RESEARCH-ARTICLE

Are External Contributions Important to Project Productivity in Open Source Software? A Deep Insight based on Issue Entropy

YANG SHEN, National University of Defense Technology China, Changsha, Hunan, China

TAO WANG, National University of Defense Technology China, Changsha, Hunan, China

XUNHUI ZHANG, National University of Defense Technology China, Changsha, Hunan, China

YANG ZHANG, National University of Defense Technology China, Changsha, Hunan, China

CHENG YANG, National University of Defense Technology China, Changsha, Hunan, China

YUE YU, Peng Cheng Laboratory, Shenzhen, Guangdong, China

View all

Proceedings of the ACM on Human-Computer Interaction, Volume 9, Issue 7 (October 2025)
https://doi.org/10.1145/3757399
EISSN: 2573-0142

Are External Contributions Important to Project Productivity in Open Source Software? A Deep Insight Based on Issue Entropy

YANG SHEN, National University of Defense Technology(NUDT), China and State Key Laboratory of Complex & Critical Software Environment(CCSL), China
TAO WANG**, NUDT, China and CCSL, China
XUNHUI ZHANG, NUDT, China and CCSL, China
YANG ZHANG, NUDT, China and CCSL, China
CHENG YANG, NUDT, China and CCSL, China
YUE YU, Peng Cheng Laboratory, China
HUAIMIN WANG, NUDT, China and CCSL, China

In the realm of open source software (OSS) development, the resolution of issues is not just a technical task but a pivotal activity that drives ongoing enhancement and secures a project's sustainability. Contributing to issues is the majority form for external contributors to take part in OSS projects. Although the significance of external contributors is recognized, their contributions in issue process still lacks full quantification and clarity, and the correlation between their contributions and the project productivity remains unclear. In response, we propose issue entropy, a novel metric that quantifies the complexity of event sequence in issue process applied to study the external contributions. The metric applies principles of information theory to scrutinize granular details within a project's issue-related activities, providing a unique lens to understand and assess external contributions. To explore the correlation between external contributions and project productivity, we employed issue entropy as a novel way to examine external contributions, and analyzed its correlation with new bugs, commits, and bug fix time, serving as proxies for OSS project productivity. Our findings reveal a strong positive correlation with new bugs, variable relationship with commit volume based on company sponsorship, and significant negative correlation with bug fix time. We also observe the significant interactions between external contributions and other factors, such as project age and the number of files. Moreover, issue entropy offers a new perspective on the health of the OSS project ecosystem, potentially supporting further research and practical applications.

CCS Concepts: • **Software and its engineering → Software development process management**; • **Human-centered computing → Collaborative and social computing**.

Additional Key Words and Phrases: Open Source Software, Issue Tracker Complexity, Developers' Activity, Issue Entropy, Productivity

ACM Reference Format:
Yang Shen, Tao Wang, Xunhui Zhang, Yang Zhang, Cheng Yang, Yue Yu, and Huaimin Wang. 2025. Are External Contributions Important to Project Productivity in Open Source Software? A Deep Insight Based on Issue Entropy. *Proc. ACM Hum.-Comput. Interact.* 9, 7, Article CSCW218 (November 2025), 26 pages. https://doi.org/10.1145/3757399

*Corresponding Author(taowang2005@nudt.edu.cn)

ACM 2573-0142/2025/11-ARTCSCW218
https://doi.org/10.1145/3757399

1 Introduction

In the dynamic ecosystem of open-source software (OSS) development, the process of issue resolution extends beyond a mere technical necessity; it is also an important mechanism for the communication and coordination among stakeholder, and then affect the evolution of open source projects [9]. Traditionally, the issue tracker has been a pivotal tool for software teams, utilized for reporting, discussing, and keeping track of defects and feature requests [8]. It serves as both the project's memory and its ongoing to-do list, capturing the collective efforts and conversations of its contributors.

The process of issue resolution relies on the collaborative efforts of a community that extends beyond just core developers. A broader range of contributors, commonly referred to as external contributors, is a vital force within the OSS realm [4, 30]. These individuals may not directly shape the project's strategic direction but make invaluable contributions through pull requests and issue tracker [39], encompassing bug reports, feature requests, fixes, and providing insightful feedback crucial to ongoing project development [23, 59]. As suggested by [43], issue trackers predominantly host the activities of external contributors. This empirical evidence not only highlights the critical role of external contributors in collaborative workflows but also validates the significance of our approach to identifying and studying external contributions in issue trackers. Moreover, a robust influx of participation from external contributors is widely recognized as a key indicator of successful open-source projects [23, 39, 57].

Although the significance of external contributors is recognized, their contributions in issue process still lack full quantification and clarity, and the relationship between these contributions and project productivity remains unclear. Given that up to 50% estimated effort in development is devoted to issue resolution [11], comprehending these dynamics becomes not only beneficial but also necessary for efficient project management.

