# A COMPARATIVE STUDY OF THE NPM, PYPI, MAVEN, AND RUBYGEMS OPEN-SOURCE COMMUNITIES

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Saurav Gupta

June 2024

COMMITTEE MEMBERSHIP

TITLE: A comparative study of the NPM, PyPI, Maven, and RubyGems open-source communities

AUTHOR: Saurav Gupta

DATE SUBMITTED: June 2024

COMMITTEE CHAIR: Ayaan M. Kazerouni, Ph.D.
Assistant Professor of Computer Science

COMMITTEE MEMBER: Stephen R. Beard, Ph.D.
Assistant Professor of Computer Science

COMMITTEE MEMBER: John Clements, Ph.D.
Professor of Computer Science

ABSTRACT

A comparative study of the NPM, PyPI, Maven, and RubyGems open-source communities

Saurav Gupta

Open-source software (OSS) ecosystems, defined as environments composed of package managers and programming languages (e.g., NPM for JavaScript), are essential for software development and foster collaboration and innovation. Although their significance is acknowledged, understanding what makes OSS communities healthy and sustainable requires further exploration. This thesis quantitatively assesses the health of OSS projects and communities within the NPM, PyPI, Maven, and RubyGems ecosystems. We explore five research questions addressing project standards, community responsiveness, contribution distribution, contributor retention, and newcomer integration strategies. Our analysis shows varied documentation practices, insider engagement levels, and contribution patterns. Our findings highlight both strengths and different areas for improvement across ecosystems. For example, RubyGems excels in the adoption of project documentation and exhibits the most even distribution of contributions among all contributors, including highly active contributors. and a very responsive community, but it needs to improve contribution retention and attract newcomers to the projects. Meanwhile, NPM and Maven show a trend toward getting new contributors, characterized by a high ratio of individual contributions. They need to better adopt a code of conduct, pull request templates, and increase the number of active contributors in a project. This thesis offers insights to developers and maintainers on how to strengthen ecosystems and support vibrant communities effectively.

# ACKNOWLEDGMENTS

I am writing to express my sincere gratitude to Dr. Ayaan M. Kazerouni, my thesis advisor, for his immense contribution in providing me with valuable insights, guidance, and knowledge throughout the research process. His expertise and support have played a crucial role in shaping this thesis and overcoming obstacles.

I want to sincerely thank my committee members, Dr. Stephen R. Beard and Dr. John Clements, for dedicating their invaluable time to being on my committee and supporting me on my research journey.

I want to thank my parents for their unwavering support and the opportunities they have provided me. Their encouragement and belief in my abilities have been a constant source of motivation.

I thank Andrew Guenther for uploading this template on GitHub.

Lastly, I would like to thank all my friends and well-wishers who have offered their encouragement and understanding during this undertaking.

This thesis would not have been possible without these individuals' collective support and guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# Chapter 1

# INTRODUCTION

The evolution of Open-source software (OSS) ecosystems has been pivotal in shaping the modern software landscape. In this thesis, following Davis et al. [1], we define an "ecosystem" as a collection of software projects that are developed and co-evolve in the same programming language and environment. We refer to ecosystems by the names of their default package registries, such as "NPM" [2] for JavaScript, "PyPI" [3] for Python, "Maven" [4] for Java, and "RubyGems" [5] for Ruby.

The long-term sustainability of OSS projects (and the ecosystems in which they live) can be affected by factors like community interaction dynamics and recruitment and retention of contributors [6, 7, 8, 9]. This thesis presents a comparative analysis of a number of OSS community metrics for the NPM, PyPI, Maven, and RubyGems ecosystems. We studied the GitHub repositories and community interactions in the top 1,000 most downloaded packages in each ecosystem. Our findings reveal diverse practices and highlight strengths and areas for improvement. For instance, the RubyGems community excels in the adoption of documentation (e.g., having a high presence of Contributing guidelines, Code of Conduct, and Pull Request Template Files in repositories) but struggles with contributor retention, while the NPM and Maven ecosystems are good at attracting new contributors but need to improve in certain other aspects (like the adoption of documentation and project insiders' engagement with external contributors).

We answer five core research questions, organized into three themes: project metadata, dynamics of community engagement, and recruitment and retention of contributors.

**Project Metadata.**

***RQ1:*** *How prevalent are community guideline documents in repositories in the four ecosystems?*

We checked repositories for the presence of key files that provide project guidelines. These file naming conventions and their roles are standardized on GitHub.[1] Specifically,

- README.md, which provides an overview of the project and usage instructions.
- CONTRIBUTING.md, which details guidelines for contributing to the project.
- PULL_REQUEST_TEMPLATE.md, which offers a template to standardize pull request submissions.
- CODE_OF_CONDUCT.md, which sets expected behavior norms for contributors.
- LICENSE, which specifies the legal terms for software use and distribution.

**Community Responsiveness and Participation.**

***RQ2:*** *How do project insiders respond to issues opened by outsiders in the four ecosystems?* Active engagement and support are good indicators of the overall health and responsiveness of the OSS project. We studied the rate and speed of response by project maintainers ("insiders") to issues opened by external users ("outsiders").

[1]See the Building Communites documentation page at https://docs.github.com/en/communities.

***RQ3:*** *How is the distribution of contributions among project contributors characterized within four ecosystems?* The distribution of contributions towards projects highlights the inclusivity and collaborative breadth within each project's ecosystem. We analyzed the number of commits by contributors to determine the distribution of contributions between highly active members and lower-contributing members.

**Contributor Attraction and Retention.**

***RQ4:*** *What is the contributor retention rate within four ecosystems?* Contributor involvement provides community effectiveness in terms of member retention and the sustainability of their engagement in OSS projects. We studied member engagement in terms of its retention rates by analyzing the frequency of subsequent pull requests within the project of an ecosystem.

***RQ5:*** *What percentage of repositories across four ecosystems feature a "good first issue" label, and what proportion of contributions to projects within each ecosystem come from newcomers?* The presence of a "good first issue" label signals a project's openness to incorporating support and motivates new contributors. This study evaluates the community's welcomeness to newcomers by analyzing the prevalence of GFI labels across ecosystems. It also seeks to investigate the effectiveness of the GFI labels through the percentage of newcomers contributing to OSS projects.

*Summary of findings.* This work compares the NPM, PyPI, Maven, and RubyGems ecosystems across various quantitative metrics, including repository metadata, insider engagement, contribution disparity, contributor retention, GFI welcomeness, and newcomer contribution. Our findings have revealed distinct patterns and correlations within each ecosystem:

- NPM and Maven ecosystems exhibit trends toward attracting more newcomers to projects.
- PyPI demonstrates strength across all metrics but faces challenges in contribution disparity among contributors.
- RubyGems showcases a strong community but requires an influx of newcomers.
- The presence of "Good First Issue" labels correlates with increased newcomer contributions across all ecosystems.

Thesis highlights the patterns in the form of strengths and areas for improvement in each ecosystem, along with several avenues for future research. Firstly, further investigation is needed to understand the underlying reasons behind observed patterns. While thesis focused on quantitative analysis, qualitative research could provide valuable insights into the motivations driving ecosystem behaviors. Additionally, future studies could extend our analysis to other OSS ecosystems, encompassing a broader range of variables and contexts. By continuing to explore these areas, we can deepen our understanding of OSS dynamics and contribute to the advancement of sustainable and thriving open-source communities and health.

This thesis's outline is as follows. We examine the evolution and significance of OSS ecosystems (Chapter 2: Background); Selection of Ecosystems and Data collection (Chapter 3: Study Overview); Methods and findings for each research theme are presented in Chapter 4 (Project Metadata), 5 (Community Responsiveness and Participation), and 6 (Contributor Attraction and Retention). The discussion (Chapter 7) reflects on our findings, and Chapter 8 outlines future work.

Chapter 2

# BACKGROUND

## 2.1 Software Ecosystem

The landscape of open-source software (OSS) has evolved significantly, transforming from a niche movement into a mainstream force within the software industry. The emergence of software package managers like npm, PyPI, Maven, and RubyGems has been pivotal in this transformation, making them indispensable for modern software development practices by providing an extensive repository of reusable code, libraries, and tools.

OSS began as a collaborative venture among enthusiasts and has become a fundamental aspect of the global software infrastructure. This evolution is marked by critical milestones, such as the introduction of the GNU General Public License (GPL) in the 1980s and the emergence of significant projects like Linux and Apache [10]. The proliferation of OSS platforms reflects a shift in software development, moving from proprietary to open and collaborative models, significantly impacting development practices, economic models, and technological innovation [11]. The shift has enabled rapid prototyping, community-driven innovation, and reduced barriers to entry for startups, thereby democratizing software development. The impact extends to how businesses leverage OSS for competitive advantage, integrating open-source components into their proprietary solutions to enhance agility and innovation for the software ecosystem and architectural health [12].

The success of an OSS project is intricately linked to its community. OSS communities, composed of volunteers, demonstrate diverse structures and governance models.

Contributors' motivations vary, ranging from a passion for technology and a desire for peer recognition to collaborative learning and professional development opportunities [13]. The diversity within OSS communities often reflects a broader spectrum of skills, perspectives, and collaborative dynamics, enriching the development process [13, 14].

Managing OSS platforms presents unique challenges. Sustaining active participation, ensuring quality control, and addressing security vulnerabilities are constant concerns. The disparity in contributions and the need for effective project governance also pose significant hurdles [14]. The effectiveness of governance models in balancing openness with the need for structure and leadership is critical to the sustainability and health of these projects [15].

OSS is instrumental in driving innovation in the software industry, facilitating a culture of open collaboration that has been key to several technological breakthroughs. This culture extends beyond the industry, impacting education and promoting global digital equity [16, 17]. Thus, studying OSS ecosystems is not merely about understanding software development practices but exploring collaborative innovation, community dynamics, and the democratization of technology, providing crucial insights into the future of software development and the global tech community at large [16].

## 2.2 Project Metadata files in OSS

Project metadata files like the README.md, CONTRIBUTING.md, LICENSE, CODE_OF_CONDUCT.md, and PULL_REQUEST_TEMPLATE.md are pivotal in shaping collaborative development practices.

The **README.md file** is often the first point of contact for anyone encountering an open-source project. It provides a window into the software, offering essential information about its purpose, usage, and installation processes [18]. The impact of comprehensive README files on project popularity cannot be overstated; clear documentation is closely linked to the success of open-source projects. Studies have indicated that well-documented projects tend to attract more contributors and users, enhancing their visibility and viability within the community [19].

The **code of conduct** in OSS projects is a vital element that outlines expected behaviors and norms within the community. It serves as an interaction guideline, promoting a respectful and inclusive environment. This aspect of the presence of an OSS health file is integral to fostering a healthy community dynamic and is essential for collaborative software development.

**Licensing** in OSS is another crucial component. It specifies the terms under which the software can be used, modified, and distributed. Different licenses offer varying degrees of freedom and restriction, influencing how the software can be integrated into other projects. Licensing decisions can significantly affect the adoption and spread of the software [20].

Using **pull request templates** and detailed **descriptions** enhances the comprehensibility and usability of software repositories [21]. Such practices ensure that contributions are consistent and well-documented, facilitating more accessible code reviews and collaborations. They contribute significantly to the overall quality and maintainability of the software [22].

## 2.3 The Dynamics of Community Interaction in OSS

This section explores the role of core members of OSS project development teams ("project insiders"), focusing on their influence on community engagement.

The immediacy and quality of interactions are pivotal in shaping user experience and encouraging active participation. Quick, constructive responses to queries and suggestions are essential, whether from beginners seeking help or experts offering advanced insights. Such interactions create a welcoming atmosphere that encourages users to transition from passive observers to active contributors. This shift is crucial for the ongoing growth and improvement of the software, as noted in the studies of Bettenburg et al. and Jensen et al. [23, 24]. The OSS Insider's role in facilitating high-quality interactions is invaluable in fostering a vibrant community.

The strength of the OSS model lies in its diverse and global user base, which brings a wide range of perspectives and skills [25]. The OSS Insider's ability to effectively engage with this diverse community is crucial. They must navigate cultural and linguistic differences with inclusive and adaptable communication strategies. Such an inclusive approach not only enhances the problem-solving process but also strengthens the collaborative foundation of the OSS community. It ensures that the software stays relevant and resilient in the face of rapid technological changes, as explored by Marin et al. and Mendez-Duron et al. [26, 27].

Users frequently contribute to the software's evolution by reporting bugs, suggesting new features, or even directly offering code solutions. A participatory approach ensures that OSS projects are continually refined and remain aligned with the needs of their user base. The OSS Insider is instrumental in facilitating and nurturing a feed-

back loop. This role is essential in maintaining the project's relevance and operational effectiveness, as highlighted in studies by Daniel et al. [28].

The role of the OSS Insider extends beyond technical expertise. It involves nurturing a vibrant, inclusive, and collaborative community. Such efforts are crucial in ensuring that OSS projects achieve technical excellence and become successful communal software development models.

Given the pivotal role of community interactions in shaping the trajectory of OSS projects, it is imperative to explore the multifaceted nature of participation within ecosystems. This discussion will highlight the various roles and contributions that individuals undertake, underscoring their collective impact on the advancement and sustainability of open-source initiatives.

## 2.4 OSS Participation

This section explores how diverse forms of participation within OSS communities impact the overall success of OSS projects, both in terms of community engagement and market achievement.

Contributors to OSS projects embody various roles, ranging from programmers to designers, testers, documenters, and end-users providing feedback. Such a range of roles, as highlighted by Hars and Ou and Daniel et al., contributes to the holistic development of OSS projects, fostering innovation and ensuring the software meets a broad spectrum of user needs [29, 30]. An inclusive environment leads to robust, versatile software products that align closely with user requirements. A diverse composition within OSS communities, encompassing different cultural, educational, and professional backgrounds, enhances creative problem-solving and software adaptabil-

ity. According to Xu et al. and MR Martínez-Torres, such diversity is pivotal in developing user-friendly software that caters to a diverse market [31, 32].

