Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 4, August 2024, pp. 163-175

DOI: https://doi.org/10.52436/1.jutif.2024.5.4.2143
p-ISSN: 2723-3863
e-ISSN: 2723-3871

UNDERSTANDING THE TRENDS OF DEVELOPER CONTRIBUTIONS IN NUGET, PYPI, AND NPM ECOSYSTEMS

Naufal Dzakia Raiffaza*[1], Yusuf Sulistyo Nugroho[2]

[1,2]Informatic, Faculty of Engineering, Universitas Muhammadiyah Surakarta, Indonesia
Email: [1]l200204248@student.ums.ac.id, [2]yusuf.nugroho@ums.ac.id

(Article received: May 27, 2024; Revision: June 11, 2024; published: July 29, 2024)

Abstract

Open-source software (OSS) projects have taken the software development industry rapidly by encouraging cooperation, creativity, and knowledge exchange. However, despite the widespread adoption and success of OSS, there is limited understanding of how contributions are distributed across different types of activities, such as code, documentation, and issue triage, and how these contributions vary over time within different OSS ecosystems. This gap in knowledge can impact effective project management and community engagement strategies. To address this problem, we aim to look into the patterns of developer contributions within the three main OSS ecosystems hosted on GitHub, namely NuGet, PyPI, and NPM. We examine the distribution of type-based contribution and the trends of developer activities within these ecosystems. We classify contributions into code, documentation, and issue triage using a mixed-methods approach that combines content analysis and time-series analysis, and we analyze the timeline variations in contribution trends. Our results show differences in pull request activity and developer contribution patterns between ecosystems. The 'npm-expansion' repository leads in open pull requests, while the 'warehouse' repository in PyPI dominates closed pull requests. The NPM ecosystem shows the highest number of activities when it comes to open pull requests. Notable peaks can be seen in trends in code development and maintenance activities, indicating the changing priorities of various projects. The findings shed light on the changes of OSS contributions and highlight the value of varied roles and ongoing community involvement. The comprehension of contribution patterns in open-source software projects is improved by this study, which also provides guidance for project management, resource allocation, and community support strategies. The knowledge acquired can direct the creation of instruments and systems that support more cooperative and productive open-source software ecosystems.

Keywords: *contributions, developers, NPM, NuGet, PyPI.*

1. INTRODUCTION

The impact of open-source software (OSS) projects on the software development industry is significant, as they foster collaboration, innovation, and knowledge sharing among individuals. The developers who contribute their time, knowledge, and experience to open-source software (OSS) projects are the motivation behind their growth and sustainability [1], [2], [3]. The success of these projects depends on the active involvement and various contributions of developers, including tasks such as coding, documenting, issue reviewing, and giving support [4], [5], [6], [7], [8].

As the advancement of OSS projects grows, it is crucial to understand the patterns of developer contributions. This is particularly true for software ecosystems such as NuGet, PyPI, and NPM, which serve as primary hubs for developers to distribute and apply reusable software components [9], [10], [11], [12], [13]. Many developers with various backgrounds have contributed to these ecosystems on a collaborative coding platform, GitHub.

GitHub, as a collaborative platform for software projects, has become important in current software development processes. They facilitate code reuse, connection management, and collaboration [14], [15]. Moreover, platforms like GitHub not only function as a location for collaborating on code, but also provide a way for financially supporting open-source software (OSS) developers with features like GitHub Sponsors that motivate developers to participate and contribute more to the projects [16], [17], [18]. Furthermore, the developer community on GitHub illustrates the variety and complicated nature of support within the open-source ecosystem [19][20].

Several previous works have investigated various aspects of OSS development on GitHub, including factors that influence motivation, patterns of collaboration, and elements that contribute to project effectiveness [21]. Although they found new insights related to the contribution characteristics of developers, there is still a need for a full analysis of what developers have added to different package repositories. Agrawal et al. [22] studied the impact of "hero" developers, highlighting the risks of over-reliance on key contributors. Bosu et al. [23] explored

the motivations and challenges of blockchain developers, providing strategies for sustaining long-term ecosystem health.

In addition, Casalnuovo et al. [24] analyzed GitHub developer onboarding, emphasizing the importance of language proficiency and existing relationships in successful integration. Constantinou and Mens [25] examined how social and technological factors shape the Ruby ecosystem on GitHub, aligning with our study's goals. Furthermore, Kononenko et al. [26] identified crucial elements of code reviews, while Ortu et al. [27] found that impolite requests can delay agile software development projects. Joblin et al. [28] classified developers into core and peripheral groups, and Dias et al. [29] investigated the roles of personnel and volunteers in company-owned OSS projects, increasing our understanding of diverse contribution types.

Despite the valuable insights provided by previous studies, there is limited understanding of how contributions are distributed across different types of activities (such as code, documentation, and issue triage) and how these contributions vary over time within different OSS ecosystems. This gap in knowledge may impact effective project management and community engagement strategies. Therefore, this study aims to fill this gap by examining the actual contributions made by developers to NuGet, PyPI, and NPM repositories on GitHub. Specifically, we investigate the distribution of type-based documentation and how developer contribution patterns vary over time within these ecosystems.

To guide our study, we formulate two main research questions:

1. What is the distribution of type-based documentation across NuGet, PyPI, and NPM ecosystem?
2. How do developer contribution patterns vary over time in NuGet, PyPI, and NPM ecosystem?

The results show differences in developer contributions and pull request activity across NuGet, PyPI, and NPM ecosystems. The number of open and closed pull requests varies among repositories, with NPM's 'npm-expansion' having the most open pull requests and PyPI's warehouse having the most closed ones. The trend analysis shows significant changes in code development and maintenance activity over time. Some repositories, like 'NuGet.Client' and 'NuGetGallery', peaked in certain years, while others, such as 'NuGet.Server', declined. PyPI's 'warehouse' showed variations and increases, and NPM's 'cli' peaked in 2021, with 'node-semver' and 'npm-expansion' showing significant fluctuations over the period.

The findings of this research provide an understanding of the various methods through the developers who contribute to open-source software (OSS) projects and how these contributions changed during the development process. This greatly impacts managers, maintainers, and users who contribute to open-source software (OSS) projects. Furthermore, the knowledge acquired from this research can direct the development of tools and platforms that improve and facilitate various forms of contributions, leading to more active and collaborative OSS ecosystems [30], [31].

2. METHODS

This study applies a mixed-methods approach to investigate the distribution and temporal patterns of developer contributions across NuGet, PyPI, and NPM ecosystems hosted on GitHub. Content analysis is used to categorize contributions into types, that are code, documentation, and issue triage, by examining commit messages, pull requests, and issue interactions. Time-series analysis is used to examine the fluctuations in contribution trends across the lifespan of repositories. The dataset is collected from GitHub using the API and a tool, PyDriller [32].

In detail, we present this section including research questions, data collection, content analysis, and an online appendix.

