Towards Understanding Open-Source Software Communities

Youness Hourri[1]

[1]Software Engineering Lab, University of Mons, Belgium

Abstract

Development bots are increasingly used in Open Source Software (OSS) projects on social coding platforms like GitHub to automate tasks, enforce coding standards, and streamline communication. Studies of specific bots highlighted their influence on the software development process, both positively, by automating repetitive tasks and improving efficiency, and negatively, by introducing challenges such as workflow disruptions or excessive notifications. This research proposal provides an overview of my research context, goals, and current research progress. My research focuses on the roles of contributors during collaborative OSS development and the effects of development bots on such collaboration. The primary goals are to (i) understand the evolving roles and interactions of contributors and (ii) assess how bots affect productivity, efficiency, and communication. By investigating GitHub activities such as pull requests, issue tracking, and code reviews, I will classify contributors and study their communication patterns. By comparing projects that utilize bots with those that do not, I will assess the impact of bots on key aspects of team dynamics, such as collaboration frequency, decision-making efficiency, and role distribution among contributors. The analysis will focus on measurable outcomes, including the speed of pull request reviews, the number of successfully merged code changes, and the rate of issue resolution. Additionally, I will examine how the presence of bots affects long-term contributor engagement and overall project velocity. This research will provide insights into human-bot interactions, develop guidelines for effective bot design, and suggest strategies to enhance contributor integration and communication, ultimately improving OSS project success.

Keywords

Open Source Software, Development Bots, Contributor Roles, Collaborative Software Development

1. Introduction

The evolution of version control systems has significantly transformed software development practices [1]. Among these systems, Git, introduced by Linus Torvalds in 2005, has emerged as a leading tool due to its distributed nature and powerful branching capabilities [2]. GitHub, launched in 2008, leveraged Git's robust version control capabilities and added a web-based interface, hosting services, and collaborative features, thereby fostering an environment for enhanced collaborative development [3]. These platforms have become integral to Open Source Software (OSS) development, enabling developers from around the world to contribute to projects, manage version control, and track changes effectively.

Social coding platforms like GitHub have revolutionized OSS collaboration by enhancing transparency and accessibility [4]. They facilitate not only code sharing and version control but also provide features such as pull requests, issue tracking, and project management tools. These features support asynchronous collaboration, allowing contributors to work independently while maintaining a cohesive project structure [5]. The inherent transparency in these platforms ensures that all contributions and discussions are publicly visible, promoting accountability and encouraging community engagement. However, it is important to note that not all contributions or discussions occur on GitHub itself, as contributors may also use other communication channels such as mailing lists, chat platforms, or forums.

Automation tools, such as GitHub Actions, have further optimized software development workflows by automating tasks like code building, testing, and deployment. Since its introduction in 2019, GitHub Actions has become a widely adopted CI/CD solution due to its ease of use and seamless integration with

BENEVOL24: The 23rd Belgium-Netherlands Software Evolution Workshop, November 21-22, Namur, Belgium

youness.hourri@umons.ac.be (Y. Hourri)

0009-0000-2068-5041 (Y. Hourri)

GitHub [6]. By supporting reusable actions, these tools help enhance both efficiency and consistency across projects.

Development bots, another form of automation, perform a wide range of functions that aid in the maintenance and enhancement of OSS projects [7]. These bots can manage pull requests, perform code reviews, update dependencies, and even triage issues. Popular examples include Dependabot, which automatically updates dependencies to secure versions [8], and Mergify, which automates the merging of pull requests based on predefined conditions. Bots often include conversational capabilities to interact with end users through textual messages (in chatbots) and speech (in voicebots), working as an interface between users and services [9]. By taking over routine and repetitive tasks, bots free up human contributors to focus on more complex and creative aspects of development.

However, the introduction of development bots also brings socio-technical challenges. While bots can significantly reduce the workload of human maintainers, they can also disrupt established collaboration patterns and workflows [6, 10]. Bots may generate excessive notifications, leading to alert fatigue among contributors, or they might perform actions that conflict with the expectations or intentions of human developers [11]. Recent studies have explored developer preferences for bot interactions, revealing that while bots generally increase productivity, their communication frequency and autonomy need careful balancing to align with developer expectations [12].

This research aims to understand the roles, interactions, and impact of development bots in OSS projects to improve community efficiency and collaboration. By examining GitHub activities such as pull requests, issues, and code reviews, I seek to understand how contributor roles evolve, how bots affect team dynamics, and what strategies can be employed to optimize human-bot collaboration.

2. Background

Understanding the roles and interactions of contributors in OSS projects has been a significant focus of research. Nakakoji et al. [13] proposed the “onion model”, categorizing community members into roles such as passive users, readers, bug reporters, bug fixers, peripheral developers, active developers, core members, and project leaders. This model highlights the layered structure of OSS communities, where contributors can move from peripheral to core roles based on their involvement. Xu et al. [14] simplified this model into four primary roles, emphasizing the blurred boundaries between these roles. Crowston and Howison [15] underscored the sporadic contributions of peripheral developers versus the regular contributions of core developers, essential for understanding OSS project sustainability and productivity.