The significance of external contributions cannot be fully understood without examining the dynamics of the issue resolution process itself. Specifically, the issue resolution process involves a series of fine-grained events, such as creating, commenting, assigning, and closing, which form an event sequence. This sequence not only represents the technical steps taken but also reflects the collaborative efforts, feedback loops, and decision-making processes involved. The complexity of these sequences indicates the nature and depth of contributions, with more intricate and varied sequences suggesting a greater contribution to the project through issue resolution. As Anne et al.[7] suggest, a set of diverse and sufficiently varied team and process approaches is essential to addressing the challenges in open-source software development.

Recognizing this, we propose using the complexity of these event sequences as a novel way to examine external contributions. Specifically, we employ information theory and design a metric named issue entropy to measure the complexity based on the sequence of fine-grained events within the project's issues, providing a novel lens to view and assess external contributions. By analyzing complexity of event sequences within the issue tracker, we can capture the nuanced patterns of contributions that simple metrics (e.g., the number of events) might miss. Issue entropy reflects not only the volume of interactions but also the coordination challenges and the variety of contributions involved—elements essential for understanding the impact of external contributors.

Furthermore, To explore the relationship between external contributions and project productivity, we employ issue entropy to study the complexity of external contributions. In line with Forsgren's recommendations [25], we adopt three key indicators as proxies for **project productivity, corresponding to Performance, Activity, and Efficiency & Flow dimensions: the number of new bugs, the number of new commits, and the bug fix time**. The reasons for selection are explained in 4.1. Our empirical analysis is conducted using detailed event data from eight prominent

OSS projects on GitHub, which include both corporate-driven and community-driven projects. This analysis is guided by the following research questions.

- **RQ1:** *What is the correlation between external contributions and the number of new bugs in OSS projects, and what factors could alter this dynamic?*

The number of new bugs is commonly used as a proxy for evaluating software quality in projects [52, 64, 66]. We use this metric to capture project productivity in the Performance dimension, assessing the outcome of the software development process [25]. The result suggests that a project's external contributions is significantly positively correlated with the number of new bugs. This correlation tends to decrease as projects mature and increase with the number of forks, particularly for non-company projects.

- **RQ2:** *How are external contributions correlated with the number of new commits in OSS projects, and what factors may affect this connection?*

In many studies focusing on OSS development and community, the number of commits is considered as the productivity metric [52, 62, 63]. We adopt this widely used productivity indicator to capture project productivity in the Activity dimension, assessing the output of the software development process [25]. The result suggests that external contributions has a distinct correlation with the number of new commits: it's inversely related in non-company projects but shows a positive correlation in company projects. We also observe significant interactions between external contributions and other factors. For example, the negative effect of external contributions on new commits grows as the number of forks and files rises in non-company project.

- **RQ3:** *To what extent do external contributions correlate with bug fix time in OSS projects, and what factors could strengthen or weaken this association?*

The bug fix time is also regarded as an important metric for productivity [26, 47]. We adopt this metric to capture project productivity in the Efficiency & Flow dimension, assessing the efficiency of the software development process [25]. Our regression analysis reveals that external contributions is significantly negatively correlated with bug fix time. Similarly, we have observed intriguing interactions between external contributions and other factors. Notably, the negative effect of external contributions on bug fix time decreases with the increase in the number of forks for non-company projects, but not vice versa, whereas it intensifies within company projects.

This article aims to systematically study external contributions through event sequences analysis and examine their relationship with three dimensions of OSS project productivity. The overall framework of this paper is shown in Figure 1. Through the above studies, we conclude that external contributions play an important role in understanding the development of OSS projects. And the proposed metric is an effective and useful indicator in assessing complexity of event sequences in issue processes, which shows significant correlations to project productivity indicators about OSS development and maintenance.

In summary, this work makes the following contributions.

- We propose issue entropy to measure complexity of event sequence in issue process, providing a novel lens to view and assess external contributions.
- We identify external contributions and employ issue entropy to study their complexity, and identify its relationship with new bugs, commits, and bug fix time, serving as proxies for OSS project productivity. Our findings highlight the utility of issue entropy in elucidating project productivity.
- We discuss the implications of issue entropy in understanding and guiding practice of issue resolution for OSS development.