Market success is closely linked to the nature and extent of community engagement. Diverse and active communities contribute to a positive reputation, increased reliability, and a broader user base, which are essential for achieving market success, as discussed by Daniel et al. and Shah [30, 33].

Recognizing the spectrum of participation that enriches OSS, the narrative advances to the integration and engagement of newcomers and external contributors. Their inclusion is crucial for bringing fresh ideas and sustaining the vibrancy of OSS communities, necessitating strategies that promote their active involvement and ensure the ongoing evolution of open-source projects.

## 2.5 Newcomer to the OSS and First-timer Engagement

This section delves into the critical role of newcomers in OSS projects. It highlights the significance of their contributions and challenges and explores strategies for effectively attracting and onboarding first-time participants into projects.

Newcomers, particularly first-timers, are the lifeblood of OSS communities. Successful integration of newcomers leads to a continuous replenishment of skills, ideas, and perspectives, which is crucial for the long-term success of OSS projects [34, 35]. A newcomer's early interactions with a project are critical as they learn about the project's culture, collaboration practices, and technical details. This engagement fosters a culture of inclusivity and constant learning, which are fundamental values of the OSS philosophy.

Despite their potential contributions, newcomers often encounter significant barriers. Barriers range from technical challenges, such as understanding complex codebases, to social obstacles, like integrating into an established community with its unique culture and norms. Overcoming such challenges is vital for ensuring diversity, which fuels innovation and growth in the OSS ecosystem [36].

One effective strategy for integrating newcomers is through "good first issue" labeled issues and pull requests. Such tasks are introductory tasks tailored to help first-timers understand the project's codebase and workflow. Adopting this method not only eases the entry of newcomers but also demonstrates a commitment to building an inclusive and nurturing OSS community. Initiatives like these are essential for lowering the barriers to entry and ensuring that the community remains vibrant and innovative [37].

# Chapter 3

# STUDY OVERVIEW

This thesis aims to quantitatively measure the health of open-source software (OSS) projects and communities at an ecosystem scale. We achieve this through analysis of metrics collected from repository mining across four major OSS ecosystems: NPM (Node Package Manager), PyPI (Python Package Index), Maven (for Java), and RubyGems (for Ruby). This study captures a snapshot of the current health of these OSS communities. In this chapter, we discuss the motivation behind selecting these four ecosystems and outline the data collection and analysis basis for further research, guided by our research questions.

## 3.1 Selection of Ecosystems

Recent surveys and studies highlight the widespread popularity and usage of JavaScript/Node.js, Python, Java, and Ruby (and their respective package registries). The StackOverflow Developer Survey of 2023 [38] consistently shows JavaScript, Python, and Java among the most popular programming languages used by developers globally with the following distribution:

- JavaScript: 63.61%
- Python: 49.28%
- Java: 30.55%
- Ruby: 6.23%

Particularly, JavaScript has maintained its position as the most used programming language, highlighting npm's relevance in current development practices [38]. Python, extensively applied in diverse fields, remains highly favored for its versatility and ease of use. Java's long-standing presence continues to be significant, reflecting its steady demand across various sectors. While Ruby may not rank as highly in recent surveys, its inclusion is justified by its dedicated community and specific applications in web development. By focusing on this diverse quartet, this study encompasses a broad spectrum of OSS activities.

## 3.2 Data Collection and Analysis

A comprehensive investigation of the top 1000 GitHub repositories across npm, PyPI, Maven, and RubyGems formed the basis of this study. We selected the most downloaded packages from each package manager/registry. These 4000 data points, representing GitHub repositories, formed the basis for our further explorations based on the research questions (see Figure 3.1).

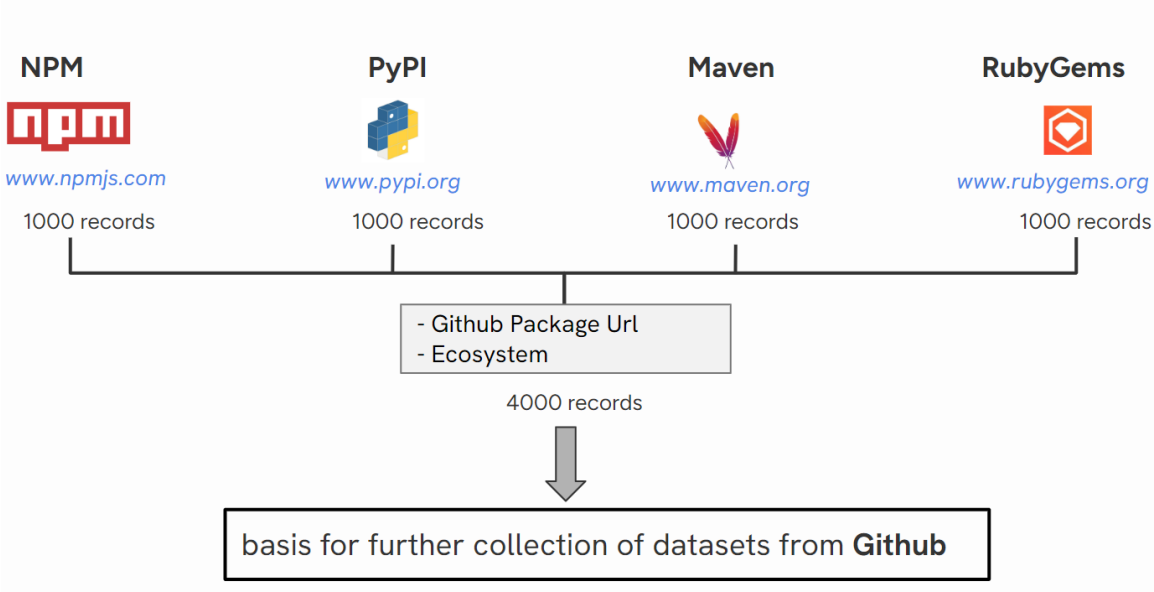


**Figure 3.1: Representation of data collection across four ecosystems**

We utilized JavaScript to write scripts for fetching data from the GitHub open APIs using REST and GraphQL. The following example code snippet was used to fetch data from the package registries.

```
const axios = require("axios");
const fs = require("fs");

async function fetchData(apiUrl, ecosystem) {
  let allRecords = [];

  for (let offset = 0; offset < 1000; offset += 250) {
    try {
      const response = await axios.get(
                apiUrl,
                { params: { from: offset } }
      );
      const records = response.data.objects;
      allRecords = allRecords.concat(records);
    } catch (error) {
      console.error("Error fetching data:", error);
      break;
    }
  }

  const csvData = allRecords.map((record) => ({
    name: record.package.name,
    githubURL: record.package.links.repository,
    ecosystem: ecosystem
  }));
}

// Example API URLs of the registries with passed ecosystem arg
fetchData("https://registry.npmjs.com/-/v1/search", "npm");
fetchData("https://pypi.org/pypi", "pypi");
fetchData("https://search.maven.org", "maven");
fetchData("https://rubygems.org/api/v1/search.json", "rubygems");
```

Table 3.1 depicts 20 example packages that we studied, 5 from each ecosystem.

We used the Python libraries NumPy and Pandas [39, 40] for data analysis and Seaborn [41] for data visualization.

**Table 3.1: Example of Combined Dataset from 4 Package Registries**

| Name | Github URL | Ecosystem |
|---|---|---|
| debug | https://github.com/debug-js/debug | npm |
| fs-extra | https://github.com/jprichardson/node-fs-extra | npm |
| glob | https://github.com/isaacs/node-glob | npm |
| react | https://github.com/facebook/react | npm |
| webpack | https://github.com/webpack/webpack | npm |
| requests | https://github.com/psf/requests | pypi |
| certifi | https://github.com/certifi/python-certifi | pypi |
| PyYAML | https://github.com/yaml/pyyaml | pypi |
| s3transfer | https://github.com/boto/s3transfer | pypi |
| pyjwt | https://github.com/jpadilla/pyjwt | pypi |
| java-annotations | https://github.com/JetBrains/java-annotations | maven |
| shiro | https://github.com/apache/shiro | maven |
| javafx-controls | https://github.com/openjdk/jfx | maven |
| java-jwt | https://github.com/auth0/java-jwt | maven |
| volley | https://github.com/google/volley | maven |
| json | https://github.com/flori/json | rubygems |
| rack-test | https://github.com/rack/rack-test | rubygems |
| ruby-jwt | https://github.com/jwt/ruby-jwt | rubygems |
| rails-dom-testing | https://github.com/rails/rails-dom-testing | rubygems |
| ruby-progressbar | https://github.com/jfelchner/ruby-progressbar | rubygems |

The following chapters present the methods and results for our research questions.

# Chapter 4

# PROJECT METADATA

In this chapter, we examine the presence of project metadata files within open-source software development across four ecosystems. Our focus lies on assessing the prevalence and adoption of key documents such as the README, CODE_OF_CONDUCT, and CONTRIBUTING files. These documents play a critical role in establishing collaborative, inclusive, and respectful communities within the open-source landscape. We aim to understand the extent to which these GitHub-recommended files are adopted within open-source projects. The methodology involves analyzing data from 4,000 GitHub repositories and examining the presence of specified files and keys. Our findings offer insights into the community health metrics of major open-source ecosystems, revealing trends in documentation practices and areas for potential improvement.

## 4.1 RQ1.How prevalent are community guideline documents in repositories in the four ecosystems?

GitHub Metadata, including files like README, CODE_OF_CONDUCT, and CONTRIBUTING, is crucial for managing open-source software projects. These documents create a foundation for building collaborative, inclusive, and respectful communities [6]. Given the rapid expansion of the open-source ecosystem, understanding the role and influence of these metadata is increasingly important in software development research. This research question seeks to empirically examine how widely these

GitHub-recommended files and guidelines are adopted in open-source communities and their significance.

## 4.2 Method

In the analysis, data was sourced from the 4,000 repositories, where each repository underwent examination for the presence of 6 files or keys (See Figure 4.1). Below are the files/keys used in the analysis along with their descriptions:

- README.md: It contains important information about the project, including how to use it, install it, and contribute to it, serving as the entry point for users and developers to understand and enhance project visibility and accessibility.
- Description: It provides a brief overview of the project, helping users quickly understand its purpose and relevance.
- PULL_REQUEST_TEMPLATES.md: It guides contributors in submitting meaningful and well-structured pull requests, streamlining the review process, and improving collaboration within the project.
- CONTRIBUTING.md: It outlines the project's contribution process, including instructions for reporting issues, submitting code changes, participating in discussions, and encouraging community engagement and external contributions.
- LICENSE: It specifies the terms and conditions under which the project's code and resources can be used, modified, and distributed. Including a license ensures legal clarity and protection for both project maintainers and users.
- CODE_OF_CONDUCT.md: It establishes behavioral expectations and guidelines for project participants, promotes a respectful and inclusive community

environment, and enforces positive interactions while mitigating potential conflicts or harassment incidents [42].

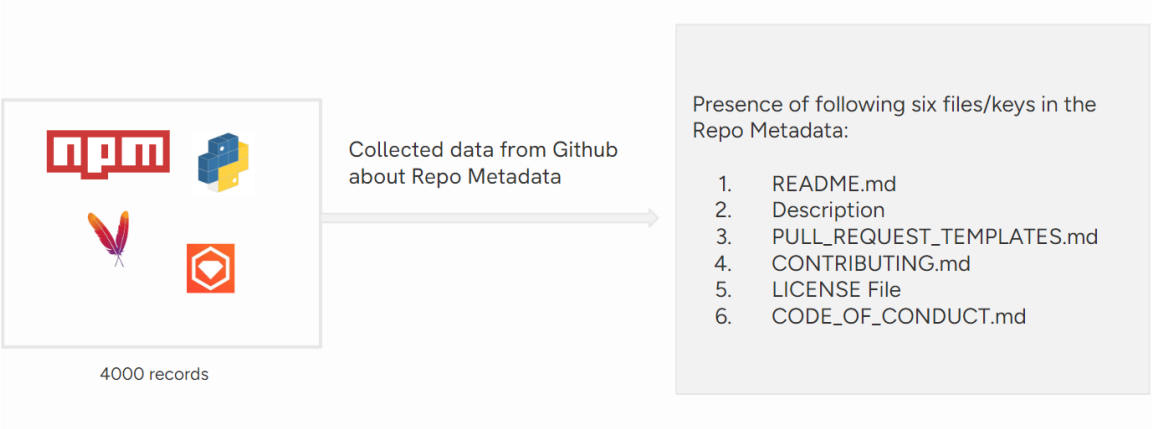


**Figure 4.1: Collecting project metadata from GitHub repositories for each ecosystem**

The dataset was categorized by ecosystem, with a focus on the percentage of repositories in each that included various specified files. This organization facilitates a straightforward comparative analysis across different ecosystems.

## 4.3 Results

**README File Inclusion**: All ecosystems exhibited a nearly universal presence of README files in the 1000 most popular packages. This was largely expected and aligns with conventional norms in open-source projects. NPM and PyPi each showcased a full 100%, followed closely by RubyGems at 99.88% and Maven at 98.85%. The inclusion of this data, while seemingly obvious, is vital for the sake of completeness and underscores the README file's critical role as an entry point for community members and contributors [18] (See Figure 4.2).