Communication patterns within OSS projects are crucial for effective collaboration. Studies by Crowston et al. [16] and Joblin et al. [17] employed social network analysis to reveal how communication structures support or hinder collaboration. Core developers often serve as central nodes in these networks, facilitating coordination and decision-making processes. Cheng et al. [18] highlighted the multifaceted nature of contributor roles by differentiating between developmental and collaborative core developers.

OSS communities exhibit unique self-organization patterns that support project sustainability. Trinkenreich et al. [19] identified “project-centric roles” such as programming and system administration and “community-centric roles” focused on strategic management tasks. These "hidden figures" are crucial for community health but are often underrepresented in studies focusing on source-code contributions.

Development bots aim to automate repetitive tasks and improve workflow efficiency. However, bots can also introduce challenges. Ghorbani et al. [12] explored developer preferences for bot interactions, finding that developers prefer bots that are personable but show limited autonomy. Experienced developers appreciate more autonomous bots, indicating a need for configurable settings tailored to developer preferences. Brown and Parnin [10] and Paikari and van der Hoek [11] noted that bots could generate excessive notifications, leading to alert fatigue among contributors. These studies stress the importance of balancing bot automation with effective communication strategies.

Storey et al. [20] and Wang et al. [21] emphasized the diverse roles bots play in OSS projects, from specialized mechanics to project butlers. Their research underscores the importance of understanding how different bot configurations and roles impact team dynamics and project outcomes. Wessel et al. [7] characterized the functions and benefits of various bots in OSS projects, suggesting that well-configured bots can significantly enhance team productivity and project efficiency.

Identifying bot accounts is essential for analyzing their impact on OSS projects. Various methods have been proposed for detecting bots. Golzadeh et al. [22] created a ground-truth dataset and classification model that identifies bots through GitHub issue and pull request comments, using features such as the number of empty comments and comment patterns to achieve high precision. Abdellatif et al. [23] developed BotHunter, a machine learning-based approach that uses 19 features, including account profile information and activities, to identify bots with an F1-score of 92.4%. Another tool, RABBIT, introduced by Chidambaram et al. [24], uses recent GitHub event histories and an XGBoost classification model to detect bots with high accuracy while being scalable enough to process thousands of accounts per hour. These methods provide strong frameworks for distinguishing human contributors from bots, which is essential for accurately analyzing human-bot interactions in this research.

While prior research has examined the roles of contributors and the use of bots in OSS projects, significant gaps remain, particularly when shifting the focus from individual projects to entire ecosystems. Existing studies primarily explore the automation of repetitive tasks and efficiency gains within isolated projects, but little attention has been given to how bots influence long-term team dynamics, contributor role transitions, and communication patterns across larger ecosystems. Additionally, the socio-technical interactions between humans and bots, such as their impact on decision-making and collaboration, are underexplored, especially in complex, multi-project environments. My research aims to address these gaps by providing a comprehensive analysis of human-bot interactions across multiple projects within an ecosystem, focusing on collaboration, communication, and role evolution. This will help uncover new insights into how bots affect open-source communities at a broader level, offering strategies to optimize their integration and improve overall project success.

3. Research Goals

The objective of this research is to analyze and better understand the dynamics of collaboration within OSS ecosystems by examining the roles and interactions of both human and bot contributors, and assessing the impact of development bots on team dynamics, communication patterns, and project outcomes across multiple projects. Unlike previous studies that have primarily focused on individual OSS projects, this research expands its scope to explore the broader ecosystem, examining how contributors interact across multiple projects and how bots influence productivity, efficiency, and decision-making at scale.

Understanding these dynamics can provide insights into how well current bot deployment and management strategies support collaboration and productivity. This knowledge will be crucial in determining best practices for bot integration, identifying situations where bots can either enhance or disrupt workflows. By focusing on collaboration, role transitions, and decision-making processes, this research aims to offer empirical insights and actionable recommendations, not only for understanding how bots affect OSS ecosystems, but also for aligning their deployment with human contributors' needs, thus fostering more effective communication and project outcomes.

Goal 1: Understand Contributor Roles and Interactions in OSS Projects

This goal focuses on understanding how both human and bot contributors collaborate within and across OSS projects. By "roles", I refer to the various functions and responsibilities that contributors assume within a project, such as developers, maintainers, documentation writers, bug reporters, reviewers, peripheral developers, active developers, core members, and project leaders. In addition to classifying contributors into these roles, this research will examine community patterns to identify missing roles, shifts in responsibilities, and instances of role abandonment. I will study how these roles evolve over

time, examining communication patterns, types of contributions, and the progression from peripheral to core roles. The aim is to uncover insights into the socio-technical dynamics that support or hinder productivity and collaboration within OSS communities. To do so, I will study three research questions:

RQ1: **How do contributor roles vary across different OSS projects and change over time?**
Understanding how contributor roles vary across different OSS projects and change over time helps identify effective collaboration practices. This research will begin by identifying a comprehensive set of roles either through a literature survey or by analyzing contributor activities, while also considering how project characteristics (e.g., size, complexity, domain) influence the adoption of specific roles. By clustering contributors into roles, I will evaluate several algorithms to capture the nuances of role transitions and examine how contributors adopt multiple roles, either within a single project or across ecosystems. This analysis will help identify gaps in contributor roles and explore the prevalence of role overlap, uncovering patterns that may not be evident through manual classification. The goal is to identify role transitions, role gaps, and inform strategies for improving contributor retention, aligning roles with project needs, and facilitating smoother role transitions across OSS ecosystems.

