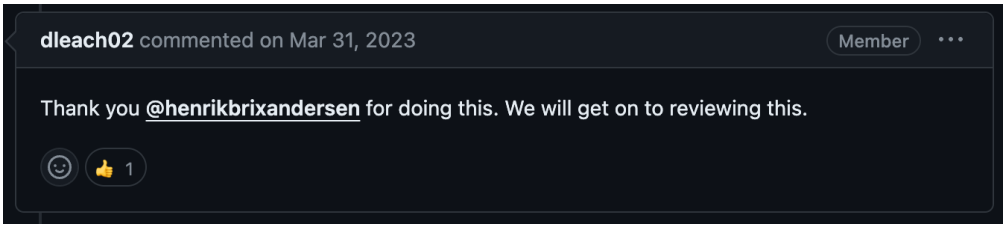

Figure C.2: Screenshot of an example pull request comment.

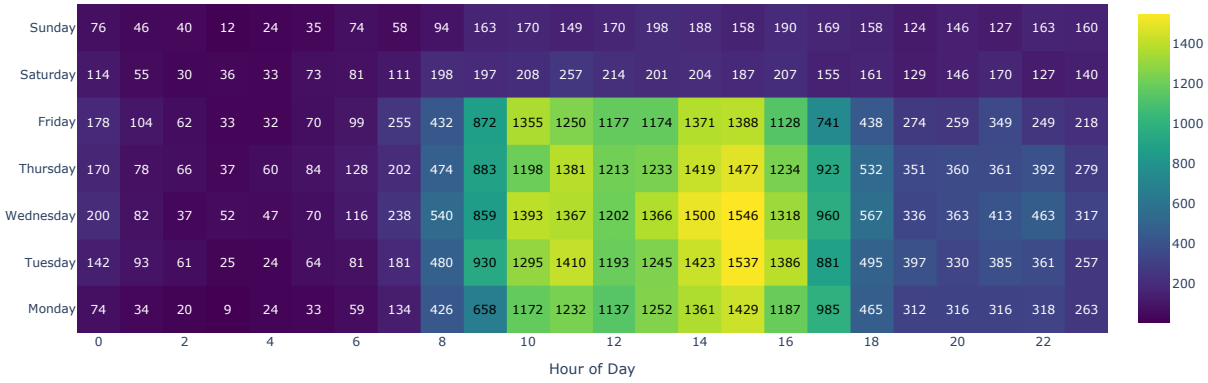

Figure C.3: The code commit activity heatmap.

C.1.4 Theme: Constantly Changing & Evolving

Fig. C.4 & C.5 show the monthly commit frequency and the number of newcomers in the project, both the commit frequency and the number of newcomers were increasing in general, which indicated the project community was constantly changing and evolving.

C.1.5 Theme: Ethics

The project provides a Code of Conduct document[2] adapted from the Contributor Covenant (version 2.1). There was a statement on the pledge of the project community:

"*We as members, contributors, and leaders pledge to make participation in our*

[2]`https://github.com/pytorch/pytorch/blob/main/CODE_OF_CONDUCT.md`

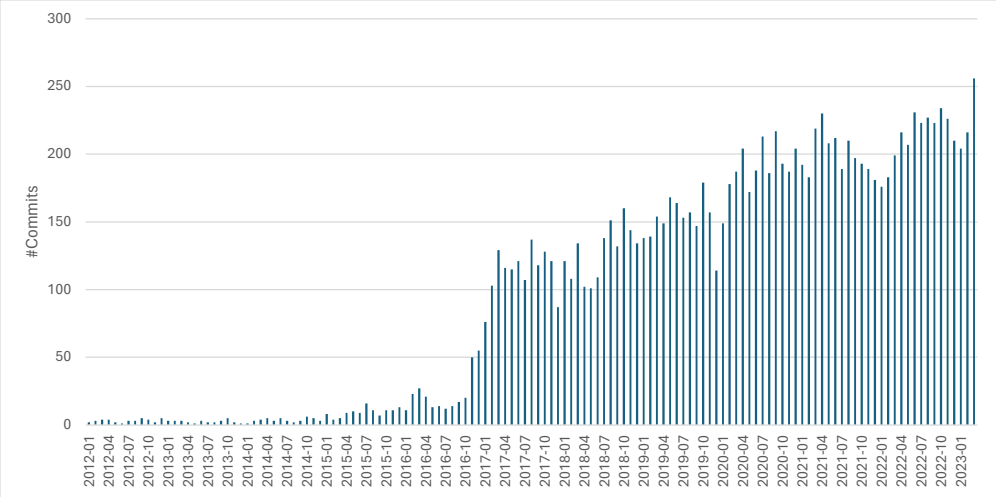

Figure C.4: The monthly commit frequency.