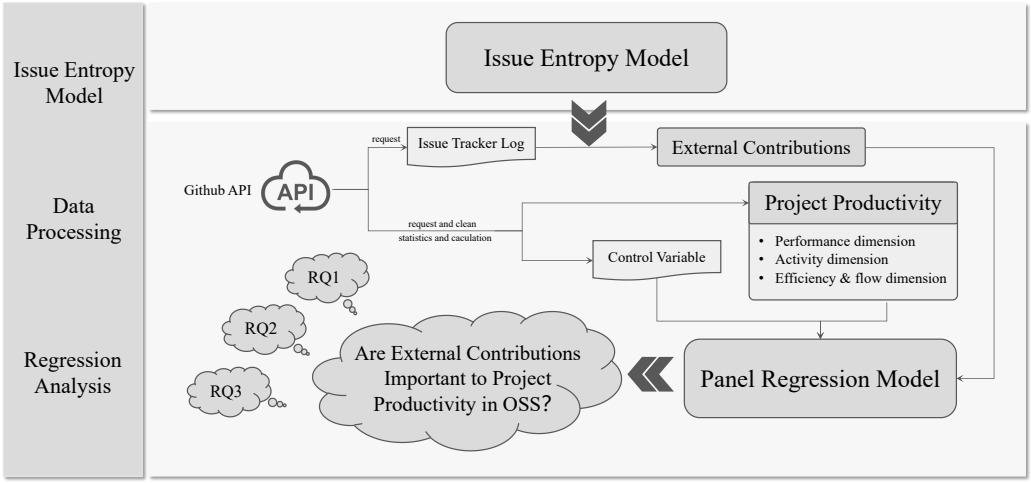

Fig. 1. Research framework of this paper

The rest of the paper is organized as follows. Section 2 presents the related work. In Section 3, we present the process of calculating the proposed issue entropy and discuss its properties. Section 4 introduces the design of the study. We report the results and findings in Section 5. In Section 6, we discuss the implications and the threats to validity of this work. Finally, Section 7 concludes the paper and outlines potential future work.

2 Related Work

2.1 Issue tracker and External Contributor

Issue tracker is an essential component of open source software development, serving as a key tool for social interaction and collaboration [18, 71], significantly impacts project productivity [17, 19, 56].

The vitality of open source software is closely linked to external contributions, a dynamic that has been extensively studied within the realm of issue tracker. Padhye et al. [49] distinguish between core, external, and mutant code committers, revealing that for projects in popular scripting languages, the presence of external committers rivals that of core committers. The types of contributions that are typically most significant, as argued by Daniel et al. [23], are task requests and forum discussion posts, which are critical forms of external contributions. As suggested by [43], issue trackers predominantly host the activities of external contributors. These findings collectively establish that external contributions within issue trackers are high and pivotal to the advancement of open source software.

The analysis of activities such as bug resolution and feature request implementations is endorsed by Crowston et al. [22], who assert that these activities yield process data indicative of a project's health and progress. They also emphasize the importance of quantifying the number and engagement level of volunteer developers as key indicators of OSS success.

Several studies have aimed to quantify individual developer performance in open-source software projects. Gousios et al.[28] measured individual contributions by calculating a weighted sum of events, while Robles et al.[55] converted developer activities into effort measured in person-months, using full-time developer effort as a benchmark. Zhao et al.[70] constructed a social network based on developer activity data within issue trackers, using the centrality of developer nodes to measure

individual contributions. These studies focus on quantifying individual developer contributions and do not thoroughly analyze the relationship between these contributions and the project's overall development. Furthermore, while some work has discussed the importance of external contributions to project development, they primarily focus on metrics such as the number of developers[57], the number of commits[40], without delving into the details of contributor activities.

In our work, we propose issue entropy, a novel metric derived from event logs, to measure the complexity of event sequence in issue process, allowing for a more comprehensive modeling of external contributions. Additionally, we explore the relationship between external contributions and project productivity in greater depth and provide specific recommendations for project management.

2.2 Event Log Analysis in OSS

Given the abundance of event logs within large repositories, several studies have applied sequence analysis to investigate these logs. Howison et al. [33] reconstructed work episodes from archival records and, through qualitative analysis, identified potential patterns within event logs. Brian et al. [36] developed sequence analysis methods suited to large-scale distributed collaborations, recommending the use of techniques such as pattern mining, sequence similarity, and probabilistic analysis. Christian et al. [44] focused on event transition probabilities and proposed heatmapping as a way to promote pattern discovery.

Although sequence analysis has been widely adopted in OSS research, its primary focus has been on extracting temporal relationships among events. In contrast, our work emphasizes the combinations of event types and how these combinations evolve over time, offering new insights into external contributions.

Some studies have also explored project attributes by defining sequence-based metrics. Lindberg et al. [42] examined unresolved interdependencies within OSS by defining metrics such as degree of development interdependencies, activity variation, and order variation. Hassan et al. [32] analyzed code defects, using the complexity of code change sequences to improve defect prediction models. Wang et al. [65] explored fork diversity by defining commit sequence metrics and studying how fork diversity influences project development.

Consistent with these studies, we also use entropy to capture sequence-based diversity and complexity. Uniquely, our focus is on event sequence in issue process. Additionally, our metric employs a variant of Shannon entropy, extending the concept from a static snapshot to a dynamic model that emits data at a variable rate. Furthermore, we adopt a joint entropy approach, considering both event distribution and user distribution.

2.3 Information Theory

Information theory is fundamentally concerned with the quantification and characterization of information contained within a message [58]. Central to this field is the measurement of uncertainty, which is intrinsically linked to the concept of information. Take, for example, the output from a device emitting one of four possible symbols: A, B, C, or D. Prior to the emission, uncertainty exists about which symbol will be produced. That is, the output distribution is unknown. The emergence of a symbol reduces this uncertainty, and consequently, we gain information about the system's behavior.