RQ2: **What are the communication patterns and interaction dynamics among contributors in OSS projects?**
Identifying communication patterns can reveal how collaboration and coordination are maintained within OSS projects. These patterns provide insights into how information flows, decisions are made, and contributors interact. By analyzing communication logs and interaction data, I will build social graphs to visualize how contributors engage within projects. This analysis will help linking communication patterns to various contributor roles, highlighting how transitions between roles occur and which roles are more involved in communication.

RQ3: **How do OSS communities self-organize, and what are the key factors in the distribution of roles?**
Exploring the structure of OSS communities and the key factors in role distribution can significantly improve community health and contributor integration. I will use social network analysis to map and analyze relationships and influence patterns, which refer to how contributors impact decision-making and coordination within projects. These influence patterns will be quantified using metrics such as centrality, communication frequency, and involvement in key decisions. By distinguishing between human and bot roles, I aim to identify how bots influence these patterns and the roles taken by humans in their presence. This analysis will help uncover successful self-organization practices that can be applied to other OSS ecosystems.

The insights gained from Goal 1 lay the groundwork for Goal 2 by illustrating how development bots can assist and augment human workflows within OSS collaboration. Understanding the roles and community structures identified in Goal 1 enables informed decisions about when and where bots should intervene, particularly concerning contributor transitions from peripheral to core roles. Additionally, insights into communication patterns and self-organization within communities reveal opportunities for bots to fill collaboration gaps and enhance productivity. By pinpointing central contributor roles, such as reviewers and core developers, Goal 1 clarifies how bots can effectively support decision-making processes. Therefore, the analysis in Goal 2 will focus on the influence of bots throughout different stages of contributor evolution and team dynamics, leveraging the findings from Goal 1 to assess bot effectiveness across various contexts.

Goal 2: Explore the Role of Development Bots in OSS Collaboration

This goal aims to explore the role of development bots in influencing collaboration, productivity, efficiency, and decision-making within OSS projects and ecosystems. By evaluating how bots automate tasks, facilitate communication, and affect team dynamics, I will measure their influence on contributor

roles and workflows. The research will compare projects, communities, and ecosystems with and without bots to analyze differences in the roles and activities of human contributors. I will conduct both empirical and qualitative studies to collect data on bot usage, employing statistical techniques such as correlation analysis and regression modeling to identify patterns and relationships. These insights will help in developing strategies to optimize bot usage and enhance team performance across OSS ecosystems. Goal 2 will be broken down into the following three research questions:

RQ4: **What are the perceptions and attitudes of OSS contributors towards the use of bots in their projects?**
To understand OSS contributors' views on bot usage, surveys covering familiarity, benefits, drawbacks, and preferences will be distributed via mailing lists, forums, and social media. This feedback will be crucial for improving bot acceptance and effectiveness within OSS communities.

RQ5: **How do different types of bots influence the productivity and efficiency of OSS development teams?**
By examining key productivity metrics—such as code commit frequency, review turnaround time, issue resolution rates, and release cycles—this research will assess how different bot types affect team efficiency. Comparing OSS projects, ecosystems, and communities with similar characteristics but differing in bot usage will allow me to isolate the influence of bots on team performance.

RQ6: **How do bots influence decision-making processes and role transitions within OSS teams?**
This question will explore how the integration of bots affects decision-making dynamics and the transition of contributors between roles (e.g., from peripheral to core contributors). Using qualitative studies and empirical data, I will assess whether bots enable faster decision-making, reduce decision bottlenecks, or create new dynamics around role assignments.

4. Research Progress

For this research, a comprehensive dataset is essential to analyze the roles and interactions of contributors and the impact of development bots within OSS projects. The primary goal of the dataset is to answer the key question: *Who did what, and when?* To address this, the dataset must satisfy several critical requirements:

- **Historical Coverage**: It must span a significant period, ideally over a year, to allow for the observation of long-term trends in contributor behavior, role transitions, and the influence of development bots. This historical perspective is crucial for understanding how collaboration evolves over time.
- **Scale**: The dataset must capture contributions from a wide range of contributors across numerous repositories, representing diverse collaboration models. Activities such as opening pull requests, commenting on issues, pushing commits, and reviewing code should all be included to provide a comprehensive view of interactions and contributions across different projects.
- **Ecosystem-Level Scope**: It should encompass multiple projects within a larger ecosystem, with each project potentially involving several repositories. Contributors should actively collaborate across multiple projects, to enable the analysis of contributor and bot interactions both within and across projects. This will provide a complete picture of collaboration patterns, productivity, and the dynamics within the broader ecosystem.

By meeting these requirements, the dataset will be well-suited to capturing the complex dynamics of OSS collaboration and providing deep insights into how both human contributors and development bots influence the success and efficiency of these projects.