**Description Inclusion**: Nearly all repositories across ecosystems had included descriptions, with NPM at 99.18%, PyPI at 95.07%, RubyGems at 97.65%, and Maven

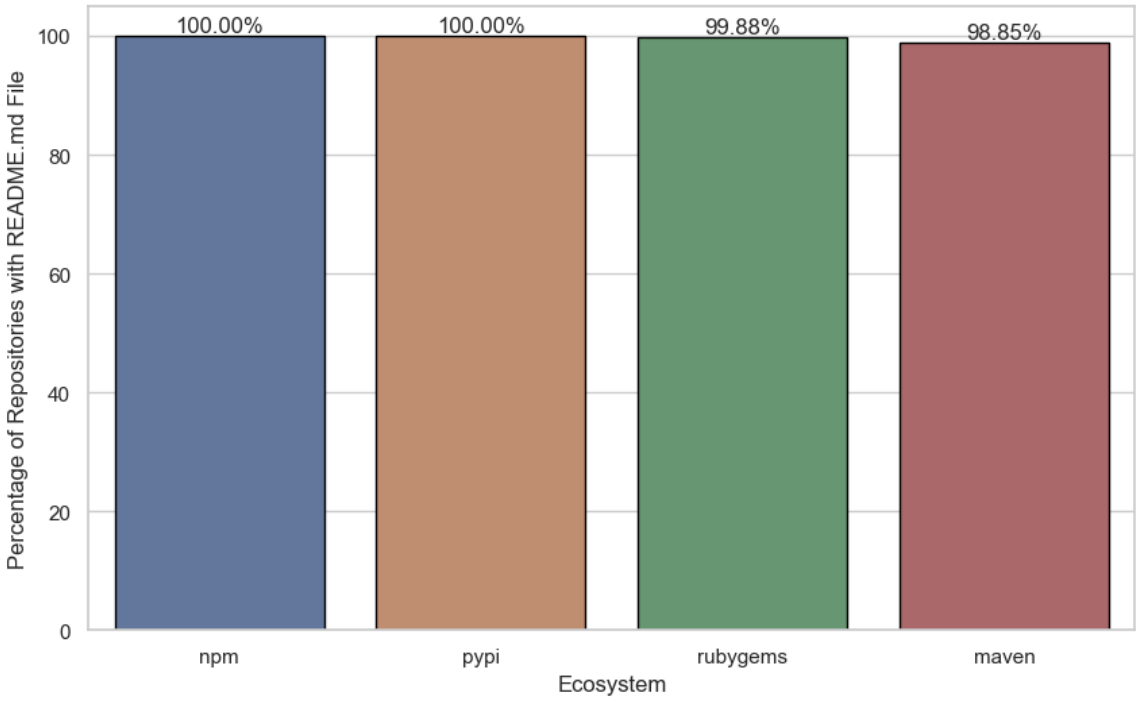


**Figure 4.2: Percentage of repositories in each ecosystem that had README files**

slightly lower at 95.61%. This suggests that most repositories adequately inform potential contributors and users about their projects (See Figure 4.3).

**Pull Request Template Availability**: In the category of Pull Request Template Availability, RubyGems and PyPI led with 37.18% and 35.96% of their repositories featuring templates for pull requests, respectively. NPM followed at 25.43%, with Maven having the lowest at 21.55%. This variation in template availability across ecosystems suggests an interesting area for further exploration, particularly regarding the impact of pull request templates on the quality and efficiency of contributions (See Figure 4.4).

**Presence of Contributing.md**: Presence of contributing.md was most prevalent in RubyGems at 57.41% and PyPI at 56.16%, followed by NPM at 50.97% and Maven

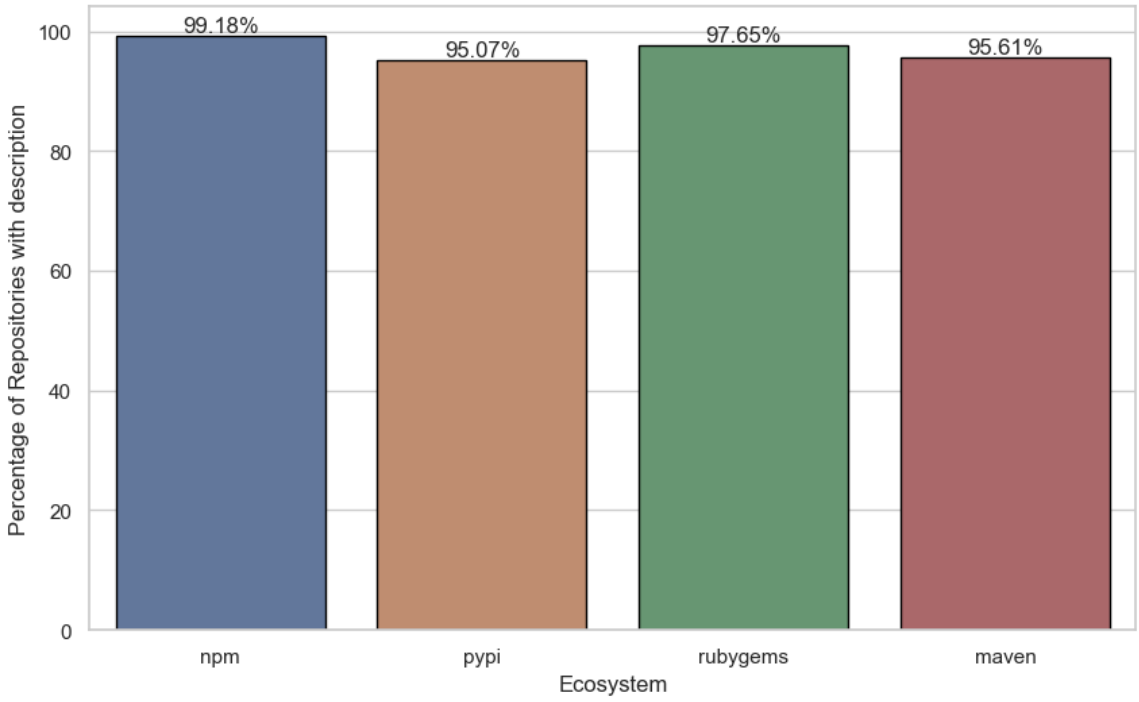


**Figure 4.3: Percentage of Description Inclusion across Ecosystems**

at 45.61%. This shows a moderate to high commitment to guiding new contributors across ecosystems (See Figure 4.5).

**License Inclusion**: Nearly all repositories included a license, with PyPI at 99.26% and RubyGems at 97.06%, reflecting a strong adherence to legal best practices. NPM followed closely at 97.34%, and Maven had the lowest at 89.64% (See Figure 4.6).

**Code of Conduct Adoption**: RubyGems repositories had the highest presence of a CODE_OF_CONDUCT.md file at 48.00%, followed by PyPI at 43.35%, NPM at 38.30%, and Maven at 35.77% (See Figure 4.7). However, it's notable that none of the ecosystems reached a presence of 50% or more for the inclusion of a code of conduct file. This finding underscores a collective need for improvement across all ecosystems regarding the adoption of a code of conduct.