2.1. Research Questions

2.1.1. RQ1. What is the distribution of type-based documentation across NuGet, PyPI, and NPM ecosystem?

Understanding the types of contributions made by developers in different ecosystems can help to highlight their unique roles. The distribution of contribution categories across the NuGet, PyPI, and NPM ecosystems provides valuable insights into each community's emphasis and priorities. For instance, an ecosystem may prioritize feature development and implementation if it obtains contributions that are more code-centric than documentation modifications. Conversely, a community that prioritizes user support and problem fixes may be indicated by a higher percentage of contributions related to issue analysis and management. By analyzing these trends, project maintainers can more effectively align their resource allocation, recognition programs, and communication methods with the specific needs and dynamics of their own ecosystems.

2.1.2. RQ2: How do developer contribution patterns vary over time in NuGet, PyPI, and NPM ecosystem?

Examining patterns in developer contributions can give important information about the development and status of open-source projects. Through monitoring the changes in contribution patterns over time, we can identify possible fluctuations in contributor activity that could point to times of higher community involvement or possible difficulties. For example, an increasing number of code contributions over time may indicate an

established developer community and a healthy project ecosystem. Understanding these patterns can help project managers make better decisions about resource allocation, release planning, and community. By proactively addressing changes in contributions, project leaders may develop a more sustainable and stronger open-source community.

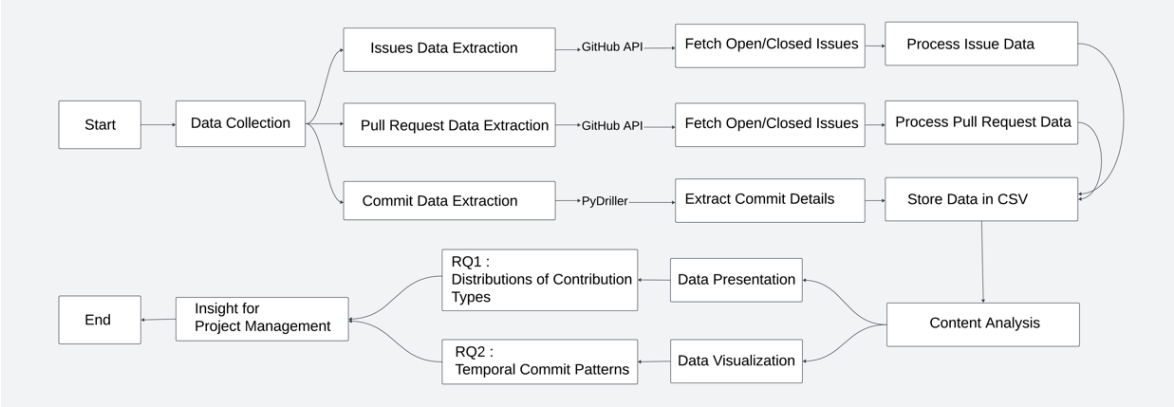


Figure 1. The procedure of our study, from data collection to insight extraction

2.2. Data Collection

To facilitate our analysis, we extracted the data from three most popular repositories based on star ratings within the NuGet, PyPI, and NPM on GitHub [33]. This process involves the GitHub API to obtain relevant information, such as commit messages, pull requests, and issue interactions, as shown in Figure 1.

To collect the commit messages, we utilized the PyDriller tool to extract the commit history of each repository. PyDriller enables the extraction of crucial information such as the commit hash, message, author, year, and the count of lines added and removed for each updated file [32]. In addition, to extract issue data and pull requests, we used the GitHub API to fetch both open and closed issues for each repository. This collected issue information includes title, tags, status (open/closed), and corresponding URLs. Similarly, we collect the title,status, date, year, and the corresponding URL for pull requests and save it to csv file.

This dataset is used in our investigation of contribution patterns and time-series analysis. It allows us to investigate the research questions and obtain valuable information about the distribution and temporal trends of contribution types in the NuGet, PyPI, and NPM ecosystems.

2.3. Content Analysis

2.3.1. RQ1. What is the distribution of type-based documentation across NuGet, PyPI, and NPM ecosystem?

To solve this RQ, we analyzed commit messages, pull requests, and issue interactions from the three most popular NuGet, PyPI, and NPM projects based on GitHub stars. This study will involve utilizing automated techniques to retrieve and categorize data from GitHub pull requests, including information about code changes, documentation, and issues [33], [34]. The automated process consists of fetching data from the GitHub API and organizing it into structured formats such as CSV files and these techniques enable efficient and scalable analysis of large datasets by extracting relevant metadata.

Contributions to source code were detected by looking at commit messages and pull requests that included changes to files with programming language extensions (e.g. *.cs* for C#, *.py* for Python, *.js* for JavaScript). Documentation contributions were recognized through commits and pull requests that modified files commonly associated with documentation, such as README files, wiki pages, and files ending in *.md* or *.rst*.

We separated open and closed issues based on their state on GitHub. Open issues are ones that are still unsolved and require action, whereas closed issues have been fixed or addressed. Similarly, pull requests were classified as open if they were still for review or merging, and closed if they had already been merged or closed without merging [35].

To find the distribution of type-based documentation across the three ecosystems, we quantified and analyzed the proportions of different contribution categories among three repositories by reviewing the data of contributions from each of them.

2.3.2. RQ2: How do developer contribution patterns vary over time in NuGet, PyPI, and NPM ecosystem?

To investigate RQ2, we built a timeline of contributions over the lifespan of each selected repository. This analysis involved tracking the average commit lines added and lines deleted. By aggregating this data, we generated a visualization that shows the evolution of contribution patterns within each ecosystem. Furthermore, we also applied time-series analysis techniques to uncover recurring patterns [36]. Using data visualization, we aimed to depict trends and fluctuations in contribution activity,

such as the annual changes in lines of code added or deleted , [37], [38], [39]. The goal is to more easily understand the pattern of developer contributions in the three analyzed OSS projects (i.e. NuGet, PyPI, and NPM).

With the use of this method, we were able to identify the times when contributor interaction was highest or lowest as well as any periodic trends. By comparing the patterns across NuGet, PyPI, and NPM repositories, we gain insights into how each ecosystem changed over time, informing strategies for community management and sustainability.

2.4. Appendix

To facilitate the replication and reproducibility, the datasets used in this study and the results are made publicly available on GitHub at https://github.com/raiffaza/Havia-Research.

3. RESULTS

3.1. RQ1. What is the distribution of type-based documentation across NuGet, PyPI, and NPM ecosystem?

3.1.1. Commit and Issue

Table 1 compares the activity and engagement levels across the NuGet, PyPI, and NPM ecosystems by analyzing the distribution of commits between source code and documentation, and the status of issues. The NuGet ecosystem shows high activity in the 'NuGet.Client' and 'NuGetGallery' repositories, with a focus on source code development, while 'NuGet.Server' has an almost balanced distribution at source code and documentation.

Table 1. Frequency of commits and issues from 9 repositories across 3 ecosystems

Ecosystem	Repository Name	Commit		Issues	
		Source Code	Documentation	Open	Closed
NuGet	Nuget.Client	71.08%	28.92%	-	-
NuGet	NuGet.Server	51.78%	48.22%	-	-
NuGet	NuGetGallery	60.99%	39.01%	11.58%	88.42%
PyPI	linehaul	66.81%	33.19%	50.00%	50.00%
PyPI	stdlib-list	36.91%	63.09%	13.16%	86.84%
PyPI	warehouse	37.27%	62.73%	12.92%	87.08%
NPM	cli	54.32%	45.68%	15.61%	84.39%
NPM	node-semver	49.91%	50.09%	11.11%	88.89%
NPM	npm-expansions	1.80%	98.20%	23.38%	76.62%

In the PyPI ecosystem, the 'warehouse' and 'stdlib-list' repositories are highly active with a significant focus on documentation and many closed issues, while 'linehaul' is less active in documentation. The NPM ecosystem features an almost balanced focus in the 'cli' and 'node-semver' repositories, with 'npm- expansion' placing a strong emphasis on documentation.