Shannon introduced a method to quantify this uncertainty or entropy in a distribution. The Shannon entropy, denoted as H_n, is defined as:

$$H_n(P) = -\sum_{i=1}^{n}(p_i \log_2 p_i)$$

where $p_i \geq 0, \forall i \in 1, 2, ..., n$ and $\sum_{i=1}^{n} p_i = 1$. By defining the amount of uncertainty in a distribution, H_n captures the complexity within an output distribution and is extensively utilized in software engineering research, as a prevalent metric for the complexity of system [3, 32, 38].

Transitioning to a dynamic framework, when a device emits data at a specific temporal rate, a variant form of Shannon entropy per unit time comes into play:

$$H = f \cdot H_n$$

Where f is the frequency of data emitting. This adaptation propels the concept of entropy from a static snapshot (the entropy of a single output distribution) to a dynamic model that incorporates the temporal element, reflecting the variability of information over time. This dynamic perspective enables a more holistic assessment and comparison of the information output by various systems or processes, offering a richer, time-sensitive analysis of informational dynamics.

3 Issue Entropy

In this section, we introduce issue entropy as a metric to quantify the complexity of event sequence in issue process. We present calculation detail of issue entropy, and discuss its behavior in various scenarios. Additionally, we give its application practice in external contributions.

3.1 Definition and Calculation of Issue Entropy

We consider event generation process of an issue within a period of time as a device which emits data continuously at a certain rate(that is, the single-symbol discrete memoryless information source in information theory[58]), and the data of a single emission is defined as an event executed by a user within the issue. Then we can apply the ideas of information theory to quantify the complexity of event sequence in issue process.

A real issue case from the rails/rails project illustrates the calculation of issue entropy. As shown in Figure 2 , events are classified based on the joint attribute of user and event type. Each event is represented as a pair (u_i, e_j), where u_i denotes the user, and e_j indicates the type of event they perform. This joint classification enables us to analyze both the nature of the events and the specific user involved. To analyze the issue, a time division is defined, such as day-wise or month-wise, within which the joint probability distribution P of the issue's events is established. This distribution provides the posterior probability of different events occurring for various users over a specified time period. Let $P(u_i, e_j)$ represent the probability that user u_i performs an event of type e_j. In the shaded area of Figure 2, representing the target time window, five events occur, with the event (1c7, commented) appearing twice. Thus, $P(1c7, \text{commented}) = 2/5 = 0.4$. Similarly, $P(skipkayhil, \text{labeled}) = 2/5 = 0.4$, and $P(skipkayhil, \text{commented}) = 1/5 = 0.2$. These probabilities reflect the distribution of actions across different users, offering insights into how users engage with the issue process over time.

Within each time window, the frequency of events varies; hence, we can define the entropy of an issue for a given period as follows:

$$H_{issue} = \frac{N}{T} \sum_i \sum_j -P(u_i, e_j) \log_2 P(u_i, e_j) \tag{1}$$

$$H_{project} = \sum H_{issue} \tag{2}$$

Where N denote the total of events within the time window, T denote the length of time window, and $\frac{N}{T}$ is the frequency of events.

In Figure 2, the details column outlines each event within the issue, which involves processes such as creation, discussion, categorization, declaration, PR submission for resolution, and closure.

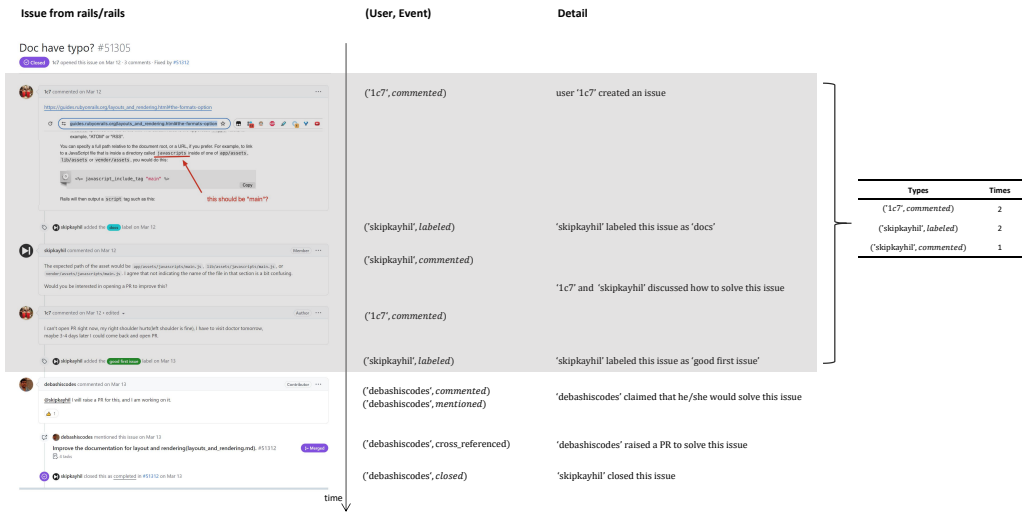

Fig. 2. Calculation of $P(u_i, e_j)$

This range of activities reflects a thorough resolution process, conducted by multiple contributors and highlighting a high level of team collaboration. Issue entropy serves as a quantitative measure of the complexity of the event sequence, offering deeper insights into the collaborative dynamics and engagement patterns within an OSS project. By analyzing issue entropy, we gain insights into how interactions and contributions are distributed across event types and among different users, revealing patterns in team collaboration and the overall depth of contribution to issue resolution.