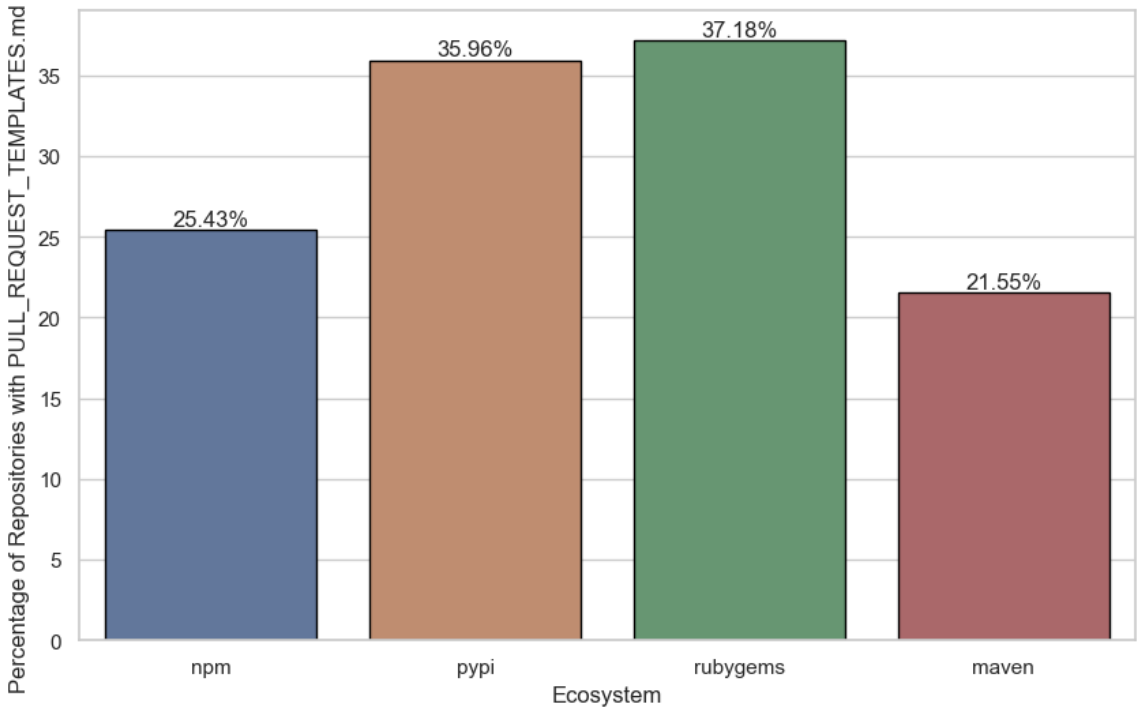


**Figure 4.4: Percentage of Pull Request Template Availability across Ecosystems**

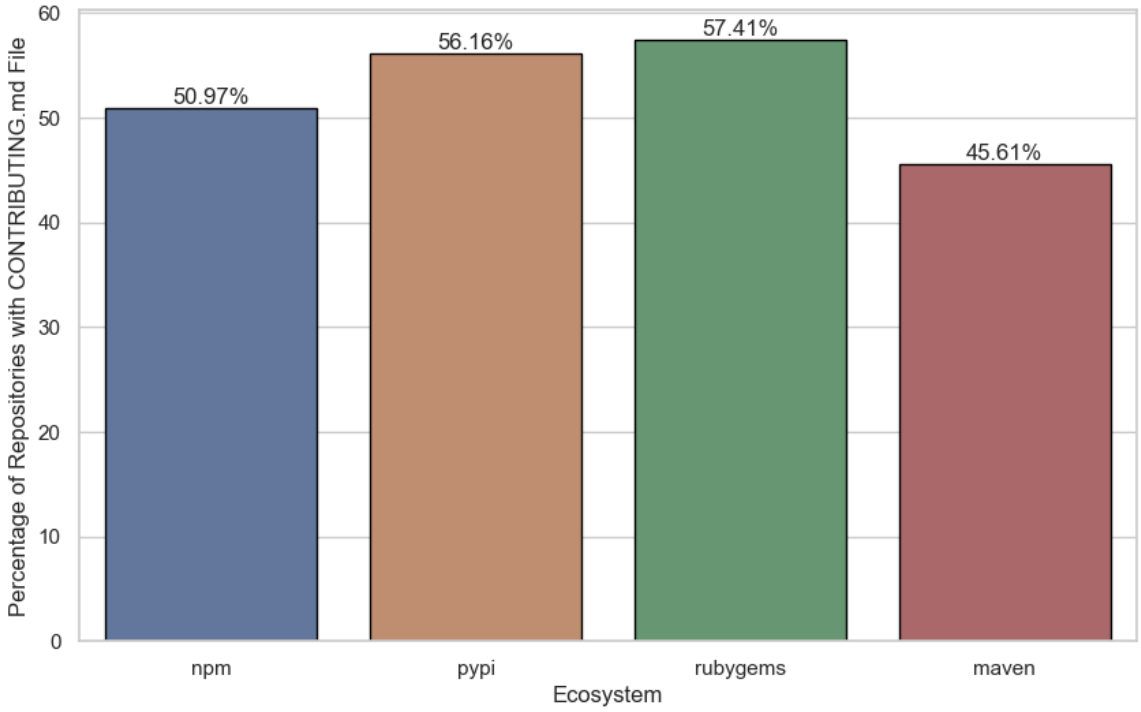


**Figure 4.5: Percentage of Contributing Guidelines Presence across Ecosystems**

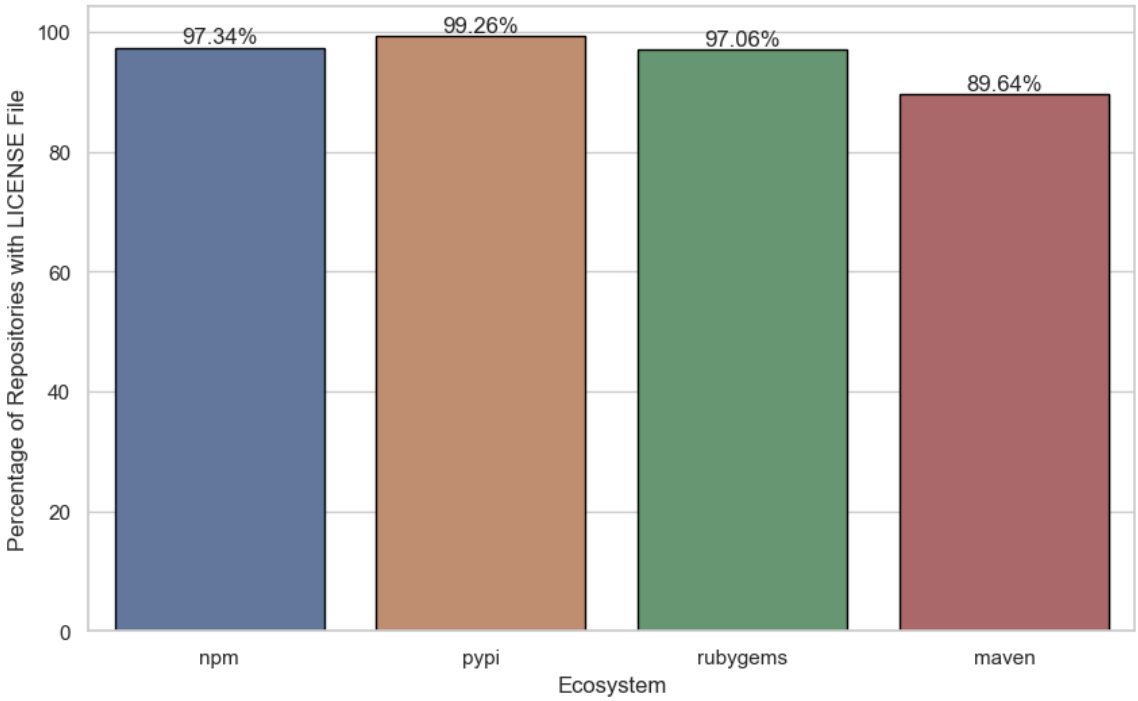


**Figure 4.6: Percentage of License Inclusion across Ecosystems**

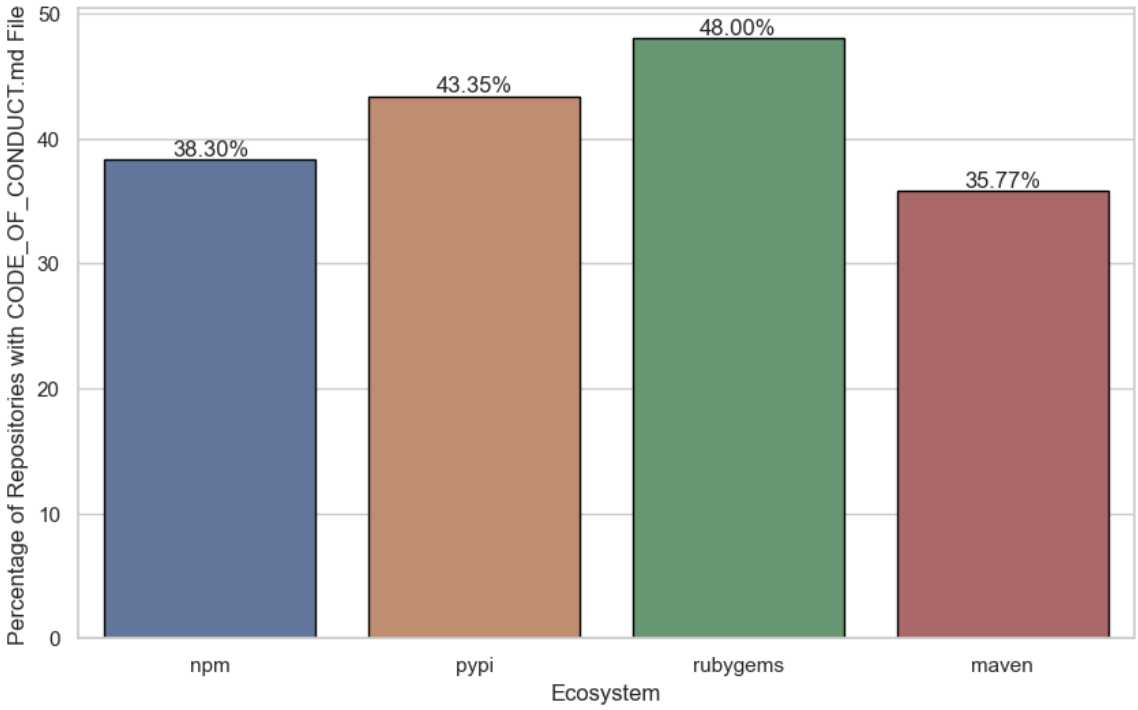


**Figure 4.7: Percentage of Code of Conduct Adoption across Ecosystems**

# Chapter 5

# COMMUNITY RESPONSIVENESS AND PARTICIPATION

In this chapter, we examine the dynamics of community engagement within OSS ecosystems, focusing on the interaction between project insiders and external contributors. The chapter is structured to address specific research questions that explore community responsiveness and participation patterns.

## 5.1 RQ2.How do project insiders respond to issues opened by outsiders in four ecosystems?

OSS projects are defined by their collaborative nature, drawing on the diverse contributions of a global community of developers. This community typically comprises *insiders* who are regular and long-term contributors, and *outsiders* who contribute occasionally or are new to the project. The success of OSS projects relies heavily on the vitality and participation of their community. This community influences the developed software's quality, sustainability, and progress [7, 8]. In an era where OSS underpins much of the global digital infrastructure, the roles of these contributors become even more significant.

Outsiders' contributions bring fresh perspectives, innovative ideas, and diverse skills. Yet, their involvement is often impeded by barriers such as unfamiliarity with the project's codebase and community dynamics. The insiders' responsiveness to outsiders' contributions and queries is therefore vital. It fosters effective collaboration, a sense of belonging, and motivation among newcomers; otherwise, they abandon the project [36].

### 5.1.1 Method

The data about the repository's issues and their comments was sourced from 4,000 GitHub repositories using the GitHub API. We collected the following data about each issue:

- **Issue Created Time:** The timestamp when the issue was opened provides a baseline for measuring response times.
- **Author Association:** An attribute indicating the author's relationship with the project on GitHub. Possible values are: `COLLABORATOR`, `CONTRIBUTOR`, `MANNEQUIN`, `MEMBER`, `NONE`, and `OWNER` [43]. An *insider* is defined as someone whose `author_association` is `COLLABORATOR`, `MEMBER`, or `OWNER`.
- **Total Comments:** The number of comments on an issue, aiding in assessing the level of engagement it received.
- **Response Time from an Insider:** This metric calculates the time from an issue's opening to the first comment by an insider.

We focused on two objectives.

First, we quantified the engagement of community insiders with issues raised by outsiders in our chosen open-source ecosystems. Issues opened by outsiders were identified by filtering the dataset based on the 'author_association' status. We analyzed the responses of insiders, identified by their 'author_association' statuses, to assess their engagement with these issues.

Aggregating the data on an ecosystem level allowed for assessing insider engagement across different communities. We measured the rate of insider response, i.e., the percentage of outsider-opened issues that received comments from insiders. This metric

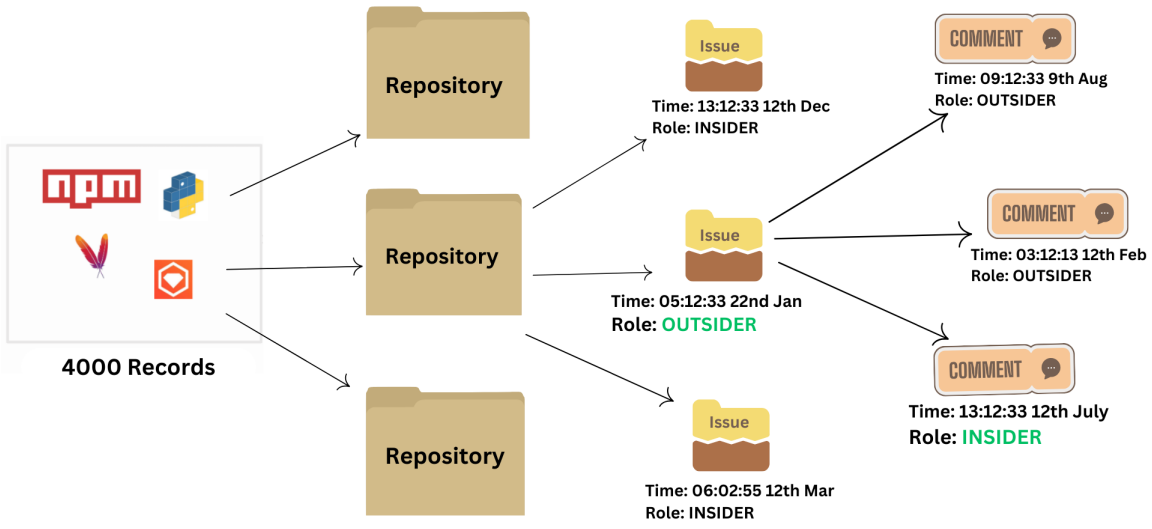


**Figure 5.1: Highlighting Insider Engagement with Outsider Issues through Comments**

measures how frequently insiders in various ecosystems engage with contributions from outsiders (See Figure 5.1).

The second objective was to assess the speed of response from insiders. This involved calculating the median time (in hours) insiders took to respond to issues raised by outsiders. The time interval between creating an issue by an outsider and posting the first comment by an insider was determined.

The median time-to-response for each ecosystem was calculated to provide a representative measure of responsiveness. Calculating the median instead of the mean helped mitigate the effect of outliers, ensuring a more accurate representation of the typical response behavior within each community.

### 5.1.2 Results

The total number of issues opened by outsiders in each ecosystem: Maven (44,786), NPM (46,664), PyPI (38,390), and RubyGems (20,435).

The analysis revealed a nuanced landscape of insider engagement across various ecosystems. In the Maven ecosystem, 43.46% of the issues opened by outsiders received comments from insiders, while the NPM ecosystem showed a similar engagement rate of 43.13%. Conversely, PyPI and RubyGems ecosystems demonstrated relative higher engagement rates, with 56.47% and 57.55%, respectively (see Figure 5.2). These findings suggest a more active insider response to outsider contributions in the latter ecosystems.

These statistics indicate significant variance in the degree of insider participation among ecosystems. In some, nearly half of the outsider issues receive insider attention, reflecting a potentially supportive and inclusive environment for new contributors.

The responsiveness analysis of different ecosystems provided key insights. Maven insiders had the longest median response time of 36.52 hours among the four, while NPM was the most responsive with a median response time of 9.43 hours. The PyPI and RubyGems ecosystems had intermediate response times of 17.94 and 12.52 hours, respectively (see Figure 5.3).

The variation in response times might be influenced by community size, issue volume, or management practices within each ecosystem. NPM's lower median response time could highlight efficient community management in addressing outsider contributions. In contrast, Maven's longer response time could indicate areas for improvement in community interaction or support structures.

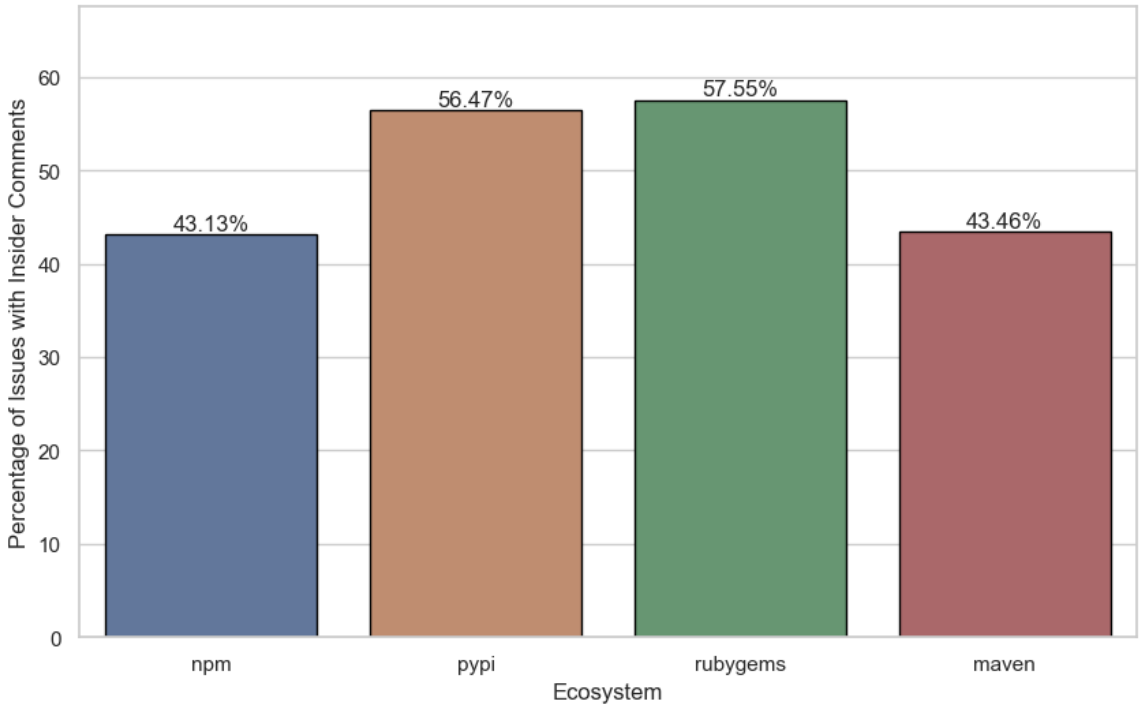


**Figure 5.2: Percentage of Issues in Each Ecosystem with Outsider-Opened Issues and Insider Comments**

Summarizing the engagement and response time metrics findings, it becomes evident that NPM insiders seem to respond quickly, but only if they respond. RubyGems appears to be the best in both regards: they are comparatively more likely to respond and to respond quickly. Insiders from Maven, though not frequently responding, tend to take the longest time when they do engage.

It is essential to note that none of these response times are inferior. Even at 36 hours, the response time in Maven is still only 1.5 days, which is commendable for volunteers maintaining open-source projects.