3.1.2. Pull Request

1) Open Pull Request

Table 2 shows the open pull requests in NuGet, PyPI, and NPM ecosystems. The data indicates varying levels of activity and contributions across different repositories within each ecosystem.

Table 2. The total open pull request of 9 repositories between 2011 and 2023

Year	NuGet			PyPI			NPM		
	NuGet. Client	NuGet. Server	NuGet. Gallery	Linehaul	Stdlib-list	Warehouse	cli	node-semver	npm-expansion
2011	-	-	-	-	-	-	-	-	-
2012	-	-	-	-	-	-	-	-	-
2013	-	-	-	-	-	-	-	-	-
2014	-	-	-	-	-	-	-	-	-
2015	-	-	-	-	-	-	-	-	-
2016	-	-	-	-	-	-	-	-	47
2017	-	-	-	-	-	-	-	-	181
2018	-	-	-	-	-	-	-	-	260
2019	-	-	-	2	-	9	-	-	249
2020	-	-	-	1	-	3	-	-	333
2021	-	-	-	2	-	2	-	-	307
2022	-	-	-	-	-	9	6	-	264
2023	1	-	3	-	-	20	10	2	352
Total	1	-	3	5	-	43	16	2	1,993

In detail, the 'NuGet.Client' repository in NuGet ecosystem had only 1 open pull request in 2023, while the 'NuGetGallery' had a total of 3 open pull requests in the same year. In contrast, the 'NuGet.Server' repository had no open pull requests throughout the entire period from 2011 to 2023.

The PyPI ecosystem shows limited activity, with the 'linehaul' repository receiving a total of 5 open pull requests between 2019 and 2021, but none in 2022 and 2023. The 'stdlib-list' repository had no open pull requests at all, while the 'warehouse'

repository had a total of 43 open pull requests, with a peak of 20 in 2023.

The NPM ecosystem, particularly the 'npm-expansion' repository, consistently had the highest number of open pull requests, with a total of 1,993 accumulated from 2016 to 2023, peaking at 333 in 2020 and 352 in 2023. The 'cli' repository had a total of 16 open pull requests, with an increase from 6 in 2022 to 10 in 2023. The 'node-semver' repository had only 2 open pull requests, both in 2023.

In general, the result demonstrates the highest activity and contributions in terms of open pull requests in each ecosystem. In the NuGet ecosystem, the 'NuGetGallery' shows the most activity although it is insignificant. The 'Warehouse' dominates the open pull request in the PyPI ecosystem with 43 activities, while 'npm-expansion' has shown the largest activity with 1,993 in the NPM ecosystem.

2) Closed Pull Request

Table 3 offers valuable information about the trends of closed pull requests in various repositories within the NuGet, PyPI, and NPM ecosystems.

Table 3. The total closed pull request of 9 repositories between 2011 and 2023

	NuGet			PyPI			NPM		
Year	NuGet. Client	NuGet. Server	NuGet Gallery	Linehaul	Stdlib-list	Warehouse	cli	node-semver	npm-expansion
2011	-	-	58	-	-	-	-	3	-
2012	-	-	112	-	-	-	-	3	-
2013	-	-	456	-	-	120	-	16	-
2014	-	-	177	2	-	112	-	22	217
2015	170	-	82	-	-	347	-	14	654
2016	907	-	178	17	-	551	-	11	513
2017	830	-	645	9	-	918	-	17	235
2018	662	-	560	11	7	1,769	129	19	193
2019	515	-	404	3	4	1,317	182	10	202
2020	645	-	263	1	8	1,438	339	15	106
2021	554	-	226	-	3	1,336	498	11	100
2022	611	-	210	-	2	1,773	866	47	105
2023	572	-	237	-	47	2,010	-	106	98
Total	5,466	-	3,382	43	71	11,691	2,014	297	2,423

Differently with the activity of the open pull request, the 'NuGet.Client' repository demonstrates the largest activity in the NuGet ecosystem, with a total of 5,466 closed pull requests. In addition, the 'NuGetGallery' shows a fluctuating number of closed pull requests from 2011 to 2023, reaching a total of 3,382 activities. However, the 'NuGet.Server' repository has no closed pull requests during the periods.

The 'warehouse' repository of the PyPI ecosystem shows the highest activities of closed pull requests, reaching a total of 11,691 in 2023. In contrast, the 'stdlib-list' and 'linehaul' repositories have relatively low pull request activity, with totals of 71 and 43 closed pull requests, respectively.

Within the NPM ecosystem, the 'cli' repository has experienced a consistent increase in the number of closed pull requests over time, reaching a total of 2,014 in 2023. The 'npm-expansion' repository also shows significant activity, with a total of 2,423 closed pull requests. The 'node-semver' repository has a relatively lower total of 297 closed pull requests.

These findings suggest that all repositories are actively maintained within their ecosystems. The 'warehouse' repository in PyPI shows significant activity, especially in recent years, which suggests continuous development from the community.

3.2. RQ2: How do developer contribution patterns vary over time in NuGet, PyPI, and NPM repositories?

3.2.1. NuGet

1) NuGet.Client

Figure 2 shows the average number of lines added and deleted per year during development activity on the NuGet.Client repository changes significantly between 2014 and 2023. The number of lines added peaked in 2015, and the number of lines deleted peaked in 2022. Overall, this graph shows the variation in code development and maintenance activities from year to year.

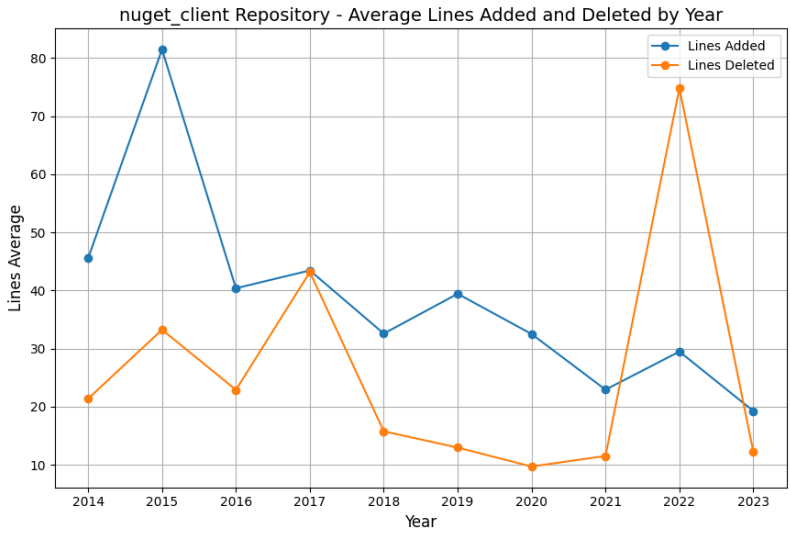
2) Nuget.Server

The data is presented in Figure 3 of the NuGet. Server repository indicates that the year 2017 had the highest level of activity. On average, 30.73 rows were added and 17.01 rows were deleted during this year. The activity was then progressively decreased since 2018, reaching only 4.88 lines added and 1.60 lines deleted on average by 2023. This suggests a decline in the rate of growth, which could mean that the repository has transitioned into a phase focused on maintenance.