The NumFocus ecosystem[1] was selected for this research because it aligns well with the key requirements outlined above: historical coverage, scale, and ecosystem-level scope. With data spanning 18 months, the ecosystem provides solid mid-term historical coverage. It includes more than 2,000 repositories across 58 projects, with contributions from over 98,000 contributors, ensuring the necessary scale for analyzing large-scale collaboration. The interconnectedness of projects such as NumPy, Pandas, Matplotlib, and Jupyter enables contributors to collaborate across multiple repositories, satisfying the ecosystem-level scope requirement. Furthermore, many of these projects make use of development bots for tasks such as pull request management and automated testing, which provides valuable data on the role and impact of bots in OSS ecosystems.

4.1. Data Collection

For each GitHub repository within the NumFocus ecosystem projects, I retrieved a comprehensive list of GitHub events, which are specific actions recorded in a repository. GitHub tracks 17 distinct event types[2], such as PushEvent, PullRequestEvent, and IssueCommentEvent, each corresponding to different contributor actions. These events contain unique payloads that capture metadata like the actor involved, the repository affected, and the specific action performed.

While GitHub's REST API[3] provides access to these events, it has significant limitations: it allows retrieving only up to 300 events at a time, and the data spans just the last 90 days. This short-term access is insufficient for a historical and large-scale understanding of long-term trends and patterns in contributor behavior. To overcome this limitation, as shown in Figure 1, I relied on GH Archive[4], which records historical GitHub events dating back several years. GH Archive also includes data from inactive or deleted repositories, making it a comprehensive source for gathering the large-scale, long-term data necessary for this research.

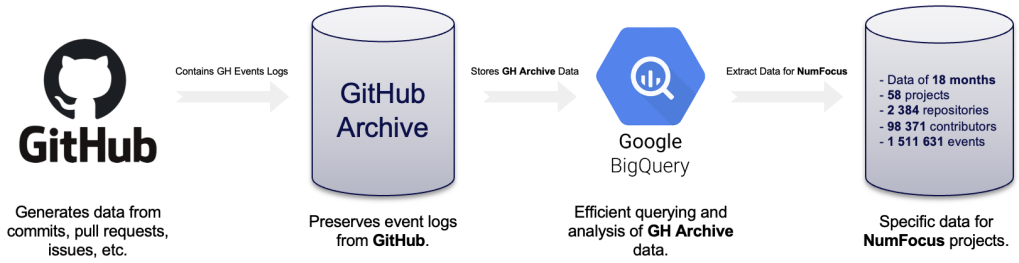

Figure 1: Data Collection Process

Using Google BigQuery, I efficiently queried and extracted the events, which currently spans from January 2023 to June 2024. It covers 58 NumFocus projects, more than 2,000 repositories, over 98,000 contributors, and more than 1.5 million events. These events capture a wide range of actions on pull requests, issues, and commits, providing a rich source of data for analyzing both human and bot contributions in OSS projects. To streamline the data extraction, I automated Python scripts interfaced with BigQuery, allowing for scalable and efficient data retrieval.

To maintain data quality, preprocessing steps were taken, including identifying and removing approximately 70 duplicate event records. Additionally, certain projects with little or no recent activity, such as The SHOGUN machine learning toolbox, were excluded from the dataset. Repositories that primarily serve as mirrors of projects hosted on other platforms, such as PETSc (a suite of data structures for scalable applications), were also removed to ensure that the analysis focuses on active and relevant

[1]NumFocus, https://numfocus.org/sponsored-projects
[2]GitHub event types, https://docs.github.com/en/rest/using-the-rest-api/github-event-types
[3]GitHub REST API documentation, https://docs.github.com/en/rest
[4]GH Archive, https://www.gharchive.org

repositories. These steps helped to ensure that the dataset accurately reflects ongoing collaboration and contribution patterns in the NumFocus ecosystem.

To better understand the structure and behavior of the dataset, I performed an exploratory data analysis, the results of which are visualized in the following figures. In this analysis, user types are categorized into human contributors and bots, with "bots" referring to automated accounts that handle routine tasks such as testing and pull request management. To accurately distinguish between these categories, I employed the RABBIT [24] bot detection tool to predict whether a GitHub account is human or bot based on activity features. RABBIT includes both bot user accounts and GitHub Apps[5], ensuring comprehensive categorization of automated contributions. This classification is crucial for analyzing how both human and bot contributors interact within the NumFocus ecosystem. The subsequent figures illustrate key trends in contributions, revealing the respective impacts of human and bot participation on collaboration and project outcomes, providing insights into their roles and interactions.