Issue entropy is particularly informative when considering both ends of the entropy spectrum. When entropy is minimal, it implies a highly repetitive process dominated by a single user repeatedly engaging in the same type of action. For example, a user solely performing "comment" actions with minimal variation suggests that the issue is either narrowly focused on specific feedback or lacks diverse contributions from multiple users. This low entropy may indicate a less collaborative environment or a straightforward resolution process with limited engagement, or it may suggest that the issue has not been fully resolved. Minimal entropy signals to project managers that there may be a need for more varied engagement to enhance collaboration and improve the issue resolution process.

Conversely, when entropy is high, it reflects a scenario where multiple event types are performed by different users, indicating a broader scope of engagement. In this case, diverse actions, such as "issue assignment", "labeling", "commenting", and "closing", are performed by a variety of contributors, each bringing unique perspectives to the issue's resolution. High entropy suggests a complex, multifaceted problem-solving process where various stakeholders participate, contributing to a richer, more collaborative environment. This increased diversity of actions and user involvement often correlates with a deeper exploration of the issue, as different contributors bring in their expertise and insights, enhancing the quality of resolution. Such an issue resolution process provides greater value to the project, as it benefits from a comprehensive, collaborative approach[7].

To illustrate the distinction between issue entropy and basic metrics such as contributor count, consider that a high number of contributors does not necessarily equate to a high level of contribution. For instance, a project may have many contributors but exhibit low issue entropy if those contributors predominantly perform similar actions, such as commenting on issues without

engaging in diverse activities like labeling or assigning tasks. This scenario highlights the importance of not only the quantity of contributors but also the quality and variety of their interactions, which issue entropy captures more effectively. In contrast, relying solely on contributor count may provide a misleading impression of project health. By analyzing issue entropy alongside such simple metrics, project managers can gain a more nuanced understanding of team dynamics and engagement levels.

3.2 Application Practice in External Contributions

Although issue entropy has the ability to capture overall contributions in issue process, the focus of this study is specifically on external contributions. The participation of external contributors tends to be sporadic, adding a layer of unpredictability and complexity to issue trackers that is not typically observed with the more consistent and structured involvement of core team members [6]. As noted by [43], issue trackers often predominantly feature activities from external contributors. This focus allows a closer examination of the unique dynamics introduced by these external contributors. Therefore, in the empirical analysis, issue entropy is calculated based solely on the activities of external contributors.

OSS development teams consist of a small, well-organized core team and a larger, loosely-coupled group of external contributors [24, 69], with the issue tracker capturing the activities of both groups. Previous studies used GitHub's repository permission mechanism to distinguish core team members from external contributors[31]. However, GitHub does not allow anyone except the repository owners to access the list of members with management privileges. Wang et al. [66] proposed a more reliable method by identifying core team members through their performance of events that require write permission. However, this approach may miss some core team members whose roles do not require write permissions. For example, the role of "Triage" on GitHub allows core team members to manage issues and pull requests without write permissions.

Building on Wang et al.'s method, this study uses a permissions-check mechanism to distinguish core team members from external contributors, but it considers all privileged events, not just those requiring write permissions.

Specifically, based on GitHub's permissions documentation [27], permissions for 53 types of issue-level events were manually annotated to determine which actions require management privileges. It was found that 16 of these events could be performed by external contributors without needing management privileges. Users who have performed events requiring management privileges are labeled as "core team member", while all others are classified as "external contributor". Notably, the required permissions for some actions depend on the target object. For instance, users can close issues they created themselves but need management privileges to close issues created by others, even though both scenarios are categorized as "Closed Event" on GitHub. See the replication package for details[1].

4 STUDY DESIGN

In this section, we firstly present the selection of key productivity metrics based on the SPACE framework, and relevant control variables. We then outline the data collection and cleaning process from eight open source projects on GitHub. Finally, we explain the use of panel regression analysis to evaluate the relationship between issue entropy and project productivity.

[1] https://github.com/anonymous2024CSCW/2024CSCW-replication-package

4.1 Variables

Forsgren et al. [25] introduced the SPACE framework, which presents a comprehensive view of productivity by assessing five dimensions: Satisfaction & well-being, Performance, Activity, Communication & collaboration, and Efficiency & flow. To capture the project productivity, we have selected three metrics closely tied to the issue tracker according to the SPACE framework guidelines, spanning three dimensions: Performance, Activity, and Efficiency & Flow. These metrics are new bugs, new commits, and bug fix time, and all of them are significant within the context of an issue tracker, where bugs represent critical tasks to be managed, commits often serve as the tangible output of issue resolution process, and the time taken to fix bugs reflects the efficiency of the issue resolution process.