3) NuGetGallery

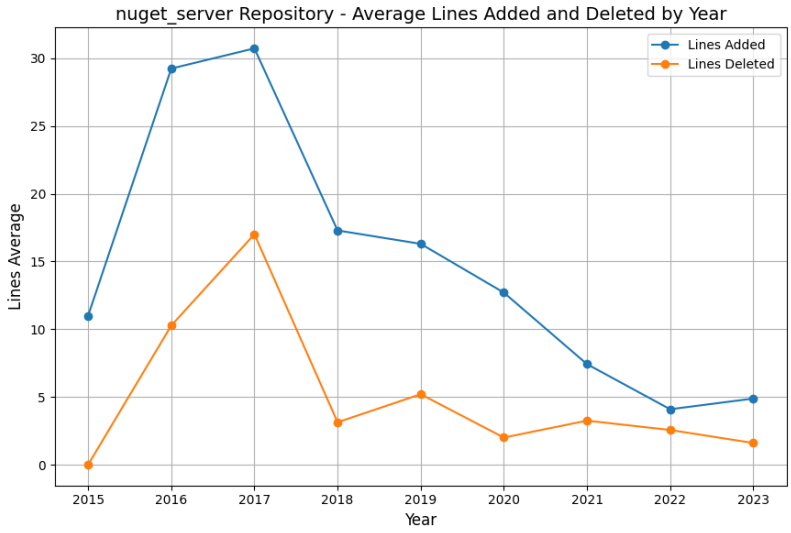
Figure 4 depicts the evolution of development activity on the NuGet Gallery repository from 2011 to 2023, emphasizing significant fluctuations in the yearly average of lines added and deleted. In 2013, the maximum number of lines added averaged around 138.48 lines, whereas in 2015, the maximum number

of lines deleted averaged around 369.48 lines. The graph illustrates significant fluctuations in code development and maintenance activity over the years, which could be attributed to changing project priorities or changing maintenance needs.



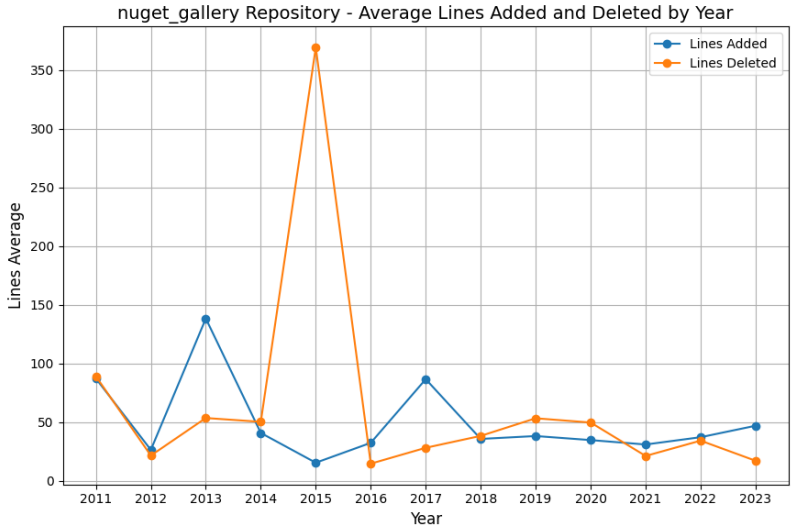
Year	Avg Lines Added	Avg Lines Deleted
2014	45.54	21.38
2015	81.45	33.25
2016	40.38	22.91
2017	43.45	43.16
2018	32.58	15.78
2019	39.43	12.98
2020	32.51	9.72
2021	22.94	11.54
2022	29.49	74.81
2023	19.28	12.28

Figure 2. Trends of average lines added and lines deleted in NuGet.Client from 2014 to 2023



Year	Avg Lines Added	Avg Lines Deleted
2015	11.00	0.00
2016	29.24	10.27
2017	30.73	17.01
2018	17.29	3.13
2019	16.30	5.20
2020	12.70	2.00
2021	7.42	3.25
2022	4.09	2.56
2023	4.88	1.60

Figure 3. Trends of average lines added and lines deleted in NuGet.Server from 2015 to 2023



Year	Avg Lines Added	Avg Lines Deleted
2011	87.38	89.23
2012	26.55	21.72
2013	138.48	53.71
2014	40.99	50.46
2015	15.61	369.48
2016	32.55	14.81
2017	86.78	28.34
2018	35.89	38.47
2019	38.35	53.48
2020	34.90	49.85
2021	31.14	21.38
2022	37.36	34.43
2023	47.12	17.29

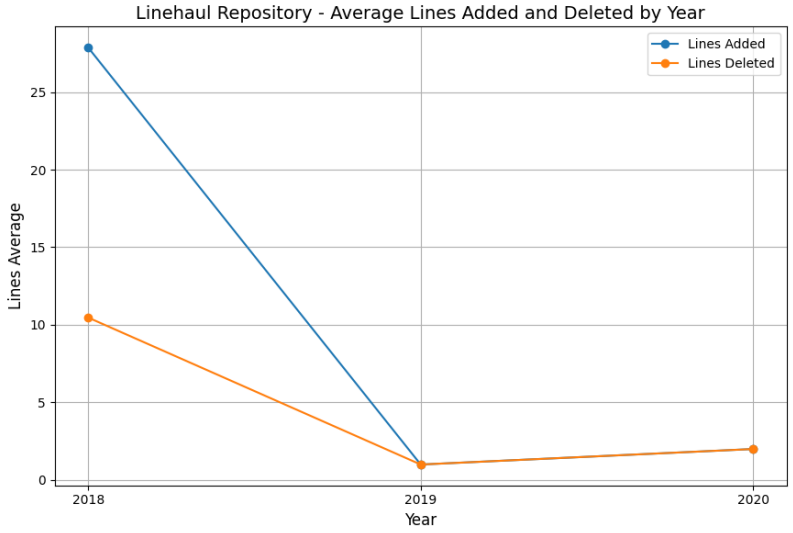
Figure 4. Trends of average lines added and lines deleted in NuGetGallery from 2011 to 2023

DOI: https://doi.org/10.52436/1.jutif.2024.5.4.2143
p-ISSN: 2723-3863
e-ISSN: 2723-3871

3.2.2. PyPI

1) Linehaul

Figure 5 illustrates the development activity on the Linehaul repository from 2018 to 2020. It shows significant fluctuations in the average number of lines added and deleted each year in the periods. The average number of lines added in 2018 was 27.87, whereas the average number of lines deleted was 10.47. In 2019, there was a decrease in both the average number of lines added and deleted, which reached a value of 1. In 2020, there was a small total change of 2 lines, both additions and deletions.