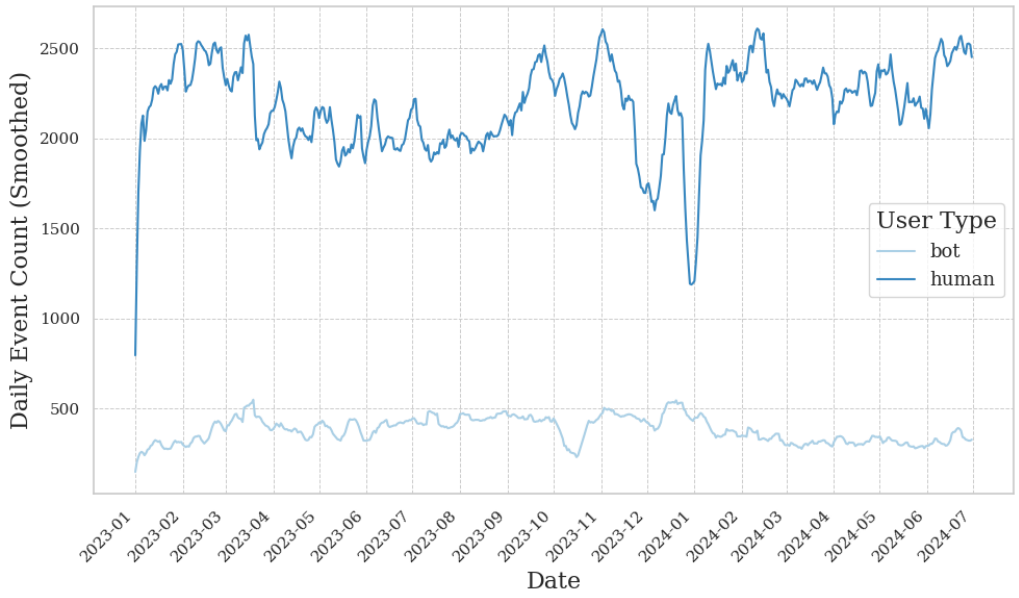

Figure 2: Contributions Over Time by User Type

Figure 2 illustrates the trends in total contributions over time by user type. The results show that human contributions are the most dominant, with peaks reaching approximately 2,500 daily events, while bot contributions, although less frequent, maintain a steady range between 200 and 500 events daily. This reflects the complementary role bots play, handling automation tasks that assist with repetitive workflows, without overwhelming the human contributions, which exhibit a cyclical pattern, likely influenced by project timelines or contributor availability.

Figure 3 presents the median number of event types per contributor, broken down by human and bot contributors. The plot reveals a more nuanced perspective on how bots and humans contribute to OSS projects. Bots are particularly active in tasks like pushing commits, where the median number of pushes per bot (72.5) significantly exceeds that of human contributors (26). Similarly, bots show higher involvement in creating and deleting resources, with median values of 17 Create and 15 Delete events, compared to 6 events for humans in both cases. This highlights the role of bots in automating repetitive and resource-management tasks.

Conversely, human contributors maintain a central role in review-based and collaborative tasks. For example, the median number of PullRequestReview and PullRequestReviewComment events is higher for humans (5 events) than for bots (3 events), indicating that humans are still more heavily involved in the review and feedback processes. For event types such as IssueComment and PullRequest, the median

[5]GitHub Apps documentation https://docs.github.com/en/apps

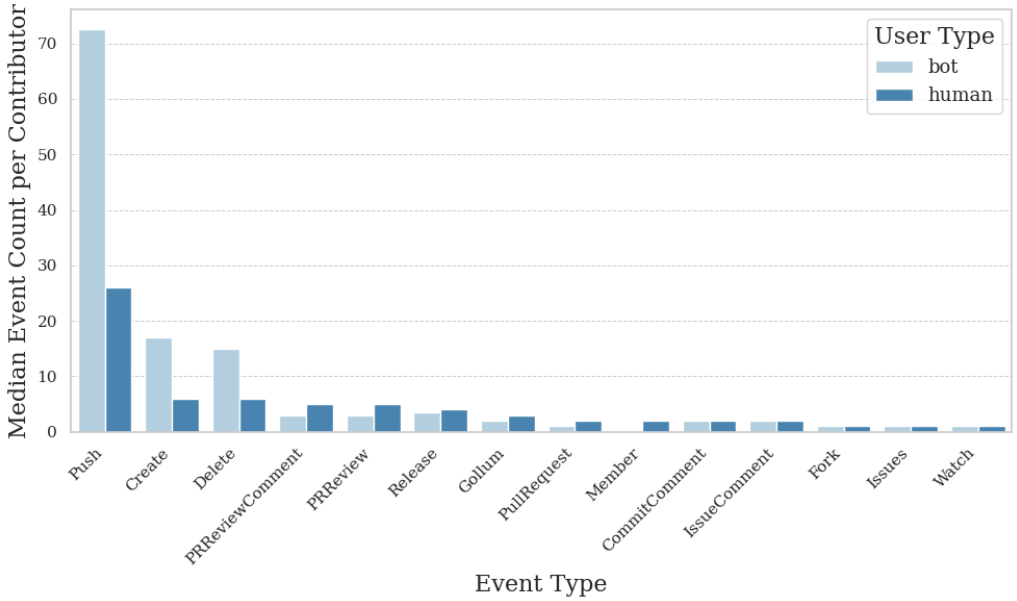

Figure 3: Median Distribution of GitHub Event Types by User Type

activity level between humans and bots is relatively similar (1–2 events), suggesting that both humans and bots contribute equally in these areas, albeit bots likely handle the more repetitive aspects.