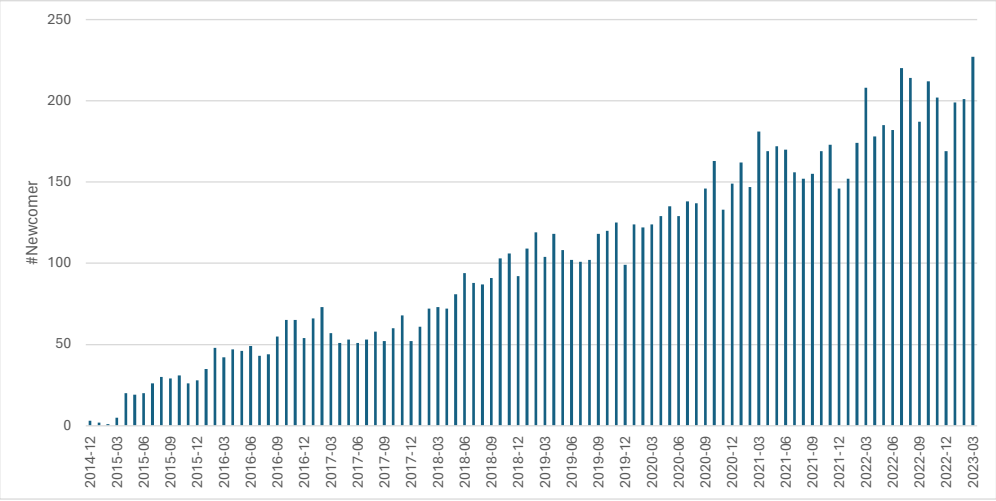

Figure C.5: The number of newcomers each month.

community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, caste, color, religion, or sexual identity and orientation."

"*We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.*"

Moreover, the document listed some examples of acceptable and unacceptable behaviors as community standards, and detailed guidelines to enforce that Code of Conduct in the community. Therefore, the project followed the standard code of conduct with explicit and detailed guidelines and enforcement processes.

C.1.6 Theme: Openness & Transparency

In the same Code of Conduct document, openness was mentioned in "an open, welcoming, diverse, inclusive, and healthy community," as one of the values in the community. However, transparency was not explicitly emphasized in the project documentation. Moreover, the project repository is hosted on GitHub, which guarantees a certain level of openness and transparency in the community.

C.1.7 Theme: Meritocracy

Three different roles are defined in the documentation[3], i.e., *Maintainers*, *Collaborators*, and *Contributors*. However, there are no explicit criteria on how to nominate/promote to those roles, or how to remove from those roles.

Moreover, based on the centrality degrees of the three role groups shown in Fig. C.6, there are significant differences across those three different roles. In general, the centrality degrees of *Maintainers* tend to be higher than *Collaborators*, and *Contributors*. Thus, meritocracy exists in the project community according to the distribution of the centrality degrees, but lack of explicit criteria/description.