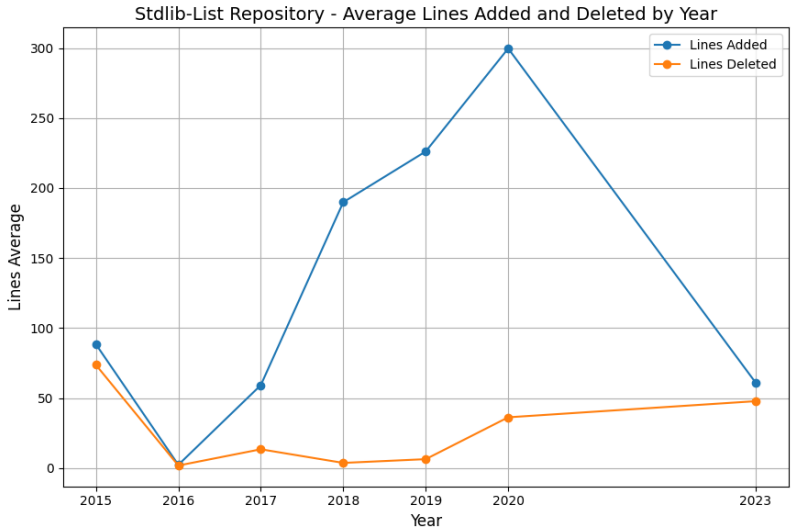


Year	Avg Lines Added	Avg Lines Deleted
2018	27.87	10.47
2019	1.00	1.00
2020	2.00	2.00

Figure 5. Trends of average lines added and lines deleted in Linehaul from 2018 to 2020

2) Stdlib-list

Figure 6 shows the activity on the Stdlib-List repository from 2015 to 2023. In 2015, the average number of lines added was 88.46, while the average number of lines deleted was 73.72. In 2020, the average number of lines added peaked at 299.69 lines, and the average number of lines deleted also increased to 36.27 lines. In 2023, the average number of lines added decreased to 61.08, while the average number of lines deleted reached 47.79. The absence of lines added and deleted in 2021 and 2022 suggests no code modifications as they are unnecessary.



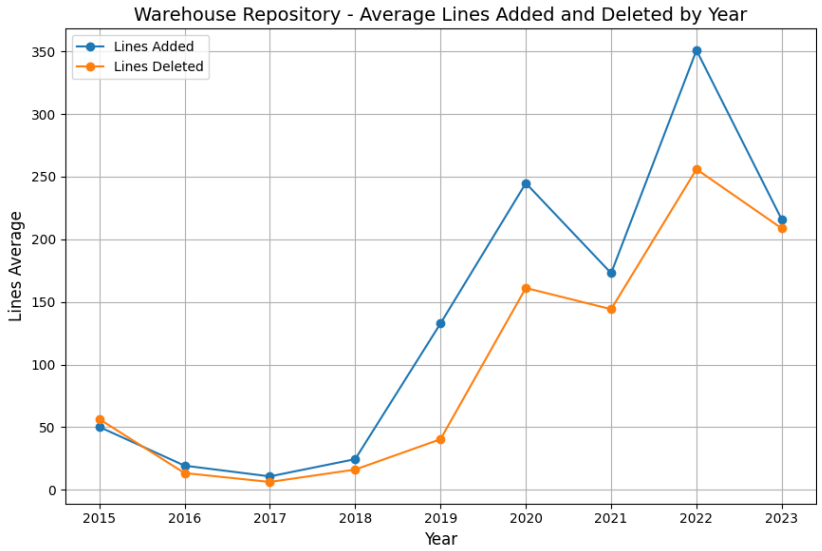
Year	Avg Lines Added	Avg Lines Deleted
2015	88.46	73.72
2016	2.33	1.83
2017	59.25	13.43
2018	189.90	3.71
2019	226.15	6.38
2020	299.69	36.27
2023	61.08	47.79

Figure 6. Trends of average lines added and lines deleted in Stdlib-list from 2015 to 2023

3) Warehouse

Figure 7 illustrates the development activity on the warehouse repository from 2015 to 2023 and shows significant fluctuations in the average number of lines added and deleted each year. In 2015, the average number of lines added was 50.20, while the average number of lines deleted was 56.45. The number of lines added peaked in 2022 with an average of 350.96 lines, while the number of lines deleted also peaked in 2022 with an average of 255.96 lines. In 2023, the average number of lines added was 215.73, while the average number of lines deleted was 208.71.

DOI: https://doi.org/10.52436/1.jutif.2024.5.4.2143
p-ISSN: 2723-3863
e-ISSN: 2723-3871



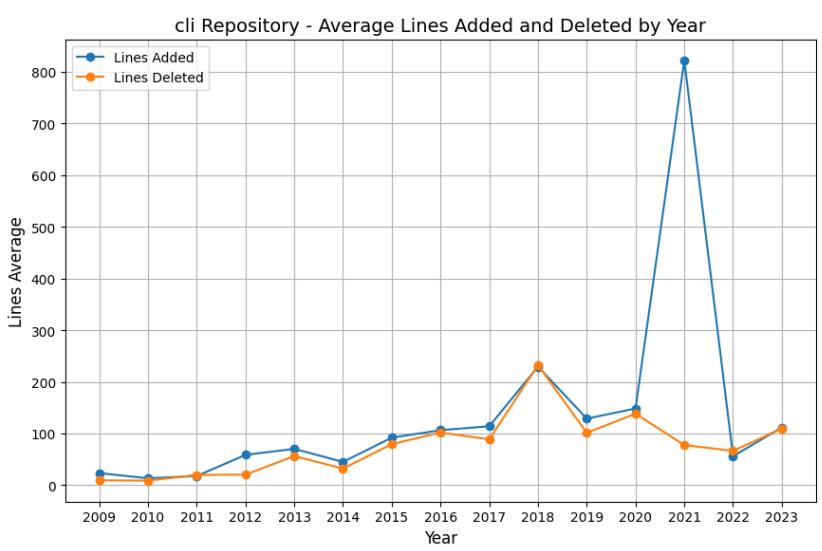
Year	Avg Lines Added	Avg Lines Deleted
2015	50.20	56.45
2016	19.25	13.31
2017	10.65	6.22
2018	24.43	16.12
2019	132.79	40.34
2020	244.83	161.13
2021	172.95	144.19
2022	350.96	255.96
2023	215.73	208.71

Figure 7. Trends of average lines added and lines deleted in Warehouse from 2015 to 2023

3.2.3. NPM

1) Cli

Based on Figure 8, the development activity on the cli repository from 2009 to 2023 shows significant fluctuations in the average number of lines added and deleted each year. In 2009, the average number of lines added was 23.44, while the average number of lines deleted was 9.45. The number of lines added peaked in 2021 with an average of 822.21 lines, while the number of lines deleted peaked in 2018 with an average of 232.48 lines. In 2023, the average number of lines added was 111.73, while the average number of lines deleted was 109.37.