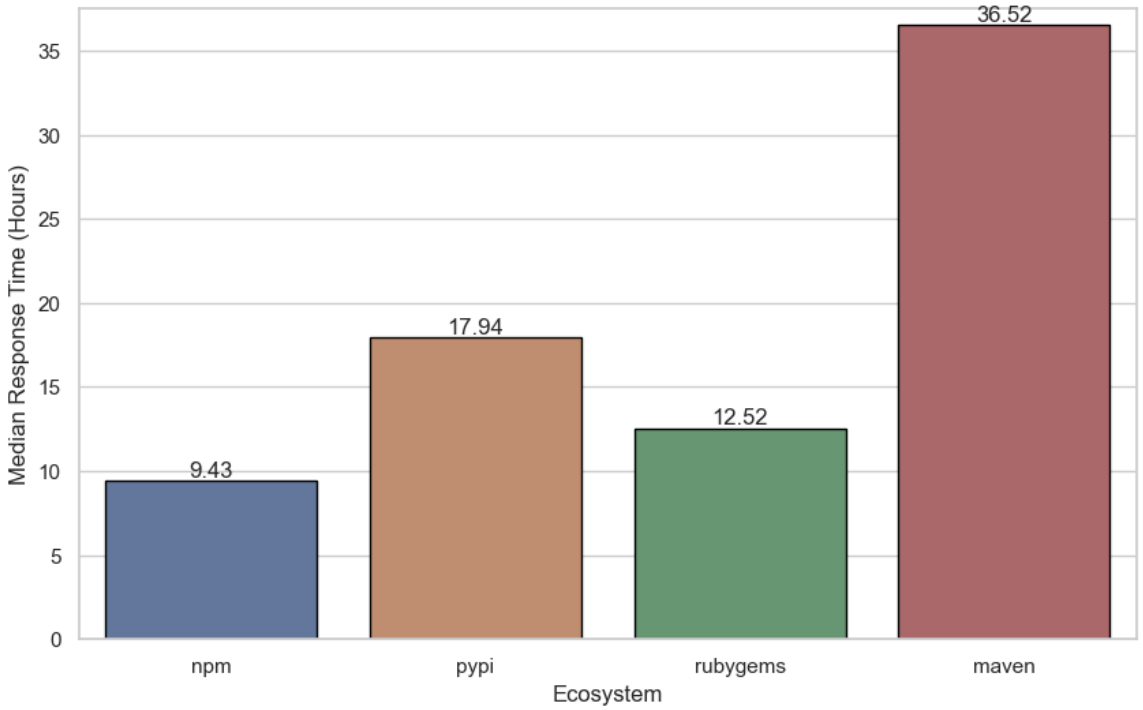


**Figure 5.3: Median Insider Response Time to Outsider-Opened Issues Across Repositories in Each Ecosystem**

## 5.2 RQ3. How is the distribution of contributions among project contributors characterized within four ecosystems?

Exploring how contributions are distributed within a community is crucial to understanding the dynamics of collaborative participation in OSS. In this context, *contributions* refer to the code changes made by community members, including code additions and deletions in the form of commits to GitHub. This research question identifies whether contributions are concentrated among a few members or more evenly spread across the participant base.

This analysis provides insights into the extent and variety of community involvement. When contributions are concentrated among a few individuals, it may indicate a hier-

archical or potentially oligarchic structure[1]. This concentration can be a risk factor for the project's adaptability and resilience [44]. In contrast, a more evenly distributed pattern of contributions is often a marker of a diverse and inclusive environment. This diversity fosters innovative problem-solving and a more comprehensive range of perspectives and contributes to a robust and sustainable community ecosystem [45].

### 5.2.1 Method

We gathered data on the number of commits made by the top 100 contributors in a GitHub repository across four ecosystems to assess their contributions to the projects. The number of commits was chosen as the primary measure of contribution over other metrics, such as lines of code added or deleted. Commits represent discrete, intentional contributions to a project, capturing the frequency of a contributor's engagement and providing a straightforward method for quantitative analysis [46, 47]. Our analytic steps are defined below and summarized in Figure 5.4.

Across different repositories, the total number of commits per project varied significantly, making direct comparisons between projects unfeasible. To address this challenge and ensure a fair comparison, it became essential to implement a normalization technique. Consequently, for each repository, we used the commit count of the contributor with the most commits as the benchmark against which to normalize values. This was achieved through a process known as *MaxScaling or Integer Scaling Normalization*, as described in [48].

[1]An "oligarchic structure" in the context of OSS communities refers to a situation where a small group of individuals or entities holds a disproportionate amount of control or influence over the project's direction, decision-making processes, and contributions. This can lead to centralization of power and may limit broader community engagement.

Following this, a series of summary statistics were employed to further describe the distribution of these normalized commit counts per author, representing their contributions to the repository. These statistics included the median number of commits, and the inter-quartile range (IQR) across all contributors to a project.

The IQR, specifically, quantifies the spread of the middle 50% of the data, offering a measure of the variability in contributor activity levels. Moreover, by analyzing the IQR, we gain insights into the differences in commit counts between high-contributing (Q3) authors versus low-contributing (Q1) authors within a repository. This approach ensures a nuanced understanding of participation and highlights the range of engagement levels among contributors in each repository.

We used the standard IQR-based formula to identify outliers [49], i.e., contributors with exceptionally high or low commit counts. Data points falling beyond $Q1 - 1.5 \times IQR$ and $Q3 + 1.5 \times IQR$ were classified as outliers and excluded from the analysis. By concentrating on data within the IQR, we mitigated the potential skewing effect of extreme contributions on our analysis.

### 5.2.2 Results

Figure 5.5 and Table 5.1 represent the distribution of the Interquartile Range (IQR) of the number of commits per author for a repository within a specific ecosystem. For a given ecosystem, we compute the median spread of contributions within each repository (i.e., a median of IQRs). A higher Median IQR indicates a greater disparity between 'high contributing' and 'low contributing' authors within that ecosystem, suggesting that most contributions come from a smaller group of authors. Conversely, a lower Median IQR suggests a smaller difference in contribution levels, indicating a more evenly distributed pattern of contributions among contributors.

In Table 5.1, the NPM ecosystem's median IQR of 0.009 says that the difference in contributions between the 75th-percentile contributor and the 25th-percentile contributor was 0.9% of the max number of commits in that repository. So, for example, if the repository's highest contributor made 1000 commits, then the difference between the 75th-percentile contributor and the 25th-percentile contributor would be 9 commits.

**Table 5.1: Descriptive statistics of the median commit difference between high-contributing and low-contributing authors in a repository across ecosystems**

| Ecosystem | **Median IQR** | **Percentage of commit difference** |
|---|---|---|
| NPM | 0.009 | 0.9% |
| PyPI | 0.012 | 1.2% |
| RubyGems | 0.005 | 0.5% |
| Maven | 0.007 | 0.7% |

PyPI had the highest median IQR value (see Table 5.1) among the studied ecosystems. This might indicate that a few contributors contribute substantially more than others, pointing to a concentrated source of contributions. Conversely, the RubyGems ecosystem exhibits the lowest IQR of 0.005 (or 0.5% of commits difference), indicating a lower difference between high-contributing and low-contributing commit authors, suggesting a more egalitarian contribution pattern. These extremes in median IQR values—highest in PyPI and lowest in RubyGems—highlight the variability in contribution dynamics across different ecosystems.

Sitting in the middle are the NPM and Maven ecosystems. For NPM, the median IQR of 0.009 (or 0.9% of commits gap between high-contributing and low-contributing) might suggest a pattern where fewer contributors contribute more substantially. Maven, with a median IQR of 0.007 (0.7% of commits) has a comparatively more even distribution of contributions, showing a broader engagement across its contributor base.

Affecting our interpretation of these results is the number of unique individuals contributing in a given ecosystem. In an ecosystem with a large number of contributors (e.g., NPM), commit counts are likelier to have higher spread than in an ecosystem with a smaller number of contributors (e.g., RubyGems).

Therefore, we wanted to also considered the total number of contributors per repository. However, the GitHub API only provides the top 100 contributors for a given repository. Therefore, the number of unique contributors is "capped" at 100 in this dataset.

Nevertheless, we calculated the median number of contributors across ecosystems and integrated this data with the distribution of contributions (see Figure 5.6). RubyGems has the largest median at 77, suggesting a wide contributor base. Paired with the fact that RubyGems also had the lowest median IQR, this suggests relatively even contribution levels between high-contributing and low-contributing commit authors. PyPI has a median of 54 contributors per repository, and the *highest* median IQR. This suggests a smaller, more active contributor group for the average repository. NPM and Maven have lower medians of 30 and 34 contributors per repository, and lower median IQRs, suggesting contributions are less evenly spread compared to RubyGems but more so than PyPI.

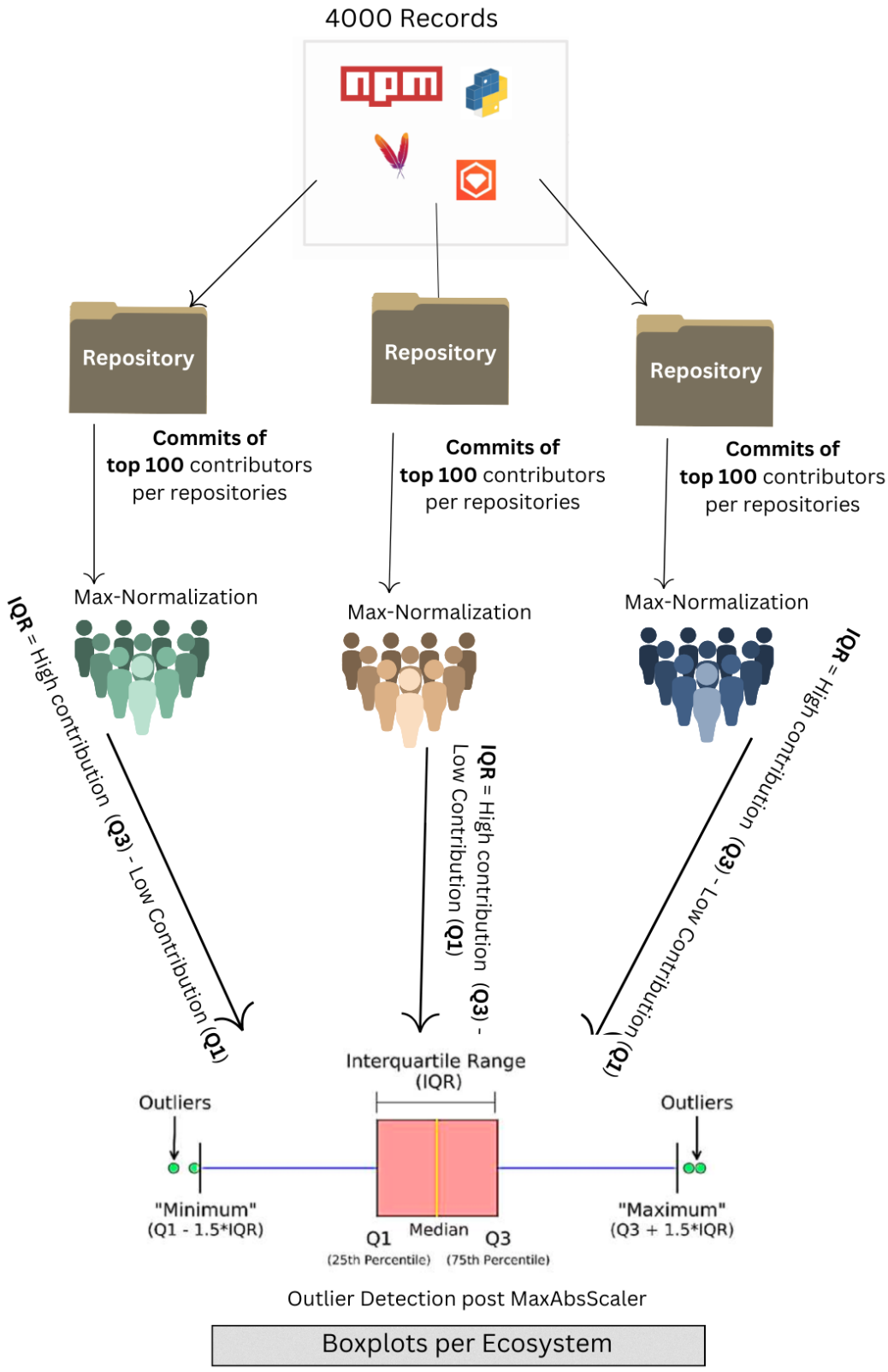


**Figure 5.4: Highlighting the normalized transformation of the commits of top 100 contributors in ecosystems**

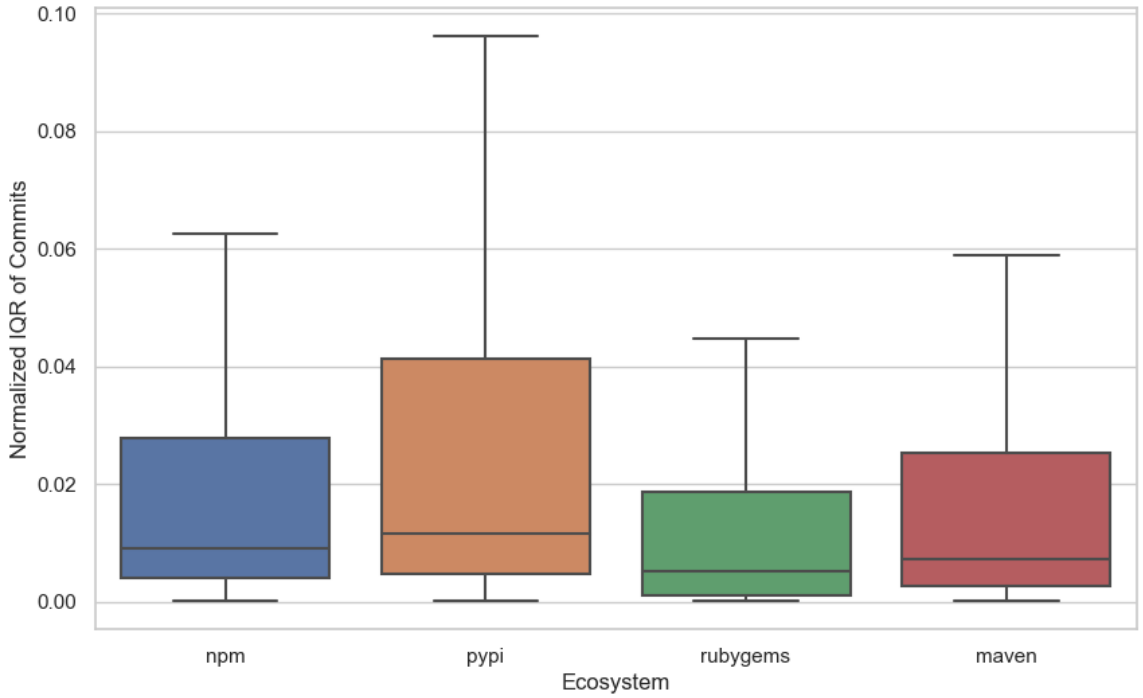


**Figure 5.5: Comparative analysis of IQR distribution for the contribution gap between high-contributing and low-contributing commit authors in a repository across ecosystems**