[3] `https://docs.zephyrproject.org/latest/project/project_roles.html`

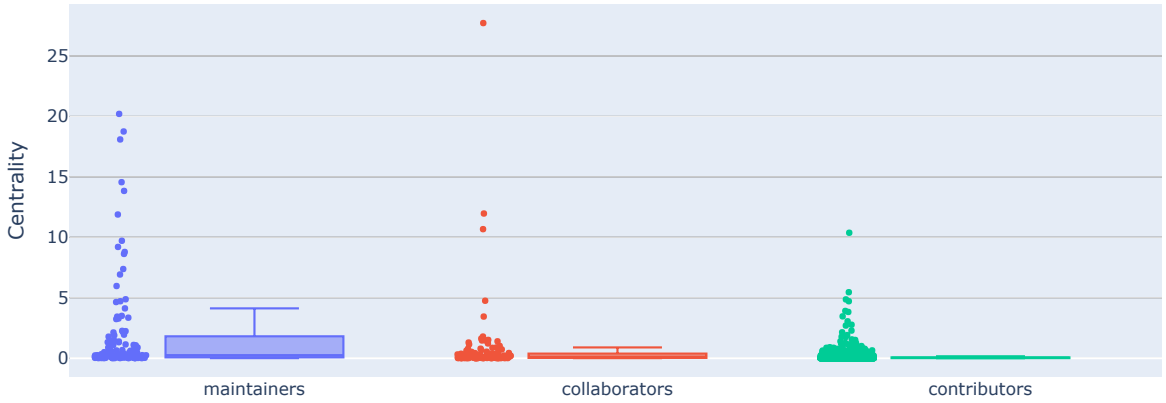

Figure C.6: The distribution of centrality degrees across three roles.

C.1.8 Theme: Reciprocity at Multiple Levels

Based on the analysis of the issue comments, the reciprocity tended to be neutral mostly, since the discussion in the comments primarily focused on technical aspects of the projects, and exchanging some information. Sometimes the sentiment related to reciprocity tends to be slightly positive, particularly when contributors provide constructive feedback to each other, and actively engage in collaboration and communication. Thus, reciprocity was not evident most of the time in the issue comments, while slightly positive sometimes.

C.1.9 Theme: Trust

Based on the trust estimation on the issue comments, the trust was evident and tended to be at a high level in the project community. Contributors actively participate in open and constructive conversations on the issue threads, including addressing concerns, sharing feedback, finding solutions, and so on. All of those indicate the existence of trust among contributors within the project community.

C.1.10 Theme: Universal Accessibility & Availability

The source code is publicly available, and people could access the source code through its GitHub repository. Furthermore, the Contribution Guidelines[4] also indicated such accessibility and availability to the public:

> "*As an open-source project, we welcome and encourage the community to submit patches directly to the project.*"

C.2 Category: Goals

C.2.1 Broad Impacts

No statements that addressed social challenges/concerns are found in the project documentation.

C.2.2 Built-in Product & Quality Orientations

Fig. C.7 shows the number of bugs reported and fixed every month. The number of bugs reported tends to be slightly increasing, which indicates more bugs are identified; the number of bugs fixed tends to be in a consistent trend as bugs reported, around 15%, which also reflects the project quality was improving. Moreover, the project documentation[5] provides ways to generate coverage reports, which could also reflect the project quality. In general, the project quality is improving, but the overall bug-fix ratio is not high enough, and it would be better to provide the test coverage report directly, rather than generating by developers.

[4] `https://docs.zephyrproject.org/latest/contribute/guidelines.html`
[5] `https://docs.zephyrproject.org/latest/develop/test/coverage.html`

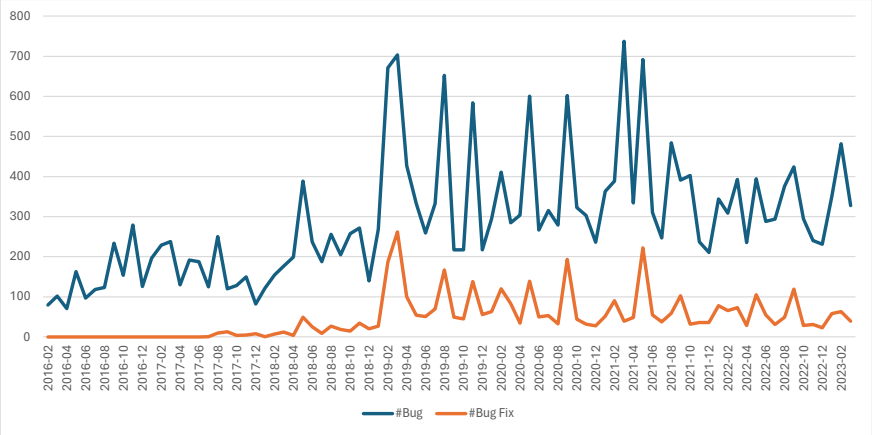

Figure C.7: The number of bugs reported and fixed every month.