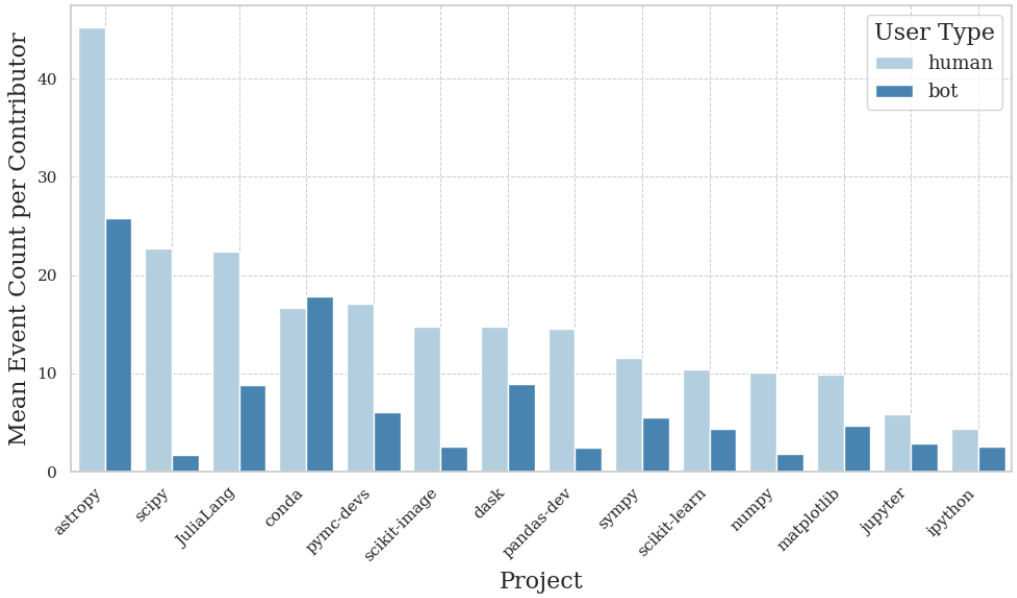

Figure 4: Mean Event Counts by Project and User Type

Finally, Figure 4 presents the average event count per contributor for human and bot users across various projects. In most projects, human contributors are generally more active, but there are exceptions where bots show significant contributions. For example, in Astropy, Scipy and Julia Programming Language, human contributors are significantly more active, with average event between 20 and 46. However, in Conda, bots are nearly as active as humans, with bots averaging 17.86 events per contributor compared to 16.67 for humans. This suggests a varying degree of automation across projects. Projects like Numpy and Pandas-dev show lower bot contributions relative to humans, indicating that bots are primarily used for routine tasks in these cases. Overall, the data reflects differing levels of human-bot collaboration and automation across the NumFocus ecosystem.

4.2. Data Mapping

A key objective of this research is to extract meaningful insights about contributor behavior and workflows in OSS projects by analyzing GitHub event data. GitHub events record observations of various actions within a repository, such as submitting pull requests or commenting on issues. However, these events do not always directly specify the exact actions performed. For instance, an IssueEvent is generated both when an issue is opened and when it is closed, but the event itself does not explicitly tell us which action occurred.

Previous work by Natarajan et al. [25] introduced a dataset that maps over one million low-level GitHub events into 24 high-level activity types. Their approach introduced a mapping that identifies activities using one or two GitHub events.

Our research builds on this by using a more comprehensive approach, where we map events to action types and group multiple related actions into a single activity type. Unlike Natarajan's mapping, which uses one or two events, our process can group up to four GitHub events to represent more complex activities performed by contributors at specific points in time. For example, in a typical pull request workflow, we can group a PullRequestEvent, PushEvent, IssueEvent, and DeleteEvent together, capturing the contributor's actions surrounding a pull request in one instance. This two-step mapping process allows us to first classify events and then analyze how multiple related actions combine into more meaningful workflows.

Step 1: Mapping Events to Action Types

GitHub events contain metadata, or a "payload," that provides additional context about the recorded action. By analyzing these payloads, we can determine the specific action that occurred. For example, an *IssueEvent* contains a field called *action* that specifies whether an issue was opened, closed, or reopened. Similarly, a *CreateEvent* contains details about whether a tag, branch, or repository was created. This additional information allows us to accurately map each event to a specific action type.

Each action is also accompanied by relevant metadata, such as the repository name, the actor (whether human or bot), and the date and time of the action. This provides the necessary context for analyzing contributor behavior at a more detailed level. For instance:

- A *CreateEvent* with *ref_type = tag* is mapped to *CreateTagAction*, representing the creation of a tag.
- A *ReleaseEvent* with *action = published* is mapped to *PublishReleaseAction*, reflecting the publishing of a release.
- An *IssueEvent* with *action = closed* is mapped to *CloseIssueAction*, indicating the closure of an issue.

By mapping raw events to action types, we simplify the complexity of GitHub event data and create a clearer framework for analyzing specific contributor actions.

Table 1 provides examples of how specific GitHub events are translated into high-level action types based on the metadata available in the event payload. This first step reduces the complexity of working directly with raw events by abstracting them into more meaningful actions.