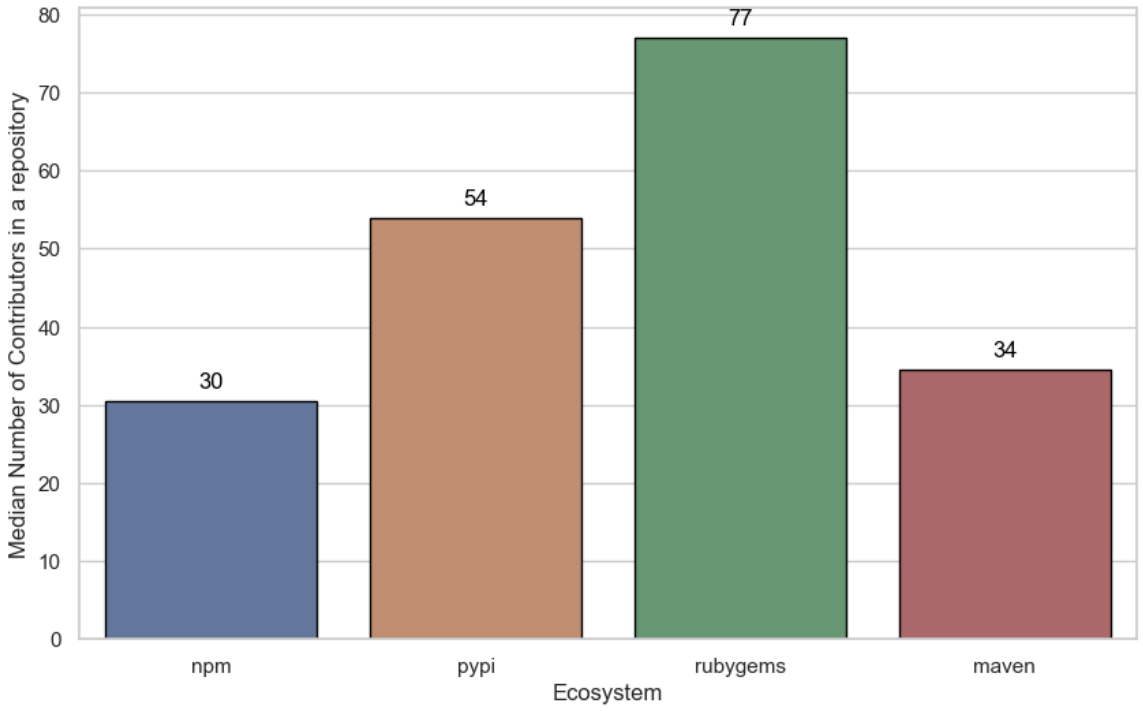


**Figure 5.6: Comparative analysis of the median number of contributors in a repository for each ecosystem**

# Chapter 6

# CONTRIBUTOR ATTRACTION AND RETENTION

The health and sustainability of OSS projects depend on their ability to attract contributors and to retain them over time. In this chapter, we study the intricacies of contributor retention rates within the four ecosystems. We also investigate the effectiveness of the presence of a "good first issue" label in attracting and retaining contributors, thereby ensuring the continued growth and innovation of projects.

## 6.1 RQ4.What is the contributor retention rate within four ecosystems?

Prior work has underscored the importance of contributor retention in OSS projects. Zhou et al. emphasized that the patterns of engagement and retention are influenced significantly by the involvement of sponsoring companies in OSS projects [50]. Moreover, Rashid, Clarke, and O'Connor discussed the challenges of knowledge loss due to contributor turnover, suggesting the need for proactive knowledge retention strategies in OSS projects [51]. Finally, Yamashita et al. discovered that larger projects tend to attract and retain more contributors [52]. This highlights the direct impact of contributor retention on the sustainability and effectiveness of OSS projects.

This study aims to contrast and compare the patterns of contributor engagement in the four ecosystems, focused on ascertaining the proportion of contributors who persist beyond their initial involvement and analyzing the frequency and nature of their continued engagement in these different environments.

### 6.1.1 Method

There can be various forms of contributions to a project to measure engagement (e.g., code contributions, documentation, feature requests, and bug reports). We have chosen to focus on Pull Requests (PRs) due to their significance in collaborative contributions [21, 22]. We looked at PRs raised in all 4000 repositories in our dataset between October 2022 and October 2023. Each PR record contains details like its creation time, username, user type, status (open or closed), and the author's association with the repository.

We only selected PRs from *project outsiders*.[1] To do this, we specifically focused on the PRs that were not submitted by bots and where the author's association did not contain the enum values `COLLABORATOR`, `MEMBER`, or `OWNER`.

Contributors' data was organized chronologically based on PR creation times for each repository to facilitate the identification of contributors with any subsequent PR in a repository. Our analysis depended on two essential metrics for each repository to evaluate engagement.

1. **Percentage of Contributor Retention (PCR):**

$$PCR = \frac{\text{Total number of contributors with more than one PR}}{\text{Total number of contributors}} \times 100\%$$

   This metric measures the likelihood of contributors becoming repeat contributors in a project community after their initial contribution in a year. The period considered is between October 2022 and October 2023.

[1]For this study, external contributors are defined as individuals contributing to a repository for the first time within the 1-year data collection timeframe, with author association values with the project as `CONTRIBUTOR`, `FIRST_TIMER`, or `FIRST_TIME_CONTRIBUTOR`. These enum values are defined in GitHub [43].

2. **Average of Contributor Repeat-Contribution (ACR):**

$$ACR = \frac{\text{Total number of PRs from contributors with more than one PR}}{\text{Total number of contributors with more than one PR}}$$

This metric measures the average number of contributions per repeat contributor for a repository in a year. This helps us calculate the extent to which authors continued to contribute after their first PR in the given period. The period considered is between October 2022 and October 2023.

### 6.1.2 Results

Upon analyzing the engagement of contributors in four ecosystems, distinct patterns were observed in Figure 6.1.

PyPI led with a 36.12% rate of contributors submitting subsequent PRs within a year, suggesting strong retention. Maven followed closely with a 34.14% retention rate. RubyGems and NPM showed potential areas for improvement in contributor engagement strategies, with retention rates of 26.66% and 21.85%, respectively.

Contributors to Maven demonstrated the highest level of activity, with an average of 10 subsequent pull PRs submitted to the same repository within a year. This indicates a significant level of sustained engagement in comparison to NPM, where the average was notably lower at 6 subsequent PRs. PyPI and RubyGems exhibited moderate activity levels, with averages of 8 and 9 subsequent PRs, respectively. It is important to clarify that these figures represent averages of PRs to the same repository, which may not fully capture contributors' overall activity across multiple projects. Therefore, while some contributors may not show long-term engagement with a single project, they could be highly active across different projects within the ecosystem.

This distinction underscores the variability in contributor engagement patterns and suggests the presence of diverse forms of participation that extend beyond long-term contributions to individual projects.

Considering both metrics, PyPI and Maven emerge as ecosystems with robust contributor engagement, balancing retention with active subsequent contributions. Despite a lower retention rate, RubyGems sees active engagement from those who stay. NPM, having the lowest figures in both metrics, may need to implement more effective engagement mechanisms to foster and sustain contributor participation.

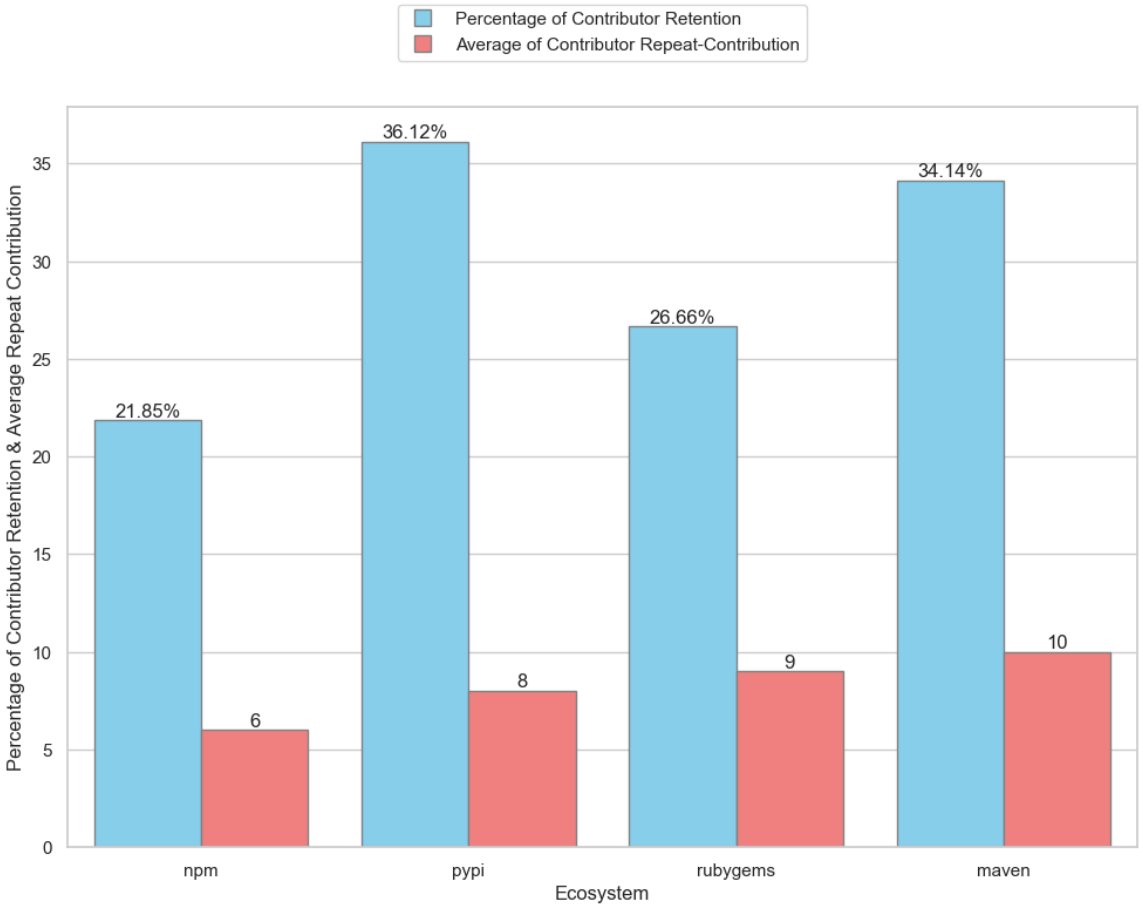


**Figure 6.1: Comparative Analysis of Contributor Engagement in OSS Ecosystems by Percentage of Repeat Contributors and Their Average Subsequent Contributions**

[1]All contributions are calculated based on the Pull Requests raised by an individual over one year in a repository.

## 6.2 RQ5.What percentage of repositories across four ecosystems feature a "good first issue" label, and what proportion of contributions to projects within each ecosystem come from newcomers?

Welcoming new contributors is essential for fostering a collaborative and dynamic ecosystem that values various contributions beyond just coding, such as community management and outreach [53, 54, 55].

A critical aspect of community welcomingness is the adoption of labels such as "good first issue" (GFI), which are instrumental in guiding newcomers to appropriate starting tasks [56, 57]. The presence and proper use of GFI labels indicate a project's readiness to integrate support and motivate new contributors. This study seeks to extend the understanding of community welcomingness by examining the prevalence of GFI labels across ecosystems, thus providing a quantitative measure of a community's efforts to lower entry barriers for newcomers.

We also study the percentage of contributors who newcomers to open-source software in general. Understanding how different ecosystems attract and integrate new members is essential. Analyzing this ratio can offer insights into the inclusivity and growth potential of these communities.

The findings are expected to provide valuable strategies for community leaders and maintainers with targeted strategies to enhance newcomer inclusion. GFI labels show ecosystem receptivity, and contribution reflects activity. These analyses provide actionable insights into integrating and retaining new contributors in OSS projects.

### 6.2.1 Method

We first identified GitHub repositories that support newcomer integration through the GFI label to assess community welcomingness. We collected and examined the issue labels from all the ecosystem's repositories. By filtering, we isolated repositories with the GFI label in their list of available labels. The proportion of repositories with GFI labels was then calculated against the total number of repositories within each ecosystem (see Figure 6.2). This metric served as an indicator of how welcoming a project is to first-time contributors and encouraged helpful contributions to the project, as it is recommended by GitHub [58].