C.2.3 Outreach

First, there are scheduled meetup events[6], i.e., developer summits, and local meetup events, to engage with both local and worldwide project communities. Second, the project utilizes several social media platforms, including Facebook, X/Twitter, LinkedIn, YouTube, Discord, etc., to engage with broader and more diverse communities. Thus, the project takes advantage of multiple platforms, and meetup formats to engage with the public.

C.2.4 Sustainability

Fig. C.8, C.9, C.10 show the number of newcomers and non-newcomers who submitted commits, issues, and pull requests every month. First, the number of contributors each month was increasing, reflecting more contributors were active in the project. Second, there was a relatively stable percentage of newcomers contributed every month, which indicated that the project constantly attracted newcomers to join. Thus, it is evident regarding the sustainability in the project community.

[6] `https://www.zephyrproject.org/events/`

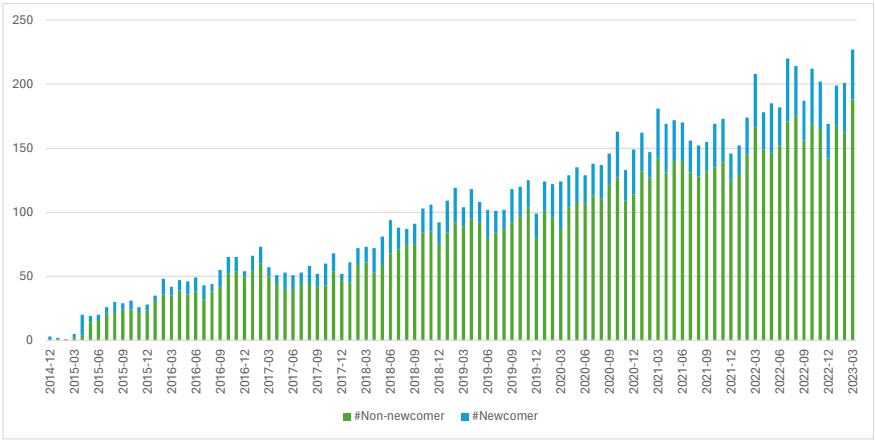

Figure C.8: The number of newcomers and non-newcomers committed to the project every month.

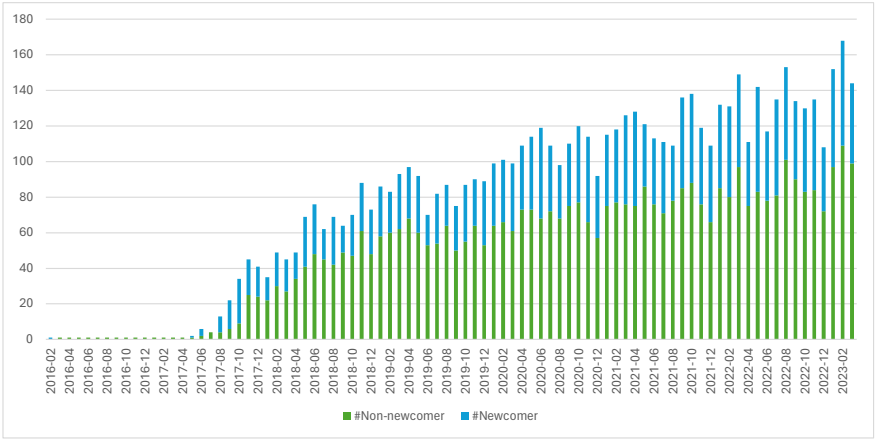

Figure C.9: The number of newcomers and non-newcomers submitted issues every month.

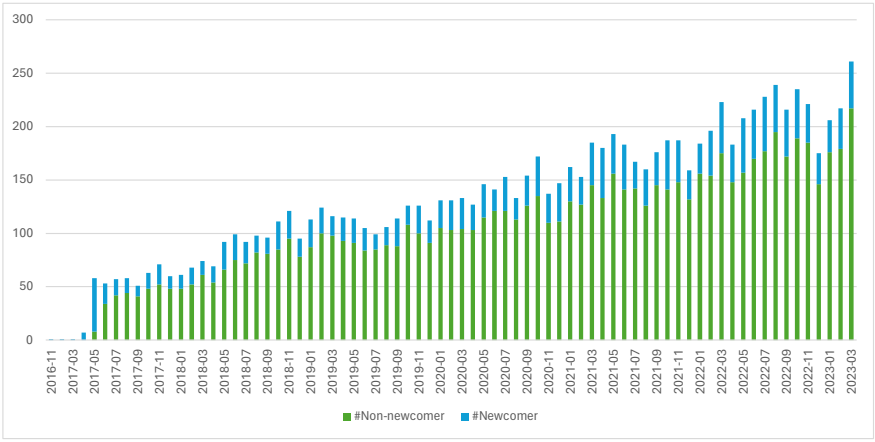

Figure C.10: The number of newcomers and non-newcomers submitted pull requests every month.