We did not include the Satisfaction & well-being dimension which typically captured through surveys, due to the impracticality of tracking these changes monthly. The Communication & collaboration dimension is also excluded, as issue entropy implicitly encapsulates information related to communication and collaboration within issue tracker. This approach aligns with the SPACE framework's guidelines, allowing for such adaptability.

In addition, we also introduce some relevant control variables to ensure a comprehensive evaluation of the project's productivity.

Number of new bugs. The number of new bugs is commonly used as a proxy for evaluating software quality in projects [37, 52, 64, 66]. We use this metric to capture project productivity in the Performance dimension, assessing the outcome of the software development process [25]. Similarly, we start to operationalize a project's productivity by the number of bugs found during a project-month. Note that we do not distinguish bug reports from core members or external contributors, because we measure the productivity of the whole project. On GitHub, the issue can be of various types, e.g., discussion, new feature request, improvement request, and so on. To categorize these issues, software developers often employ some keywords to tag them. However, because tagging is often project specific, we adopt Vasilescu et al.'s [64] method to distinguish bug issues from other issue types in this study. We set up a list of bug-related keywords, i.e., defect, error, bug, issue, mistake, incorrect, fault, and flaw, and then search for these words in both the issue tags and issue titles. If any tags or title of an issue contains at least one keyword, we identify it as a bug issue.

Number of new commits. In many studies focusing on the OSS development and community, the number of commits is considered as the productivity metric, although it is not comprehensive[52, 62, 63]. We adopt this widely used productivity indicator to capture project productivity in the Activity dimension, assessing the output of the software development process [25]. Note that we count the commits from all contributors rather than from external developers only, because we measure the impact on the productivity of the whole team. Similarly, as the productivity data, we compute the number of new commits in every project-month. A possible threat lies in that maintainers may change the origins of commits, for example, by "cherry-picking" in pull-requests [29], which can cause us to miss some contributions made by external contributors. Fortunately, we find such cases are rare in our dataset after manual inspections.

Bug fix time. The bug fix time is also regarded as an important metric for productivity [26, 47]. We adopt this metric to capture project productivity in the Efficiency & Flow dimension, assessing the efficiency of the software development process [25]. To calculate the bug resolution time each project-month, we differentiate bugs from other types of issues using a consistent methodology. For each project-month, we tally all bugs closed within the respective time window and compute the duration from creation to closure for each. The average of these duration is then taken as the bug resolution time for that project-month.

Control variables. Based on prior software engineering literature [29, 64, 65] and our experience, we include the following factors as control variables.

- **ProjAge:** The time span from project creation to the current month, measured in months. Growth dynamics may be different in younger vs. older projects [64].
- **NumCore** and **NumExt:** The number of active core developers and active external developers for the month. Active developers are defined as those who engage with the issue tracker system through at least one recorded activity within a given month.
- **NumFiles:** The number of files of the project until the current month. We use this as a proxy measure for the size of the project. Larger projects naturally receive more contributions [29, 64].
- **NumForks:** The number of forks of the project until the current month. We use this as a proxy measure for the popularity of the project. A higher number of forks typically correlates with an increased volume of bug reports[64] and augmented external contributions[65].
- **NumPrs:** The number of pull-requests opened within the current month. More opened pull-requests typically correlates with more people discussing concerns, and a longer issue fix time [41].
- **NumSubscr:** The number of issue subscriptions within the current month. A higher number of issue subscriptions typically correlates with less issue fix time [46].

4.2 Data Collection and Cleaning

To answer the three research questions presented in Section 1, we conduct an empirical study based on eight well-known open source projects which hosting their repositories on GitHub. The selection of the targeted software projects is based on the following four considerations:

- The selected projects should be engaged in software development, rather than serving as repositories for document storage or course teaching.
- To ensure a robust analysis and enable verification, we have chosen projects that not only have been in development for over a decade but also have consistently remained active. These projects are characterized by having a substantial and engaged user base, a historical record of at least 10,000 issues, and a minimum of 1,000 contributors.
- The selected projects represent a diverse sample of projects in terms of application domains, such as a popular deep-learning library (Tensorflow), a web development framework (React), a programming language (Go).
- Taking into account the trend of the increasing involvement of company in open source development, our sample includes a subset of corporate-driven software projects.