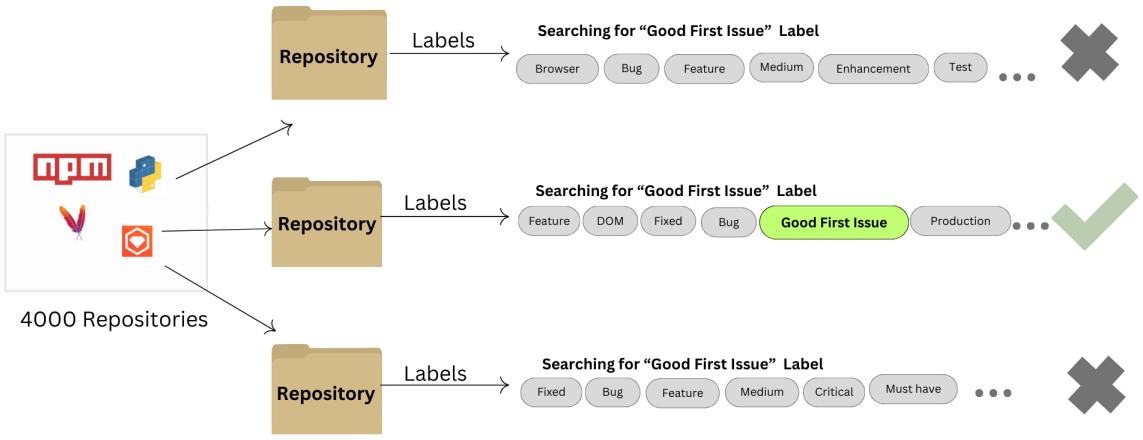


**Figure 6.2: Highlighting the process of finding the GFI Label across ecosystem repositories**

To identify contributors who are newcomers participating in open-source in general (based on GitHub activity), we collected data about contributors who made pull requests to a repository in our corpus between October 2022 and October 2023. We focused on two primary keys for each record: the author's username and the timestamp of the pull request.

From this list of PRs and authors, we identified first-time contributors as those who had never made a PR to *any* GitHub repository before. This was done by employing the GitHub search API to verify the absence of any past pull request contributions by these individuals (during any time period), using their usernames and creation times as parameters.

Subsequently, an analysis was conducted to calculate the percentage of contributors (within the October 2022 to October 2023 period) who made their first ever GitHub contributions during that period.

### 6.2.2 Results

Figure 6.3 shows that the PyPI ecosystem leads with 35.99% of its repositories marking its issues containing the GFI label, suggesting a proactive stance in targeting and motivating newcomers. The NPM ecosystem demonstrates a lower presence of the label, with 23.33% of repository labels featuring it. Maven follows with 21.70% of its repositories indicating GFI, while RubyGems shows the lowest among all with 12.73%, potentially reflecting a lesser emphasis on or need for newcomer orientation.

Figure 6.4 displays the percentage of contributors who are newcomers without prior contributions in GitHub for each ecosystem. The Maven ecosystem displayed the highest percentage, with 10.73% of contributors (within the period of October 2022 to October 2023) making their first-ever pull request. The PyPI ecosystem followed closely with 9.20%, while NPM accounted for 8.72%. The RubyGems ecosystem had the lowest observed percentage of new contributors, with 5.38%, potentially indicating a more mature or saturated community.

Taking these results together (Figures 6.3 and 6.4), PyPI shows a robust initial welcome indicated by its high GFI label adoption. Yet, its newcomer contribution, while

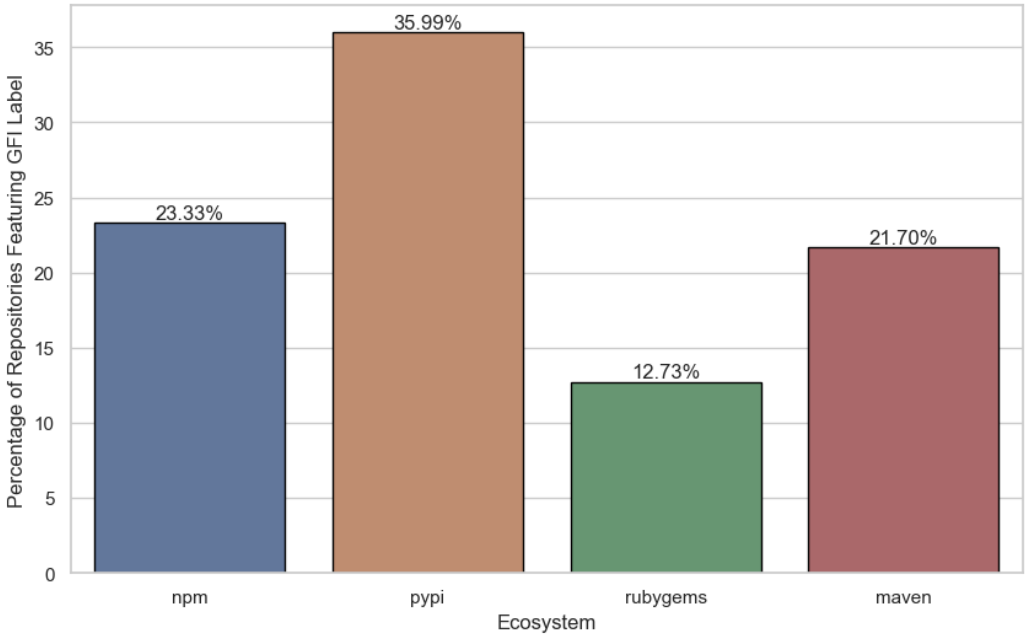


**Figure 6.3: Comparative Analysis of GFI Adoption in Ecosystems**

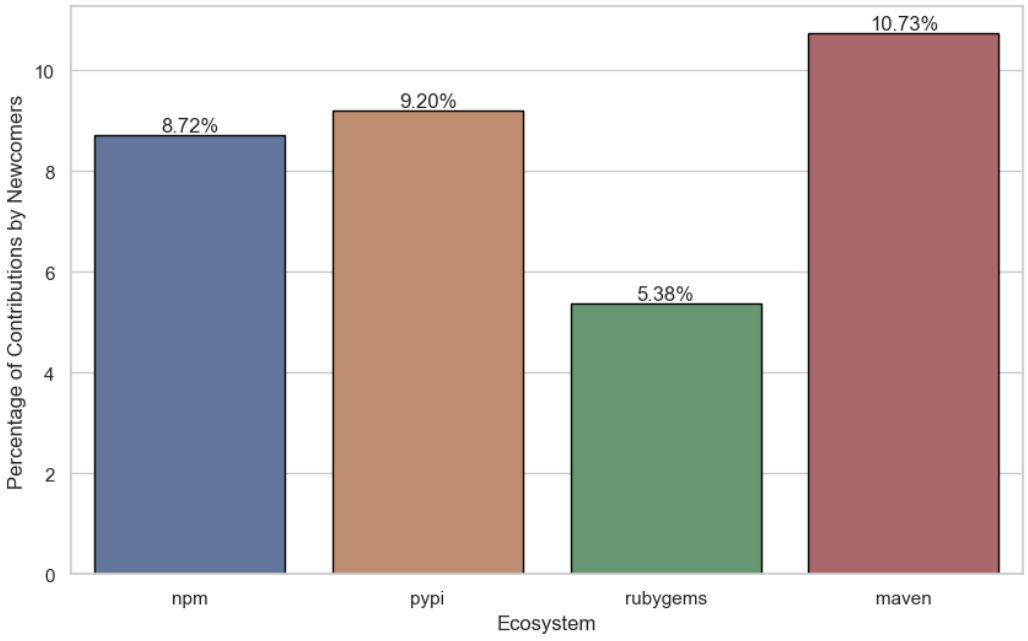


**Figure 6.4: Comparative Analysis of Newcomer Contribution Across Ecosystems**

substantial, is not the highest among the ecosystems. The Maven ecosystem, though not the frontrunner in GFI labels, leads in newcomer contributions, hinting at the possibility of other factors that may affect the process of integrating newcomers. PyPI and NPM show harmony with considerable measures in both areas, reflecting a well-rounded ecosystem. Conversely, RubyGems, lagging in both areas, points to opportunities for growth in community engagement strategies.

# Chapter 7

# DISCUSSION

We summarize and discuss the results from our research questions. We discuss each ecosystem's strengths, challenges, and potential areas for improvement. Our results are summarized in Table 7.1. We aim to understand the interplay between community dynamics, organizational structures, and newcomer experiences within these open-source software environments. From contrasting approaches to acquiring new contributors in NPM and Maven ecosystems, to addressing contribution disparities within PyPI, and examining the need for newcomer-focused initiatives in RubyGems, each section offers valuable insights into the evolving landscape of open-source collaboration. Furthermore, the discussion underscores the significance of context-specific strategies, such as the adoption of "good first issue" labels, in fostering newcomer assimilation and community accessibility.

**Acquiring New Contributors in NPM and Maven Ecosystems.** The analysis of NPM's and Maven's trend towards acquiring new contributors highlights a contrast between the lower presence of repository metadata files, insider engagement, contribution disparity, and the positive focus on GFI welcomeness and newcomer contribution. In Maven, a lower engagement rate and extended response times might reflect limitations in community organization or resource allocation, which is crucial for effective responsiveness [59]. On the other hand, NPM has the fastest response and the lowest engagement rate out of all the ecosystems, which may be due to the large volume of contributors spread across multiple projects within its ecosystems. Regarding newcomer retention, NPM demonstrates a lower retention rate compared to Maven. While NPM struggles with retaining newcomers, Maven shows a rela-

tively higher rate of retaining new contributors and has a slightly higher newcomer contribution than NPM. This suggests differing approaches to newcomer integration and attraction strategies within the two ecosystems, but both of them are showing positive trends toward acquiring new contributors.

**PyPI's Contribution Disparity.** The PyPI ecosystem demonstrates strength across various metrics, including the high presence of repository metadata, insider engagement with a quick response time of approximately 18 hours, and a welcoming environment for newcomers, as evidenced by its high GFI welcomeness and high newcomer contributions. It also has the highest contribution retention rate among all the ecosystems. However, a notable concern arises from the significant contribution disparity within the PyPI ecosystem, indicating that only a few individuals contribute a substantial amount of code compared to others (§5.2.2). This suggests a potential hierarchical structure where a small group exerts significant effort. Such imbalances may pose challenges to diversity and innovation. This imbalance suggests potential issues related to accessibility, project complexity, or varying levels of expertise among contributors. Addressing this disparity is crucial to ensuring a more equitable distribution of contributions and fostering a diverse and sustainable contributor base within the PyPI ecosystem.

**Strong Community, Need for Newcomer Focus in RubyGems.** In the RubyGems ecosystem, the strength of its community base and project metadata is evident, as reflected in its highest presence of repository metadata files and insider engagement, with response times averaging around 13 hours. Additionally, it suggests a more evenly spread contribution pattern with the highest median of 77 active contributors per project, indicating a more egalitarian and inclusive environment where a broader range of contributors participate in the project development, fostering diverse perspectives and robust problem-solving. Contribution patterns reflect the underlying

social and organizational structures of OSS communities. A more centralized contribution pattern, as seen in PyPI, might result from established contributors' control or the presence of highly skilled individuals, which can limit broader community engagement and innovation. On the other hand, a decentralized pattern, such as in RubyGems, promotes a healthier ecosystem by encouraging participation from a wide range of contributors. However, there are notable areas for improvement, particularly in contributor retention, which remains at a moderate level and requires attention. Similarly, the lower presence of the "good first issue" (GFI) label, found in only 12.73% of repositories, and the lowest newcomer contribution rate at 5.38% highlight the need for enhanced focus on newcomer onboarding initiatives. This trend suggests that although projects within the RubyGems ecosystem appear to be in a mature state, there is minimal emphasis on facilitating the integration of new contributors. Addressing these challenges could foster a more inclusive and sustainable contributor ecosystem within RubyGems, ensuring continued growth and innovation. It highlights the need for community managers and project leaders to foster environments that welcome newcomers and maintain a balance between experienced and new contributors.

**GFI label attracts newcomer contributions.** The notable adoption of a GFI label in ecosystems like PyPi, as reflected in 35.99% of its repositories featuring these labels, highlights a proactive approach to engaging newcomers. This strategy aligns with Carillo, Huff, and Chawner's findings, emphasizing the significance of tailored initiatives like GFI labels in enhancing newcomer assimilation within Free/Open Source Software projects. Such labels facilitate initial involvement by directing newcomers to suitable tasks and fostering a sense of belonging and community integration. The contrast with ecosystems like RubyGems, where GFI label adoption is comparatively lower, might reflect different community dynamics or an already established contributor base, pointing to the importance of context-specific strategies to boost

community accessibility and newcomer orientation [37]. The variation in newcomer contribution rates across different ecosystems, particularly Maven's leading position and RubyGem's lower end, suggests the influence of factors beyond GFI label adoption. Maven's higher newcomer contribution rate may be attributed to a supportive community culture or effective mentorship programs. This underscores the need for a holistic approach to fostering newcomer engagement. This approach should encompass welcoming initiatives like GFI labels and active mentorship and community support, emphasizing the complex interplay of factors shaping newcomer experiences in OSS ecosystems.