C.2.5 Innovation

Fig. C.11 shows the number of issues related to new features and functions, in general, it tends to be stable and slightly increasing, which indicates innovation existed in the project. Moreover, the sentiment towards those new features/functions in the issue comments tends to be mixed with both positive and negative sentiments, which indicates that not all new features or functions are welcomed in the project community, e.g., "No plans to implement this on STM 32 in short/mid-term." Thus, innovation happens in the project, as new features/functions are proposed, however, the general sentiments towards them are not always welcoming.

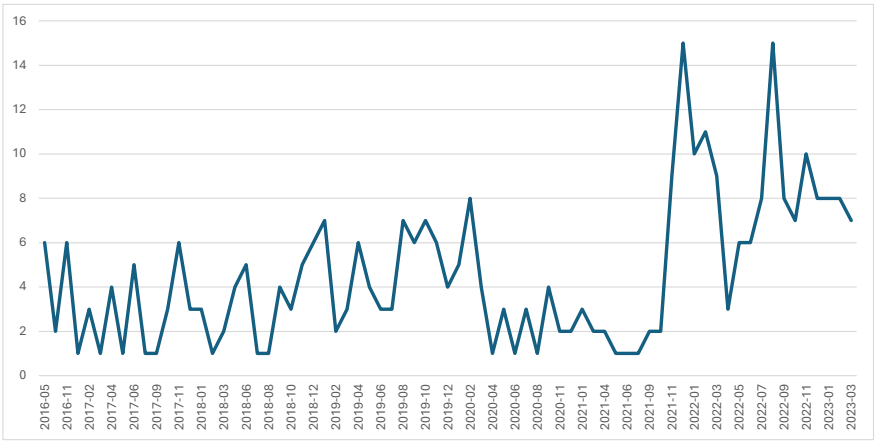

Figure C.11: The number of issues related to new features/functions every month.

C.2.6 Ubiquitous Penetration

According to the official documentation[7], the project has been widely adopted in industrial IoT, asset tracking, wearable, automotive, healthcare, worker safety, etc. There is also a list[8] of products running the project, e.g., Grid Pad 13, Lisios WaterAlarm, Open Collar, Mahi, ProGlove, etc. Thus, the project has been penetrated in various fields and industries, and

[7] `https://zephyrproject.org/zephyr-overview`
[8] `https://zephyrproject.org/products-running-zephyr/`

has been widely adopted in many commercial products.

C.3 Category: Activities

C.3.1 Copyright, Licensing, & Legal Implications

First, the project is under Apache License (Version 2.0)[9], which is a permissive open source license, while some components of the project are under different licenses, i.e., GPLv2. The licensing documentation[10] also provides a brief description of the rights granted to the developers, and further references to help developers better understand those rights. Second, a Developer Certification of Origin (DCO)[11] is required when accepting any contributed code from contributors, to avoid potential patent or copyright issues. Moreover, there is no news related to patent or copyright lawsuits with the project. In general, the project adopts standard open source licensing, and the usage of DCO does not involve copyright transfer, which tends to be simpler and less intimidating for contributors.

C.3.2 Governance & Decision Making

The governance structure of the project is the true meritocracy. There is a Technical Steering Committee (TSC)[12] formed by a group of contributors who were elected from the community. The responsibilities are described as "defining and maintaining the technical vision for the Zephyr project," "serving as the highest technical decision body," and "coordinating cross-community collaboration." The decisions should be made collectively with multiple members of TSC, e.g., "minimal of 2 approvals" is required to merge pull requests.