Step 2: Mapping Action Types to Activity Types

While mapping events to actions provides granular insight into individual contributions, it does not capture the broader sequences of actions that contributors engage in over time. Many tasks on GitHub involve a series of related actions. For example, resolving an issue may involve closing the issue and adding a comment, while publishing a release may require creating a tag.

To understand contributor workflows at a higher level, we group related actions into broader activity types. An activity represents a series of connected actions that together reflect a contributor's full

Table 1
Event to Action Mapping for Complex Cases

Event Type	Action Type	Payload Details (Key Fields)
CreateEvent	*CreateTagAction*	*Tag creation in the repository (ref_type = tag)*
	CreateBranchAction	*Branch creation in the repository (ref_type = branch)*
	CreateRepositoryAction	*Repository creation for the project (ref_type = repository)*
PullRequestEvent	*OpenPullRequestAction*	*Pull request opened (action = opened)*
	ClosePullRequestAction	*Pull request closed (action = closed)*
	ReopenPullRequestAction	*Pull request reopened (action = reopened)*
PullRequestReviewEvent	*SubmitPullRequestReviewAction*	*Pull request review submitted (action = submitted)*
DeleteEvent	*DeleteBranchAction*	*Branch deleted in the repository (ref_type = branch)*
	DeleteTagAction	*Tag deleted in the repository (ref_type = tag)*

workflow within a repository. This second level of mapping enables us to see not just what actions were performed, but how those actions combine into meaningful sequences of activity.

For example:

- A combination of *CreateTagAction* and *PublishReleaseAction* is grouped into a *PublishReleaseActivity*, representing the complete process of publishing a release.
- A combination of *CloseIssueAction* and *CommentIssueAction* is grouped into a *CloseIssueActivity*, showing the workflow involved in closing an issue and leaving a comment.

This two-step mapping process, from events to actions and then from actions to activities, allows us to capture the complexity of OSS contributor workflows more effectively. It provides a comprehensive view of how both human contributors and bots interact across multiple repositories and projects, offering deeper insights into collaboration dynamics within the NumFocus ecosystem.

As shown in Table 2, This step involves grouping related actions into broader activity types. These activity types represent complete workflows or sequences of actions that contributors engage in within a repository.

Table 2
Action to Activity Mapping for Complex Cases

Activity Type	Action Type(s)	Description
PublishReleaseActivity	*PublishReleaseAction, CreateTagAction*	Publishing a release that includes tagging the release for versioning.
CloseIssueActivity	*CloseIssueAction, CommentIssueAction*	Closing an issue and adding a comment to provide further explanation.
ReopenIssueActivity	*ReopenIssueAction, CommentIssueAction*	Reopening a previously closed issue, with additional comments.
ApprovePullRequestActivity	*ClosePullRequestAction, PushCommitsAction, DeleteBranchAction*	Approving and merging a pull request, with related actions.
SubmitPullRequestReviewActivity	*SubmitPullRequestReviewAction, CommentPullRequestReviewAction*	Submitting a pull request review after adding comments.

This two-step mapping process is essential for transforming raw GitHub event data into structured, higher-level activities, which are necessary for the subsequent analysis of contributor roles, collaboration patterns, and bot interactions. By grouping related actions, the mapping provides a clear framework for analyzing complex contributor behaviors and how bots influence productivity. This structured dataset will be used in the next phase of our research to conduct quantitative analyses, where we will evaluate the impact of bots on collaboration efficiency and examine the dynamics of contributor interactions within and across projects in OSS ecosystems.

4.3. Future Work

This research will progress through several stages, starting with presenting a comprehensive overview of the dataset collected from the NumFocus ecosystem. This dataset, covering multiple repositories, will provide detailed insights through event-action and action-activity mappings. These mappings offer a structured understanding of contributor workflows and will enable deeper analysis of how human and bot contributors collaborate over time.

The next phase will focus on analyzing the evolution of contributor roles within open-source projects, particularly the transitions from peripheral to central roles. Clustering algorithms will be applied to identify patterns in role transitions and role overlap, and to explore how these factors affect collaboration and productivity. This analysis will help develop strategies for improving contributor retention and aligning roles with project needs.

Further work will investigate communication patterns and interaction dynamics between contributors. Using social network analysis, I will examine key metrics such as centrality and communication frequency to reveal how information flows within OSS communities and how contributors interact across projects. This analysis will shed light on key contributors, decision-making processes, and the influence of bots on communication patterns and role distribution. If necessary, qualitative studies may also be conducted to gain deeper insights into individual and team dynamics.

Finally, I will examine the socio-technical dynamics that govern the self-organization of OSS communities. Building on the findings from role and communication analysis, this stage will focus on understanding how human contributors and bots influence the distribution of roles and responsibilities. The goal is to provide actionable recommendations for improving collaboration, contributor integration, and project success in OSS ecosystems.

5. Conclusion

This research focuses on data collection and mapping as foundational steps to understanding human and bot interactions in OSS projects. By gathering and organizing a large-scale dataset from the NumFocus ecosystem, I have created a framework that enables the structured study of contributor workflows through event-action-activity mappings.

The primary achievement of this work is the successful collection and organization of this dataset, which provides a solid foundation for future analysis. These early findings lay the groundwork for further exploration into how bots influence OSS collaboration, with the next stages focusing on role transitions, communication patterns, and socio-technical dynamics.