Ultimately, we selected eight renowned open source software projects for our study. The projects “golang/go”, “twbs/bootstrap”, “facebook/react”, and “tensorflow/tensorflow” are corporate-driven, whereas “rails/rails”, “vuejs/vue”, “electron/electron”, and “rust-lang/rust” are community-driven initiatives. To ensure that our dataset covers the entire process of each issue from creation to resolution and records as much user behavior as possible, we collect the issues of each project and all events related to the issues through the GitHub REST API. It does not include issues and related events that have been removed from GitHub, which are usually not considered meaningful by managers. We then excluded any action data performed by the bot (i.e. automated processes presented as GitHub users who perform event-driven actions) because we only considered human participation.

On GitHub, when multiple labels are added or removed simultaneously, each label is recorded as an individual event. To maintain as much parity as possible between different event types, we merged events of adding or removing labels by the same user at the same time. In total, we collected

229,130 issues and 3,831,808 events across these eight repositories, encompassing 30 distinct event types. See the replication package for details [2].

4.3 Regression Analysis

We calculate monthly variables for each project spanning from its creation to October 2023. For issue entropy and each control variable, we standardized them to a mean of zero and a standard deviation of one. This ensures that all estimated coefficients in the model are on the same scale so that we can perform comparative analysis. Additionally, we manually remove about 1% of values as outliers to ensure the models are robust against outliers [48].

To construct our linear models, we employ panel regression analysis, a robust econometric technique well-suited for our dataset, which is inherently unbalanced panel data. The use of simple OLS multivariate linear regression is not appropriate in this context; it fails to account for the intrinsic differences across the various projects and over the distinct time periods, as noted by Wooldridge [68].

Moreover, panel regressions provide another benefit, their intrinsic ability to handle commercial, social, and technical confounding factors without explicit enumeration. For example, project-specific factors (e.g., project domain, etc.) are treated as unobserved time invariant within the regression models. This is particularly useful when facing numerous confounding factors that are challenging to list exhaustively in statistical analyses—the selection of a subset for inclusion could introduce selective biases. While, the independent variables' effects could be precisely estimated without worrying about potential interactions and selective biases with many confounding factors when we do not aim for causality[68]. Intuitively, each project has its own characteristics, so we use the project-specific fixed effects models.

For each dependent variable, we use the least-squares dummy variable (LSDV) estimator to estimate the parameters in the project-specific fixed effects models. After we finish the model estimation, we perform a series of regression diagnostics for examining the time-specific effects and empirically justifying the use of fixed effects models. These regression diagnostics include time-fixed effects testing, F-test for (pFtest), Hausman Test (pHtest), Heteroskedasticity testing. Given that our sampled projects consist of 4 company projects and 4 non-company ones, it is natural to investigate if issue entropy's impacts on project's productivity are sensitive to these project characteristics. Therefore, we perform the same regression analyses to the two subsamples. The results are reported accordingly. All the panel regressions, if not otherwise stated, are performed with R's plm package [21]. We follow the ASA's principles to present and interpret statistical significance [67].

5 RESULTS

In this section, we report the results of regression analyses for research questions, and all data and code have been made publicly available for download.

Moreover, it's essential to understand that our regression models are intended to pinpoint correlation rather than causation. In interpreting the findings, we give some conjectures that suggest possible causal relationships. This approach is in line with standard practices in empirical research examining human and social behaviors [12, 20]. However, these conjectured causal links should be regarded as speculative and not definitive, which require further empirical investigation to be substantiated [45].

[2] https://github.com/anonymous2024CSCW/2024CSCW-replication-package

5.1 RQ1: What is the correlation between external contributions and the number of new bugs in OSS projects, and what factors could alter this dynamic?

To answer RQ1, we model the number of new bugs as a function of issue entropy. Our model accounts for a variety of control factors, including project age, the number of active core developer, the number of files, the number of forks. Additionally, we have included interaction terms between issue entropy and each of the control factors.

Table 5 summarizes the regression results across three different project samples. Models B1 use the data of all 8 sampled projects and thus represent whole sample regression results. In contrast, Models B2 use the data of 4 non-company projects, while Models B3 use the data of 4 company projects, providing subsample regression results. The explanatory power of the regression models is substantial, as evidenced by the high values of the multiple R^2 and adjusted multiple R^2, and the significance of the F-statistic. Below we describe what these models indicate.

Whole Sample Regression Results. In Model B1, issue entropy ($\mathcal{H}_{issue}$) shows a significant positive correlation with the number of new bugs, as denoted by largest coefficient of 29.974. It suggests that an increase of external contributions in issue tracker, corresponds to a significant rise in the number of new bugs within OSS projects. This finding is consistent with [66], which posits that a greater volume of code contributions from external developers may lead to a higher frequency of bugs. This could be attributed to the fact that external contributors, who may lack technical expertise, tend to produce code that is more susceptible to bugs.

For the model's control factors, ProjAge and NumCore are positively correlated to the new bugs of OSS projects, while NumForks and NumFiles are negatively. However, the effect of NumFiles fails to show statistical significance. This suggests that, in OSS projects, those that are older, have fewer forks, and possess larger active core development teams tend to discover more bugs.