[9] https://github.com/zephyrproject-rtos/zephyr?tab=Apache-2.0-1-ov-file#readme
[10] https://docs.zephyrproject.org/latest/contribute/guidelines.html#licensing
[11] https://docs.zephyrproject.org/latest/contribute/guidelines.html#dco
[12] https://www.zephyrproject.org/tsc/,

C.3.3 Personal Development

Fig. C.12 shows the distribution of Kendall Tau values for contributors' centrality degree traces, the range of Tau values is from -1.0 to 1.0, with the positive value indicating an increasing trend of the centrality degree trace, while the negative value indicating a decreasing trend. It is evident that all contributors' centrality degrees are increasing, which reflects personal development in the project.

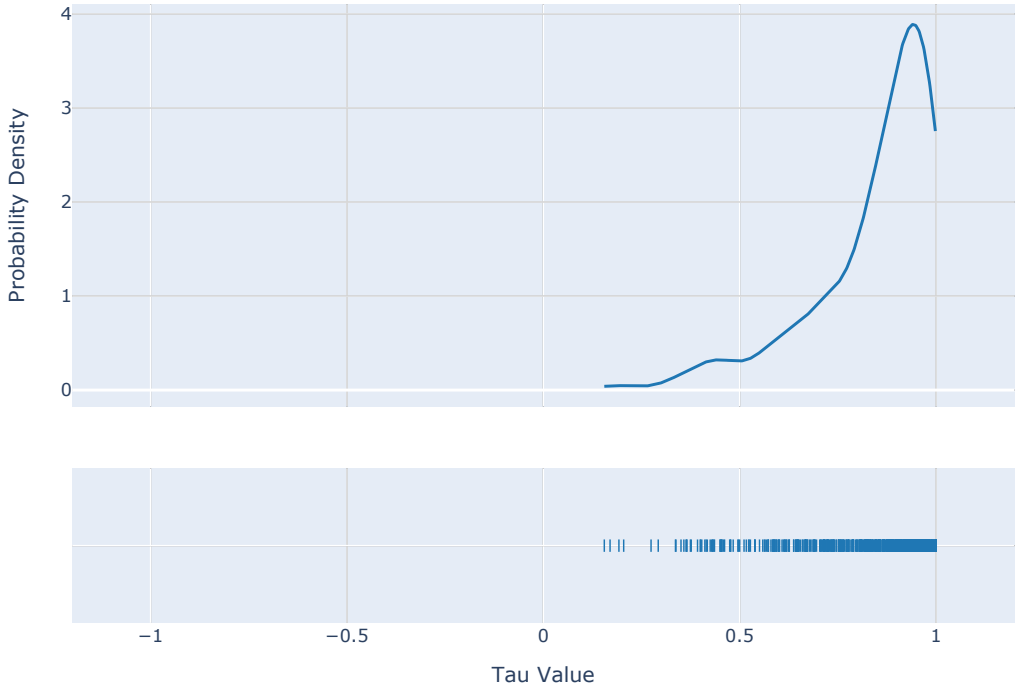

Figure C.12: Distribution of Kendall Tau values for contributors' centrality degree traces.

C.3.4 Social Production

Fig. C.13 shows the number of contributors who participated in the issue discussion, the majority of the issue discussion involved a few contributors (around two to three), and some issues even involved more than ten contributors. Thus, social production is evident in the project.