Year	Avg Lines Added	Avg Lines Deleted
2009	23.44	9.45
2010	13.44	8.91
2011	17.77	19.90
2012	58.69	20.46
2013	70.32	56.43
2014	45.15	32.23
2015	91.99	79.56
2016	106.50	101.92
2017	114.09	88.79
2018	229.46	232.48
2019	128.83	101.42
2020	148.43	138.43
2021	822.21	77.59
2022	56.12	66.52
2023	111.73	109.37

Figure 8. Trends of average lines added and lines deleted in Cli from 2009 to 2023

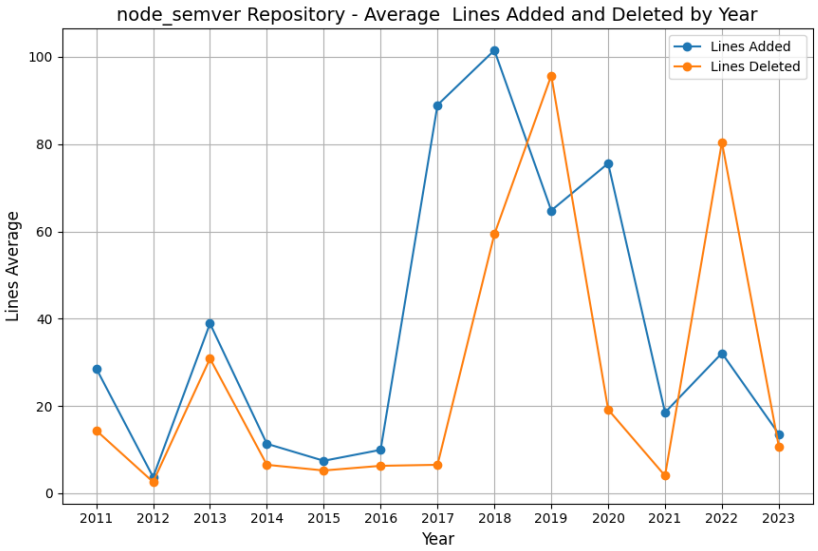
2) Node-semver

Figure 9 illustrates the development activity on the node-semver repository from 2011 to 2023, with a significant variation in the average number of lines added and deleted per year. In 2011, the average number of lines added was 28.60, and the average number of lines removed was 14.38. The number of lines added peaked in 2018 at 101.51 lines, while the number of lines deleted peaked in 2019 at 95.72 lines. In 2023, the average amount of lines added was 13.58, while the average number of lines removed was 10.71.

3) Npm-expansion

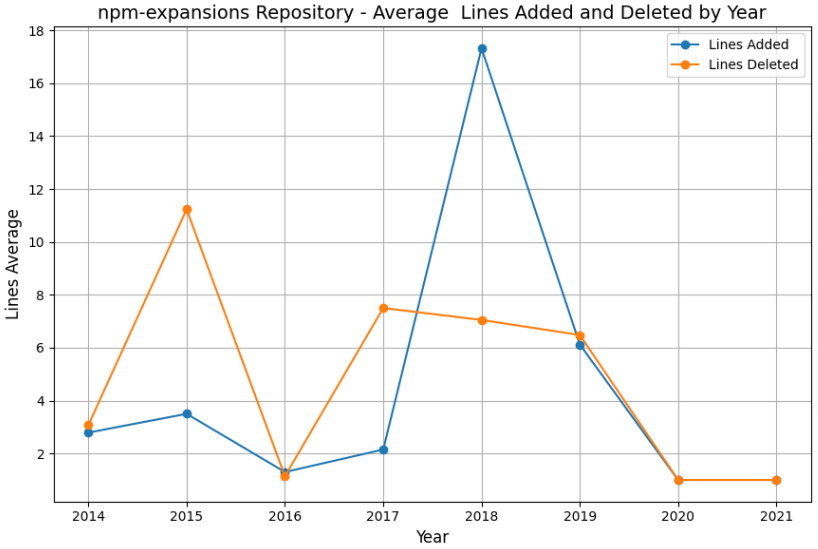
In Figure 10, the development activity on the npm-expansions repository from 2014 to 2021 shows significant variation in the average number of lines added and deleted each year. In 2014, the average number of lines added was 2.79, while the average number of lines deleted was 3.07. The number of lines added peaked in 2018 with an average of 17.32 lines, and the number of lines deleted peaked in 2015 with an average of 11.23 lines. In 2021, the average of added and deleted lines was only 1.

DOI: https://doi.org/10.52436/1.jutif.2024.5.4.2143
p-ISSN: 2723-3863
e-ISSN: 2723-3871



Year	Avg Lines Added	Avg Lines Deleted
2011	28.60	14.38
2012	3.70	2.57
2013	38.87	30.85
2014	11.32	6.50
2015	7.42	5.20
2016	9.93	6.27
2017	88.97	6.50
2018	101.51	59.35
2019	64.79	95.72
2020	75.59	19.16
2021	18.50	4.00
2022	32.09	80.43
2023	13.58	10.71

Figure 9. Trends of average lines added and lines deleted in Node-semver from 2011 to 2023



Year	Avg Lines Added	Avg Lines Deleted
2014	2.79	3.07
2015	3.50	11.23
2016	1.30	1.14
2017	2.15	7.50
2018	17.32	7.05
2019	6.12	6.48
2020	1.00	1.00
2021	1.00	1.00

Figure 10. Trends of average lines added and lines deleted in Npm-expansion from 2014 to 2021

4. DISCUSSION

The empirical study of developer contributions across NuGet, PyPI, and NPM repositories has revealed interesting results requiring for a comprehensive discussion. Following the answers to both research questions (RQ1 and RQ2), this section will explore what the results mean by looking at the types of contributions and the patterns of when developers make them.

4.1. RQ1. What is the distribution of type-based documentation across NuGet, PyPI, and NPM ecosystem?

The analysis of the distribution of contribution types across the three ecosystems reveals distinct patterns. The NuGet 'NuGetGallery' repository had a significant number of closed pull requests, totaling 3,382 by 2023, indicating an active community committed to resolving issues and improving the project. Similarly, the 'NuGet.Client' repository had 5,466 closed pull requests, suggesting substantial community engagement.

The PyPI 'warehouse' repository stood out with the highest number of closed pull requests at 11,691, along with a considerable number of commits for both source code and documentation. This represents an active community dedicated to progress and user support, placing high value on clear and understandable documentation.

The NPM 'cli' repository had a balanced distribution of source code and documentation contributions, with 2,014 closed pull requests. The 'npm-expansion' repository had the highest number of open pull requests at 1,993, indicating ongoing development and community involvement.

The study found varying contribution patterns in three ecosystems. NuGet's 'NuGetGallery' and 'NuGet.Client' had high community involvement with many closed pull requests. PyPI's 'warehouse'

had the most closed pull requests, showing an active community supporting users and documentation. Similar studies have also found that corporate involvement in OSS projects can improve project sustainability with significant contributions from paid developers, which is in line with our findings on the importance of community contributions in the OSS ecosystem [40]. This aligns with our findings on the importance of community contributions in the OSS ecosystem, highlighting how both community and corporate contributions play crucial roles in sustaining OSS projects.