Acknowledgments

This research is supported by the F.R.S.-FNRS research project F.4515.23.

References

[1] M. W. Godfrey, D. M. German, The past, present, and future of software evolution, in: 2008 Frontiers of Software Maintenance, IEEE, 2008, pp. 129–138.

[2] J. Loeliger, M. McCullough, Version Control with Git: Powerful tools and techniques for collaborative software development, " O'Reilly Media, Inc.", 2012.

[3] S. Chacon, B. Straub, Pro git, Springer Nature, 2014.

[4] L. Dabbish, C. Stuart, J. Tsay, J. Herbsleb, Social coding in github: transparency and collaboration in an open software repository, in: Proceedings of the ACM 2012 conference on computer supported cooperative work, 2012, pp. 1277–1286.

[5] J. Tsay, L. Dabbish, J. Herbsleb, Let's talk about it: evaluating contributions through discussion in github, in: Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering, 2014, pp. 144–154.

[6] M. Wessel, T. Mens, A. Decan, P. R. Mazrae, The github development workflow automation ecosystems, in: Software Ecosystems: Tooling and Analytics, Springer, 2023, pp. 183–214.

[7] M. Wessel, B. M. De Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, M. A. Gerosa, The power of bots: Characterizing and understanding bots in oss projects, Proceedings of the ACM on Human-Computer Interaction 2 (2018) 1–19.

[8] S. Mirhosseini, C. Parnin, Can automated pull requests encourage software developers to upgrade out-of-date dependencies?, in: 2017 32nd IEEE/ACM international conference on automated software engineering (ASE), IEEE, 2017, pp. 84–94.
[9] E. Shihab, S. Wagner, M. A. Gerosa, M. Wessel, J. Cabot, The present and future of bots in software engineering, IEEE Software 39 (2022) 28–31.
[10] C. Brown, C. Parnin, Sorry to bother you: Designing bots for effective recommendations, in: 2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE), IEEE, 2019, pp. 54–58.
[11] E. Paikari, A. Van Der Hoek, A framework for understanding chatbots and their future, in: Proceedings of the 11th international workshop on cooperative and human aspects of software engineering, 2018, pp. 13–16.
[12] A. Ghorbani, N. Cassee, D. Robinson, A. Alami, N. A. Ernst, A. Serebrenik, A. Wąsowski, Autonomy is an acquired taste: Exploring developer preferences for github bots, in: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), IEEE, 2023, pp. 1405–1417.
[13] K. Nakakoji, Y. Yamamoto, Y. Nishinaka, K. Kishida, Y. Ye, Evolution patterns of open-source software systems and communities, in: Proceedings of the international workshop on Principles of software evolution, 2002, pp. 76–85.
[14] J. Xu, Y. Gao, S. Christley, G. Madey, A topological analysis of the open souce software development community, in: Proceedings of the 38th Annual Hawaii International Conference on System Sciences, IEEE, 2005, pp. 198a–198a.
[15] K. Crowston, J. Howison, The social structure of free and open source software development (2005).
[16] K. Crowston, K. Wei, Q. Li, J. Howison, Core and periphery in free/libre and open source software team communications, in: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), volume 6, IEEE, 2006, pp. 118a–118a.
[17] M. Joblin, S. Apel, C. Hunsen, W. Mauerer, Classifying developers into core and peripheral: An empirical study on count and network metrics, in: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), IEEE, 2017, pp. 164–174.
[18] L.-T. Cheng, S. Hupfer, S. Ross, J. Patterson, Jazzing up eclipse with collaborative tools, in: Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange, 2003, pp. 45–49.
[19] B. Trinkenreich, M. Guizani, I. Wiese, A. Sarma, I. Steinmacher, Hidden figures: Roles and pathways of successful oss contributors, Proceedings of the ACM on human-computer interaction 4 (2020) 1–22.
[20] M.-A. Storey, A. Serebrenik, C. P. Rosé, T. Zimmermann, J. D. Herbsleb, Botse: Bots in software engineering (dagstuhl seminar 19471), in: Dagstuhl Reports, volume 9, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
[21] Z. Wang, Y. Wang, D. Redmiles, From specialized mechanics to project butlers: The usage of bots in open source software development, IEEE Software 39 (2022) 38–43.
[22] M. Golzadeh, A. Decan, D. Legay, T. Mens, A ground-truth dataset and classification model for detecting bots in github issue and pr comments, Journal of Systems and Software 175 (2021) 110911.
[23] A. Abdellatif, M. Wessel, I. Steinmacher, M. A. Gerosa, E. Shihab, Bothunter: An approach to detect software bots in github, in: Proceedings of the 19th International Conference on Mining Software Repositories, 2022, pp. 6–17.
[24] N. Chidambaram, T. Mens, A. Decan, Rabbit: A tool for identifying bot accounts based on their recent github event history, in: 2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR), IEEE, 2024, pp. 687–691.
[25] N. Chidambaram, A. Decan, T. Mens, A dataset of bot and human activities in github, in: 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), IEEE, 2023, pp. 465–469.