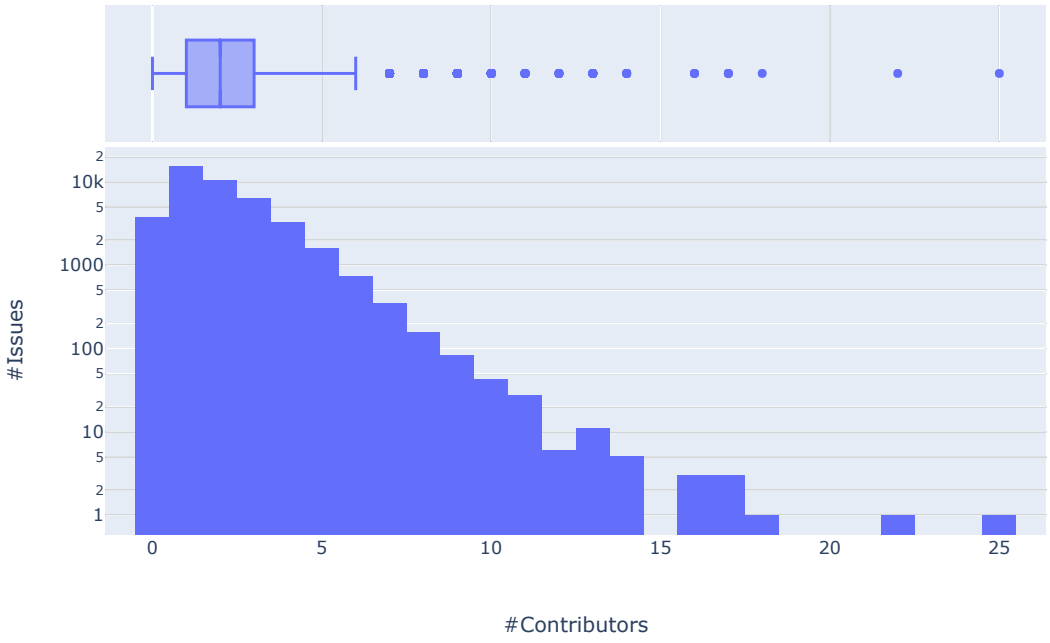

Figure C.13: The number of contributors who participated in issue comments.

C.3.5 Work Organization & Practices

In total, the code commits were submitted from 22 different timezones, as shown in Fig. C.14, which reflects the multi-timezone effort on the project development. Moreover, three different communication channels were utilized, i.e., Discord Server, mailing list, and GitHub Issues, to facilitate various types of collaboration within the project.

C.4 Category: Membership

C.4.1 Individual Identity

Based on the usage of different pronouns in the issue comments, around 36.04% of the pronouns are “I”, “my”, and “me”, while 13.82% are “we”, “our”, and “us”. That reflects

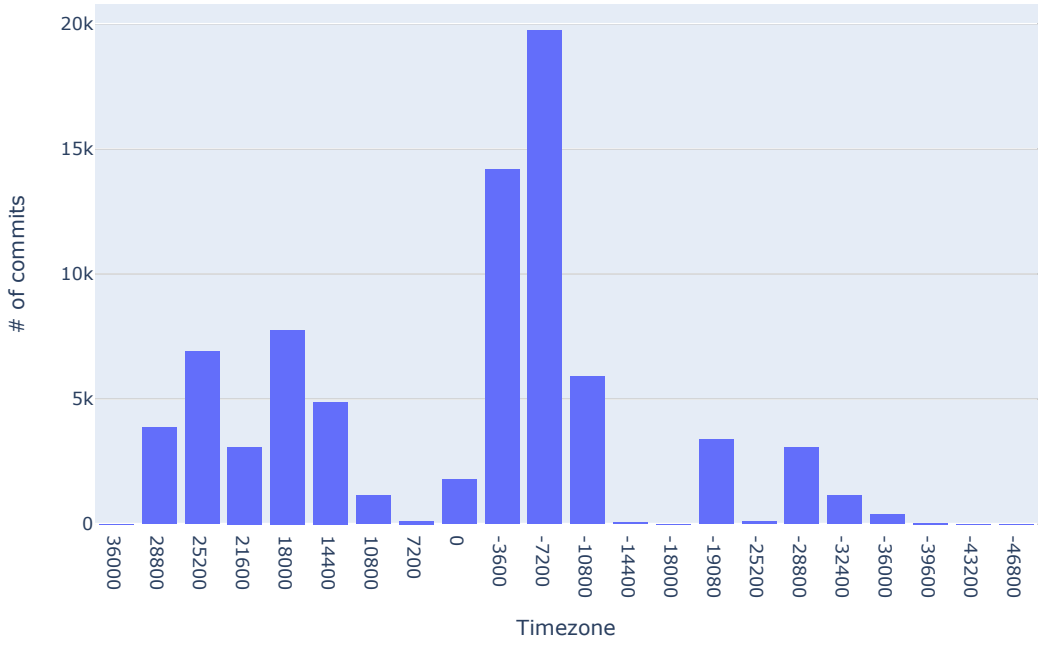

Figure C.14: The distribution of timezones with code commits.

that the contributors tend to address themselves as individuals, and only a limited number of contributors address themselves as part of the community.