4.2. RQ2: How do developer contribution patterns vary over time in NuGet, PyPI, and NPM repositories?

An analysis of trends of developer contributions provides insight into the development and condition of the repositories as time progressesActivity level fluctuations can indicate the stages of the project's lifecycle, including initial development, feature expansion, and maintenance. The repositories 'node-semver' and 'npm-expansions' exhibited different degrees of activity, with significant increases during specific years. These variations may be linked to particular project goals, such as the launch of new versions or the complete update of existing features, which can be studied in future research.

Understanding these patterns can help maintainers guess what will be needed in the future and make sure that resources are used in the most efficient way. The 'NuGet.Client' and 'NuGetGallery' repositories for NuGet showed changing patterns, with periods of intense coding followed by periods of stability. The high number of average lines that were deleted from "NuGet.Client" in 2022 points to a major refactoring project. This could be part of a larger plan to make the codebase easier to maintain and faster. The highest number of new code contributions to the PyPI "warehouse" repository happened in 2022, and then they started going down. This could mean that the phase of active development is changing to one of maintenance. This change is common in projects that are well-developed and have all the features they need.

This research shows time-varying patterns of developer contributions in the NuGet, PyPI, and NPM repositories, reflecting project lifecycle stages such as initial development, feature expansion, and maintenance. These results are in line with previous research which found that fluctuations in developer activity can indicate important phases in the open-source project lifecycle. For instance, a study of projects on GitHub identified similar patterns in developer contributions during the development and maintenance phases, reinforcing our findings [41].

4.3. Implication for OSS Ecosystems

The findings from this empirical analysis have various implications for the administration and long-term viability of open-source software ecosystems. Project maintainers can utilize the knowledge of contribution distributions to customize their approaches for community involvement, acknowledgment, and allocation of resources. For example, ecosystems that get a lot of documentation contributions might do well to spend money on better documentation tools and programs that reward contributors. Finding times when activity is high or low can help maintainers deal with potential problems earlier than they develop, like contributors getting tired or fewer people getting involved in the group.

Furthermore, the outcomes can provide guidance for the creation of tools and platforms that facilitate various forms of contributions, promoting greater participation and cooperation within open-source software communities. Through understanding the type of contributions, projects may attract in a wider range of participants while motivating continued involvement.

5. CONCLUSION

This research analyzes developer contributions to the NuGet, PyPI, and NPM repositories to understand the dynamics and future growth of the open-source software (OSS) ecosystem. The findings of this study show clear patterns in the distribution of contribution types, with the NuGet repository emphasizing more source code contributions, the PyPI 'warehouse' repository standing out in source code and documentation contributions and having many closed issues, while the NPM 'cli' repository showing a balanced distribution between source code and documentation contributions. The trend analysis reveals there were an increase in activity and significant fluctuations, such as an increase of added lines in the 'cli' repository in 2021 and a peak in code deletions in 2018. These findings suggest a stage characterized by fast development and changes. Further research is needed to explore the relationship between contribution types and project stages, and their impact on the sustainability of the OSS ecosystem. More in-depth case studies on specific repositories can provide more insights into the variables that cause these trends, and how they could be handled to maintain the long-term viability and growth of OSS projects.

DAFTAR PUSTAKA

[1] Z. Fang *et al.*, "A Four-Year Study of Student Contributions to OSS vs. OSS4SG with a Lightweight Intervention," in *ESEC/FSE 2023 - Proceedings of the 31st ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*,

Association for Computing Machinery, Inc, Nov. 2023, pp. 3–15. doi: 10.1145/3611643.3616250.

[2] M. AlMarzouq, V. Grover, J. Thatcher, and R. Klein, "An empirical examination of newcomer contribution costs in established OSS communities: a knowledge-based perspective," *Internet Research*, vol. 34, no. 3, pp. 665–689, Jan. 2024, doi: 10.1108/INTR-08-2022-0594.

[3] B. Vasilescu, V. Filkov, and A. Serebrenik, "Perceptions of Diversity on Git Hub: A User Survey," in *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, 2015, pp. 50–56. doi: 10.1109/CHASE.2015.14.

[4] 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*, vol. 4, no. CSCW2, Oct. 2020, doi: 10.1145/3415251.

[5] D. Rozas, N. Gilbert, P. Hodkinson, and S. Hassan, "Talk Is Silver, Code Is Gold? Beyond Traditional Notions of Contribution in Peer Production: The Case of Drupal," *Frontiers in Human Dynamics*, vol. 3, 2021, doi: 10.3389/fhumd.2021.618207.

[6] C. Osborne, "Public-private funding models in open source software development: A case study on scikit-learn," Apr. 2024, [Online]. Available: http://arxiv.org/abs/2404.06484

[7] L. F. Dias *et al.*, "Refactoring from 9 to 5? What and When Employees and Volunteers Contribute to OSS," in *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2020, pp. 1–5. doi: 10.1109/VL/HCC50065.2020.9127205.

[8] G. Russo Latona, C. Gote, C. Zingg, G. Casiraghi, L. Verginer, and F. Schweitzer, "Shock! Quantifying the Impact of Core Developers' Dropout on the Productivity of OSS Projects," in *Companion Proceedings of the ACM on Web Conference 2024*, in WWW '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 706–709. doi: 10.1145/3589335.3651559.

[9] R. Abdalkareem, O. Nourry, S. Wehaibi, S. Mujahid, and E. Shihab, "Why do developers use trivial packages? An empirical case study on npm," in *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Association for Computing Machinery, Aug. 2017, pp. 385–395. doi: 10.1145/3106237.3106267.

[10] Z. Chen, W. Ma, L. Chen, and W. Song, "Collaboration in software ecosystems: A study of work groups in open environment," *Inf Softw Technol*, vol. 145, p. 106849, 2022, doi: https://doi.org/10.1016/j.infsof.2022.106849.

[11] A. G. Young, A. Majchrzak, and G. C. Kane, "Reflection on writing a theory paper: How to theorize for the future," *Journal of the Association for Information Systems*, vol. 22, no. 5. Association for Information Systems, pp. 1212–1223, 2021. doi: 10.17705/1jais.00712.

[12] K. Kannee, S. Wattanakriengkrai, R. Rojpaisarnkit, R. G. Kula, and K. Matsumoto, "Intertwining Ecosystems: A Large Scale Empirical Study of Libraries that Cross Software Ecosystems," Aug. 2022, [Online]. Available: http://arxiv.org/abs/2208.06655

[13] P. Nidhi Sharma, S. L. Daniel, T. Chung, and V. Grover, "A Motivation-Hygiene Model of Open Source Software Code Contribution and Growth," *J Assoc Inf Syst*, vol. 23, no. 1, pp. 165–195, 2022, doi: 10.17705/1jais.00712.

[14] R. Kikas, G. Gousios, M. Dumas, and D. Pfahl, "Structure and Evolution of Package Dependency Networks," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 2017, pp. 102–112. doi: 10.1109/MSR.2017.55.

[15] A. Decan, T. Mens, and P. Grosjean, "An Empirical Comparison of Dependency Network Evolution in Seven Software Packaging Ecosystems," *Empir Softw Eng*, vol. 24, Jun. 2019, doi: 10.1007/s10664-017-9589-y.