**Table 7.1: Overview of Key Ecosystem Characteristics**

| | Repository Metadata | Insider Engagement | Contribution Disparity | Contributor Retention | GFI Welcomeness | Newcomer Contribution |
|---|---|---|---|---|---|---|
| **NPM** | Lowest adoption of PR Templates, Contributing, License and code of conduct | 43.13% of outsider issues receive insider comments with a very quick median response time of around 9 hrs | Higher individual contribution with a median of 30 active contributors per project | Low retention rate at 21.85% with subsequent contributions of 6 PRs | Moderate presence of GFI label with 23.33% across its repositories | 8.72% of contributors made their first-ever PR |
| **PyPI** | High PR Templates, Contributing, Licensing, and Code of Conduct adoption | 56.47% of outsider issues receive insider comments with a median response time of around 18 hrs | Significant disparity in contribution levels with a median of 54 active contributors per project | High retention rate at 36.12% with low subsequent contributions of 8 PRs | Highest proportion of repositories with 35.99% featuring the GFI label | 9.2% of contributors made their first-ever PR |
| **Maven** | Needs improvements in PR Templates, Contributing, License, and code of conduct | 43.46% of outsider issues receive insider comments with an extended median response time of around 1.5 days | High individual contribution with slight centralization, with a median of 34 active contributors per project | High retention rate, slightly lower than PyPI, with high subsequent contributions of 10 PRs | Moderate presence of GFI label with 21.7% across its repositories | 10.73% of contributors made their first-ever PR |
| **Ruby Gems** | Highest adoption of Contributing, Code of Conduct, and PR Templates among others | 55.55% of outsider issues receive insider comments with a quick median response time of around 13 hrs | Most even distribution of contributions with the highest median of 77 active contributors per project | Moderate retention rate at 26.66% with high subsequent contributions of 9 PRs | Few repositories with only 12.73% having GFI label | Only 5.38% of contributors made their first-ever PR |

# Chapter 8

# FUTURE WORK

**Qualitative Study of Newcomers' and Contributors Experiences.** The investigation presented in this thesis primarily quantifies the behaviors and patterns within OSS communities, highlighting what happens in terms of newcomer integration, contributor retention, and community engagement. However, the quantitative data lacks the depth to explain why these patterns exist or how the individuals involved perceive their experiences. Future studies could employ qualitative methods to delve into the subjective experiences of newcomers in OSS communities. This approach would uncover the motivations, challenges, and personal journeys of contributors, offering a richer understanding of the factors that contribute to a welcoming and inclusive community environment. Such studies would complement the current findings by adding depth to our understanding of the dynamics within OSS ecosystems, particularly from the perspective of those newly integrating into these communities.

**Longitudinal Analysis of Community Dynamics.** This thesis provides a snapshot of community engagement, contribution patterns, and newcomer experiences at a specific point in time and limit (e.g., contribution patterns evaluated only the top 100 contributors, and newcomer experiences data taken from October 22 to October 2023). A longitudinal study would allow us to track how these aspects evolve, providing insights into the sustainability of community practices, the long-term impact of initiatives aimed at newcomers, and how changes in the ecosystem (such as the introduction of new technologies or shifts in community governance) affect participation and retention rates. This could help in understanding how OSS communities

adapt to challenges and changes over time, and what strategies are most effective in ensuring their longevity and vibrancy.

**Study of Tech Support for Community Engagement.** There is a need to explore and expand beyond the boundaries of GitHub datasets to explore a broader spectrum of technological tools and platforms that facilitate community engagement and support within OSS projects. One avenue of investigation involves diving into community dashboards such as Stack Overflow, which serve as invaluable repositories of knowledge and expertise for developers seeking assistance with coding challenges and troubleshooting. Moreover, we intend to analyze the impact of project-specific community websites like Dev.to, which offer dedicated spaces for developers to share insights, exchange ideas, and seek guidance tailored to particular OSS projects. Furthermore, there are discussion forums like Reddit, where vibrant communities congregate to discuss a wide array of topics, including software development. By examining the role of these diverse platforms in nurturing community engagement and providing technical support, we aim to offer comprehensive insights into the multifaceted nature of tech support mechanisms within the OSS ecosystem.

**Expansion across varied OSS Ecosystems.** The scope of our research, primarily focusing on ecosystems like NPM, PyPi, Maven, and RubyGems, opens the door for future work to explore a broader spectrum of OSS communities. In broadening our research ecosystem scope, we see two main paths:

1. **Horizontal Expansion (Language Paradigm Analysis):** One promising direction involves broadening the study to encompass additional programming languages such as Go and Rust, along with their respective ecosystems. By doing so, we can delve into how diverse language paradigms influence community engagement and contribution patterns. Through comparative analysis across

these language ecosystems, valuable insights can be gained into the nuanced dynamics shaping open-source software development.

2. **Vertical Expansion (Domain-specific Comparisons):** Another crucial aspect of our future work entails exploring communities within distinct development domains, examining communities within distinct development domains—such as application versus library development or front-end versus back-end frameworks—and comparative analysis among frontend or backend libraries can shed light on how the nature of the project impacts contributor behavior and community strategies. This approach will allow for a more nuanced understanding of the factors that foster or hinder effective community engagement across a wider array of OSS projects.

# BIBLIOGRAPHY


[1] James C Davis, Christy A Coghlan, Francisco Servant, and Dongyoon Lee. The impact of regular expression denial of service (redos) in practice: an empirical study at the ecosystem scale. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, pages 246–256, 2018.

[2] NPM. Node Package Manager. `https://www.npmjs.com`.

[3] PyPI. Python Package Index. `https://pypi.org`.

[4] Sonatype. Sonatype Central Repository. `https://maven.org`.

[5] RubyGems. RubyGems. `https://rubygems.org/`.

[6] Joseph Feller and Brian Fitzgerald. *Understanding open source software development*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[7] Rajdeep Kaur, Kuljit Kaur Chahal, and Munish Saini. Understanding community participation and engagement in open source software projects: A systematic mapping study. *J. King Saud Univ. Comput. Inf. Sci.*, 34:4607–4625, 2020.

[8] Georg J. P. Link. The value of engaging with open source communities. In *Proceedings of the 13th International Symposium on Open Collaboration Companion*, 2017.

[9] Mariam Guizani, Thomas Zimmermann, Anita Sarma, and Denae Ford. Attracting and retaining oss contributors with a maintainer dashboard. In

*Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society*, pages 36–40, 2022.

[10] J. Bosch. Software ecosystems: Taking software development beyond the boundaries of the organization. *J. Syst. Softw.*, 85:1453–1454, 2012.

[11] R. Schreiber. Organizational influencers in open-source software projects. *International Journal of Open Source Software and Processes*, 2023.

[12] Simone da Silva Amorim, John D McGregor, Eduardo Santana de Almeida, and Christina von Flach G. Chavez. Software ecosystems architectural health: Challenges x practices. In *Proccedings of the 10th European Conference on Software Architecture Workshops*, pages 1–7, 2016.

[13] Amel Charleux and Robert Viseur. Exploring impacts of managerial decisions and community composition on the open source projects' health. In *Proceedings of the 2nd International Workshop on Software Health*, pages 1–8, 2019.

[14] Dominik Wermke et al. Committed to trust: A qualitative study on security & trust in open source software projects. *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1880–1896, 2022.

[15] Jonas Gamalielsson, Björn Lundell, and Anders Mattsson. Open source software for model driven development: a case study. In *IFIP International Conference on Open Source Systems*, pages 348–367. Springer, 2011.

[16] Yaxin Liu et al. Towards understanding developers' collaborative behavior in open source software ecosystems. *J. Softw.*, 12:393–405, 2017.

[17] Jonathan P. Allen. Democratizing business software: Small business ecosystems for open source applications. *Commun. Assoc. Inf. Syst.*, 30:28, 2012.

[18] Shohei Ikeda, Akinori Ihara, R. Kula, and Ken-ichi Matsumoto. An empirical study on readme contents for javascript packages. *ArXiv*, abs/1802.08391, 2018.

[19] N. Carvalho, Alberto Simões, and J. J. Almeida. Open source software documentation mining for quality assessment. *Comput. Sci. Inf. Syst.*, pages 785–794, 2013.

[20] Christopher Vendome, G. Bavota, M. D. Penta, M. Vásquez, D. Germán, and D. Poshyvanyk. License usage and changes: a large-scale study on github. *Empirical Software Engineering*, 22:1537–1577, 2017.

[21] Georgios Gousios, M. Storey, and Alberto Bacchelli. Work practices and challenges in pull-based development: the contributor's perspective. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, 2015.

[22] Jason Tsay, Laura A. Dabbish, and J. Herbsleb. Influence of social and technical factors for evaluating contribution in github. In *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014.

[23] Nicolas Bettenburg. Mining development repositories to study the impact of collaboration on software systems. *2011 44th Hawaii International Conference on System Sciences*, 2011.

[24] Carlos Jensen, Scott King, and Victor Kuechler. Joining free/open source software communities: An analysis of newbies' first interactions on project mailing lists. *2011 44th Hawaii International Conference on System Sciences*, 2011.

[25] Jenny Liang, T. Zimmermann, and Denae Ford. Understanding skills for oss communities on github. *Proceedings of the 30th ACM Joint European*

*Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022.

[26] S. L. T. Marín, M. R. Martínez-Torres, and F. Barrero. Analysis of virtual communities supporting oss projects using social network analysis. *Inf. Softw. Technol.*, 52:296–303, 2010.

[27] Rebeca Méndez-Durón and C. García. Returns from social capital in open source software networks. *Journal of Evolutionary Economics*, 19:277–295, 2009.

[28] Sherae L. Daniel, V. Midha, Anol Bhattacherjee, and Shivendu Pratap Singh. Sourcing knowledge in open source software projects: The impacts of internal and external social capital on project success. *J. Strateg. Inf. Syst.*, 27:237–256, 2018.

[29] Alexander Hars and Shaosong Ou. Working for free? motivations for participating in open-source projects. *International Journal of Electronic Commerce*, 6:25–39, 2002.

[30] Sherae L. Daniel, Ritu Agarwal, and K. Stewart. The effects of diversity in global, distributed collectives: A study of open source project success. *Inf. Syst. Res.*, 24:312–333, 2013.

[31] Bo Xu and Donald R. Jones. Volunteers' participation in open source software development: a study from the social-relational perspective. *Data Base*, 41:69–84, 2010.

[32] M. R. Martínez-Torres. Analysis of activity in open-source communities using social network analysis techniques. *Asian Journal of Technology Innovation*, 22:114–130, 2014.

[33] Sonali K. Shah. Motivation, governance, and the viability of hybrid forms in open source software development. *Manag. Sci.*, 52:1000–1014, 2006.

[34] Sebastiano Panichella. Supporting newcomers in software development projects. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2015.

[35] Shahab Bayati. Understanding newcomers success in open source community. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, pages 224–225, 2018.

[36] Igor Steinmacher et al. Why do newcomers abandon open source software projects? In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, 2013.

[37] Kévin D. Carillo, S. Huff, and B. Chawner. What makes a good contributor? understanding contributor behavior within large free/open source software projects - a socialization perspective. *J. Strateg. Inf. Syst.*, 26:322–359, 2017.

[38] Stackoverflow. Stack overflow 2023 developer survey: Most popular technologies programming scripting and markup languages. `https://survey.stackoverflow.co/2023/#section-most-popular-technologies-programming-scripting-and-markup-languages`, 2023. Accessed: 2024-01-12.

[39] Travis E Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

[40] Wes McKinney. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56, 2010.

[41] Michael Waskom. Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[42] Parastou Tourani, Bram Adams, and Alexander Serebrenik. Code of conduct in open source projects. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 24–33. IEEE, 2017.

[43] Github. Github-enums. `https://docs.github.com/en/graphql/reference/enums#commentauthorassociation`, 2024. Accessed: 2024-01-19.

[44] Karni Lotan Marcus. The pyramid fallacy: Self-organizing decentralized open systems for sustainable collective action. *SAGE Open*, 8(2):2158244018778322, 2018.

[45] Stefan Kambiz Behfar, Ekaterina Turkina, and Thierry Burger-Helmchen. Network tie structure causing oss group innovation and growth. *Problems and perspectives in management*, 15(15, Iss. 1):7–18, 2017.

[46] Oliver Arafat and Dirk Riehle. The commit size distribution of open source software. In *2009 42nd Hawaii International Conference on System Sciences*, pages 1–8. IEEE, 2009.

[47] Carsten Kolassa, Dirk Riehle, and Michel A Salim. The empirical commit frequency distribution of open source projects. In *Proceedings of the 9th international symposium on open collaboration*, pages 1–8, 2013.

[48] S. Patro and K. K. Sahu. Normalization: A preprocessing stage. *ArXiv*, abs/1503.06462, 2015.

[49] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[50] Minghui Zhou, A. Mockus, Xiujuan Ma, Lu Zhang, and Hong Mei. Inflow and retention in oss communities with commercial involvement. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 25:1 – 29, 2016.

[51] Mehvish Rashid, Paul M. Clarke, and Rory V. O'Connor. A mechanism to explore proactive knowledge retention in open source software communities. *Journal of Software: Evolution and Process*, 32, 2020.

[52] Kazuhiro Yamashita, Yasutaka Kamei, Shane McIntosh, Ahmed E. Hassan, and Naoyasu Ubayashi. Magnet or sticky? measuring project characteristics from the perspective of developer attraction and retention. *J. Inf. Process.*, 24:339–348, 2016.

[53] Felipe Fronchetti, Igor Wiese, Gustavo Pinto, and Igor Steinmacher. What attracts newcomers to onboard on oss projects? tl; dr: Popularity. In *Open Source Systems: 15th IFIP WG 2.13 International Conference, OSS 2019, Montreal, QC, Canada, May 26–27, 2019, Proceedings 15*, pages 91–103. Springer, 2019.

[54] Jaswinder Singh, Anu Gupta, and Preet Kanwal. The vital role of community in open source software development: A framework for assessment and ranking. *Journal of Software: Evolution and Process*, page e2643, 2023.

[55] Israr Qureshi and Yulin Fang. Socialization in open source software projects: A growth mixture modeling approach. *Organizational Research Methods*, 14(1):208–238, 2011.

[56] W. Xiao et al. Recommending good first issues in github oss projects. In *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*. IEEE, 2022.

[57] X. Tan and M. Zhou. A first look at good first issues on github. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2020.

[58] GitHub. Encouraging helpful contributions to your project with labels. `https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/encouraging-helpful-contributions-to-your-project-with-labels`, 2024. Accessed: 2024-02-18.

[59] Amir Hossein Ghapanchi, Claes Wohlin, and Aybüke Aurum. Resources contributing to gaining a competitive advantage for open source software projects. *ArXiv*, abs/1401.5712, 2014.

[60] Cal Poly Github. `http://www.github.com/CalPoly`.

[61] Wiki. Software ecosystem. `https://en.wikipedia.org/wiki/Software_ecosystem`.