C.4.2 Ideological Leadership

The ideological leadership is reflected in the issue comments, and primarily focuses on technical aspects, e.g., a collaborative approach to decision-making, several contributors and maintainers participated in the discussion regarding new feature proposals, design choices, implementation, etc.

C.4.3 Interpersonal Relationships

In general, the interpersonal relationships reflected in the issue comments tend to be positive, particularly professional, collaborative sentiments. The interactions among contributors mostly focused on resolving technical issues, and improving project development, professionally and constructively.

C.4.4 Member Hierarchy & Roles

The project community follows a hierarchical structure, and three roles are explicitly defined in the project documentation[13], i.e., Maintainers, Collaborators, and Contributors, as well as their rights and responsibilities. For example, "A Collaborator is a Contributor who is also responsible for the maintenance of Zephyr source code," and the responsibilities include "set goals for the short and medium terms," "participate in the feature development process," "review relevant code changes within reasonable time," and so on. Thus, the hierarchical structure and roles are clearly defined and described in the project documentation.

[13] `https://docs.zephyrproject.org/latest/project/project_roles.html`

C.5 Category: Resources

C.5.1 Dealing with Barriers & Restrictions

The project documentation provides detailed guidelines for contributors, including Contribution Guidelines[14], Contributor Expectations[15], Coding Guidelines[16], etc. Those documents cover a wide range of related topics, such as understanding source tree structure, and following contribution workflow, to help contributors better understand the project and development process, to overcome potential barriers or restrictions.

C.5.2 Incentives, Financing, & Funding

There is a membership mechanism, "Become a Member[17]," as a way to support the project. Membership fees are required for certain types of members, i.e., Platinum and Silver members. Moreover, the project also offers some career opportunities[18] within the project ecosystem and community. However, there is no donation mechanism for individuals to have small or one-time donations to support the project.

C.5.3 Knowledge & Expertise

In general, the project provides detailed documentation[19] to cover various aspects of the project. The documentation is prepared for both developers and users of the project,

[14] https://docs.zephyrproject.org/latest/contribute/guidelines.html#contribute-guidelines

[15] https://docs.zephyrproject.org/latest/contribute/contributor_expectations.html#contributor-expectations

[16] https://docs.zephyrproject.org/latest/contribute/coding_guidelines/index.html#coding-guidelines

[17] https://www.zephyrproject.org/join/

[18] https://www.zephyrproject.org/careers/

[19] https://docs.zephyrproject.org/latest/

e.g., Contribution Guidelines and Kernel documentation help developers to understand the project and facilitate development; Samples and Demos, OS Services documents help users learn how to use and run the project on supported boards and devices. Moreover, at the end of every documentation page, there is a section to indicate the last updated time of the page, and encourage readers to help keep the documentation update-to-date, by reporting errors or improving the content.

C.5.4 Supportive Facilities & Mechanisms

First, a set of tools/toolchains[20] is adopted to facilitate the project development. For example, CLion is a C/C++ IDE for debugging, Coccinelle is adopted for kernel development, as well as project SDK, ARM Compiler, etc. Second, GitHub Issue list, Discord, and mailing list are utilized as communication channels regarding the project development. However, there is no mentoring or similar mechanisms adopted in the project community.

C.6 Category: Position/Group Relations

C.6.1 Interaction with Emerging Technologies

According to the project documentation, and official website[21], the project mostly interacts with AI, Robotics, Ultra-Wideband RTLS technologies.

[20] `https://docs.zephyrproject.org/latest/develop/tools/index.html`, `https://docs.zephyrproject.org/latest/develop/toolchains/index.html`

[21] `https://zephyrproject.org/zephyr-rtos-emerging-technologies/`

C.6.2 Interaction with Commercial Software Development

First, some commercial companies are involved as platinum/silver members in the project community[22], e.g., Google, Intel, Meta, and T-Mobile. Their employees also actively participate in the project development. Then, the project has been widely adopted in many commercial products, such as tablets, and wearable devices.

C.6.3 Market & Users

The project primarily focuses on the IoT, embedded system markets[23], for smart, wireless devices. The users of the projects are mostly the developers, they would deploy the project to their products, and then deliver them to their end users.

[22] https://www.zephyrproject.org/project-members/

[23] https://www.zephyrproject.org/wp-content/uploads/sites/38/2024/03/zephyr_datasheet_0328240-nobleed.pdf