[16] P. K. Medappa, M. M. Tunc, and X. Li, "Sponsorship Funding in Open-Source Software: Effort Reallocation and Spillover Effects in Knowledge-Sharing Ecosystems," *SSRN Electronic Journal*, 2023, doi: 10.2139/ssrn.4484403.

[17] Y. Wang, L. Wang, H. Hu, J. Jiang, H. Kuang, and X. Tao, "The Influence of Sponsorship on Open-Source Software Developers' Activities on GitHub," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2022, pp. 924–933. doi: 10.1109/COMPSAC54236.2022.00144.

[18] Y. Fan, T. Xiao, H. Hata, C. Treude, and K. Matsumoto, "'My GitHub Sponsors profile is live!' Investigating the Impact of Twitter/X Mentions on GitHub Sponsors," in *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering* (ICSE), 2024. doi:

10.1145/3597503.3639127.

[19] N. Shimada, T. Xiao, H. Hata, C. Treude, and K. Matsumoto, "GitHub Sponsors: Exploring a New Way to Contribute to Open Source," in *Proceedings - International Conference on Software Engineering*, IEEE Computer Society, 2022, pp. 1058–1069. doi: 10.1145/3510003.3510116.

[20] H. S. Qiu, A. Nolte, A. Brown, A. Serebrenik, and B. Vasilescu, "Going Farther Together: The Impact of Social Capital on Sustained Participation in Open Source," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 2019, pp. 688–699. doi: 10.1109/ICSE.2019.00078.

[21] H. S. Qiu, Y. L. Li, S. Padala, A. Sarma, and B. Vasilescu, "The Signals that Potential Contributors Look for When Choosing Open-source Projects," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW. Association for Computing Machinery, Nov. 01, 2019. doi: 10.1145/3359224.

[22] A. Agrawal, A. Rahman, R. Krishna, A. Sobran, and T. Menzies, "We don't need another hero? the impact of 'heroes' on software development," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice* (ICSE-SEIP '18), 2018, pp. 245–253. doi: 10.1145/3183519.3183549.

[23] A. Bosu, R. Shahriyar, P. Chakraborty, and A. Iqbal, "Understanding the Motivations, Challenges and Needs of Blockchain Software Developers: A Survey," *Empir Softw Eng*, vol. 24, pp. 2636–2673, Aug. 2019, doi: 10.1007/s10664-019-09708-7.

[24] C. Casalnuovo, B. Vasilescu, P. Devanbu, and V. Filkov, "Developer On boarding in GitHub: The role of prior social links and language experience," in *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings*, Association for Computing Machinery, Inc, Aug. 2015, pp. 817–828. doi: 10.1145/2786805.2786854.

[25] E. Constantinou and T. Mens, "Socio-technical evolution of the Ruby ecosystem in GitHub," in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2017, pp. 34–44. doi: 10.1109/SANER.2017.7884607.

[26] O. Kononenko, O. Baysal, and M. W. Godfrey, "Code review quality: How developers see it," in *Proceedings - International Conference on Software Engineering*, IEEE Computer Society, May 2016, pp. 1028–1038. doi: 10.1145/2884781.2884840.

[27] M. Ortu, G. Destefanis, M. Kassab, S. Counsell, M. Marchesi, and R. Tonelli, "Would you mind fixing this issue?: An empirical analysis of politeness and attractiveness in software developed using agile boards," in *Lecture Notes in Business Information Processing*, Springer Verlag, 2015, pp. 129–140. doi: 10.1007/978-3-319-18612-2_11.

[28] M. Joblin, S. Apel, C. Hunsen, and W. Mauerer, "Classifying developers into core and peripheral: an empirical study on count and network metrics," in *Proceedings of the 39th International Conference on Software Engineering*, in ICSE '17. IEEE Press, 2017, pp. 164–174. doi: 10.1109/ICSE.2017.23.

[29] L. F. Dias, I. Steinmacher, and G. Pinto, "Who drives company-owned OSS projects: internal or external members?," *Journal of the Brazilian Computer Society*, vol. 24, no. 1, p. 16, 2018, doi: 10.1186/s13173-018-0079-x.

[30] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in GitHub," in *Proceedings - International Conference on Software Engineering*, IEEE Computer Society, May 2014, pp. 356–366. doi: 10.1145/2568225.2568315.

[31] H. Qiu, Y. Li, S. Padala, A. Sarma, and B. Vasilescu, "The Signals that Potential Contributors Look for When Choosing Open-source Projects," *Proc ACM Hum Comput Interact*, vol. 3, pp. 1–29, Nov. 2019, doi: 10.1145/3359224.

[32] D. Spadini, M. Aniche, and A. Bacchelli, "PyDriller: Python framework for mining software repositories," in *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Association for Computing Machinery, Inc, Oct. 2018, pp. 908–911. doi: 10.1145/3236024.3264598.

[33] Y. Hu, J. Zhang, X. Bai, S. Yu, and Z. Yang, "Influence analysis of Github repositories," *Springerplus*, vol. 5, no. 1, Dec. 2016, doi: 10.1186/s40064-016-2897-7.

[34] S. Surana, S. Detroja, and S. Tiwari, "A Tool to Extract Structured Data from GitHub," *ArXiv*, vol. abs/2012.03453, Dec. 2020, Accessed: Jun. 10, 2024. [Online]. Available: https://arxiv.org/abs/2012.03453

[35] O. Elazhary, M.-A. Storey, N. Ernst, and A.

Zaidman, "Do as I Do, Not as I Say: Do Contribution Guidelines Match the GitHub Contribution Process?," Aug. 2019, [Online]. Available: http://arxiv.org/abs/1908.02320

[36] K. Aggarwal, A. Hindle, and E. Stroulia, "Co-evolution of project documentation and popularity within Github," in *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, Association for Computing Machinery, May 2014, pp. 360–363. doi: 10.1145/2597073.2597120.

[37] S. Cao, Y. Zeng, S. Yang, and S. Cao, "Research on Python Data Visualization Technology," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Feb. 2021. doi: 10.1088/1742-6596/1757/1/012122.

[38] A. Lavanya *et al.*, "Assessing the Performance of Python Data Visualization Libraries: A Review," *International Journal of Computer Engineering in Research Trends*, vol. 10, no. 1, pp. 28–39, Jan. 2023, doi: 10.22362/ijcert/2023/v10/i01/v10i0104.

[39] S. Han and I.-Y. Kwak, "Mastering data visualization with Python: practical tips for researchers," *Journal of Minimally Invasive Surgery*, vol. 26, no. 4, pp. 167–175, Dec. 2023, doi: 10.7602/jmis.2023.26.4.167.

[40] Y. Zhang, M. Zhou, A. Mockus, and Z. Jin, "Companies' Participation in OSS Development-An Empirical Study of OpenStack," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2242–2259, Oct. 2021, doi: 10.1109/TSE.2019.2946156.

[41] V. N. Subramanian, "An empirical study of the first contributions of developers to open source projects on GitHub," in *Proceedings - 2020 ACM/IEEE 42nd International Conference on Software Engineering: Companion, ICSE-Companion 2020*, Institute of Electrical and Electronics Engineers Inc., Oct. 2020, pp. 116–118. doi: 10.1145/3377812.3382165.