For the model's interaction terms, the interaction between issue entropy and project age ($\mathcal{H}_{issue}$:ProjAge) shows a substantial negative coefficient of -12.768 in relation to new bugs. This implies that the effect of external contributions on new bugs introduction reduces as projects mature, possibly due to better issue management practices that evolve over time. Furthermore, the interaction between issue entropy and the number of active core developers ($\mathcal{H}_{issue}$: NumCore) exhibits a significant negative coefficient of -3.745. This result implies that having more active core developers can reduce the potential negative impact of substantial external contributions on software quality. This may be due to the fact that the increase in active core team members alleviates the decision-making pressure caused by a large influx of external contributions [60].

A positive significant interaction with the number of files ($\mathcal{H}_{issue}$:NumFiles, Coeff = 2.067) suggests that the complexity introduced by a larger codebase may exacerbate the external contributions' effect on bug introduction. Lastly, the interaction with the number of forks ($\mathcal{H}_{issue}$:NumForks, Coeff = 8.499) indicates that projects with more forks and external contributions tend to have more new bugs, possibly due to the challenges of integrating changes from a diverse set of contributors.

Subsample Regression Results. In Model B2 and B3, the correlation between issue entropy and the number of new bugs is consistent, which indicates a robust link between them. However, the coefficients indicating strongest effect in the non-company sample (Coeff = 78.747) and a relatively lower but still significant effect in the company sample (Coeff = 25.797). This suggests that projects without company support may be more sensitive to events complexity in issue tracker, leading to a higher incidence of new bugs.

For the model's control factors, all of them fails to show statistical significance in Model B2. While in Model B3, all control factors exhibits a consistent correlation with Model B1.

For the model's interaction term, $\mathcal{H}_{issue}$:ProjAge exhibits a consistent correlation across models B2 and B3. However, the ranking of coefficients indicate stronger effect in the non-company sample

Table 1. Regression Models for New Bugs

	Whole Sample Model B1		Noncompany Sample Model B2		Company Sample Model B3	
	Coeffs	Std. Error	Coeffs	Std. Error	Coeffs	Std. Error
$\mathcal{H}_{issue}$	29.974***	1.772	78.747***	13.876	25.797***	2.759
ProjAge	9.559***	1.479	−2.943	4.576	27.436***	2.795
NumCore	18.916***	2.550	12.476	6.729	15.949***	2.930
NumFiles	−1.594	2.264	2.200	5.652	−13.720***	3.860
NumForks	−9.258***	1.719	19.606	19.416	−9.871**	3.809
$\mathcal{H}_{issue}$:ProjAge	−12.768***	1.505	−45.598***	5.063	−9.415***	1.896
$\mathcal{H}_{issue}$:NumCore	−3.745**	1.302	4.712	2.455	−0.848	2.551
$\mathcal{H}_{issue}$:NumFiles	2.067*	0.904	9.670***	2.230	−19.614***	4.884
$\mathcal{H}_{issue}$:NumForks	8.499***	1.364	83.308***	18.997	24.634***	2.956
Multiple R^2	0.631		0.662		0.695	
Adjusted Multiple R^2	0.626		0.654		0.687	
F	193.778***		117.782***		118.079***	

***p<0.001, **p<0.01, *p<0.05

(Coeff = -45.598), while in the company sample, the effect is comparatively weaker (Coeff = -9.415). A plausible explanation could be that formal testing techniques and test automation are expensive and require sponsorship, while the user base is often the only choice for non-company projects, requiring a certain time to build up [1].

Interestingly, within Model B2, the interaction term $\mathcal{H}_{issue}$:NumFiles demonstrates a correlation consistent with that of Model B1 with coefficient of 9.670. In contrast, Model B3 presents an inverse relationship, with coefficient of -19.614. This discrepancy may indicate that in non-company projects, an increase in files tends to amplify the complexity and entropy of issues, leading to a higher likelihood of bug occurrence. Conversely, in company projects, the presence of more files may be associated with better organizational structures or more robust development protocols, which can help to systematically address and mitigate complexities, thereby reducing the propensity for new bugs despite high issue entropy.

Furthermore, in Models B2 and B3, the interaction term $\mathcal{H}_{issue}$:NumForks consistently aligns with the correlation observed in Model B1. However, the coefficient for this term in Model B2 significantly exceeds that in Model B3 (83.308 vs. 24.634). This discrepancy could be attributable to non-company projects facing heightened challenges in integrating changes from a diverse array of contributors.

In summary, we answer RQ1 as below.

(1) External contributions are significantly positively correlated with number of newly found bugs.
(2) The correlation between external contributions and number of newly found bugs decreases as projects age and increase as fork counts, especially for non-company projects.

5.2 RQ2: How are external contributions correlated with the number of new commits in OSS projects, and what factors may affect this connection?

To answer RQ2, we model the number of new commits as a function of issue entropy. Our model accounts for a variety of control factors, including project age, the number of active core developer, the number of active external contributor, the number of files, the number of forks, the number of opened pull-requests. Additionally, we have included interaction terms between issue entropy and each of the control factors.

Table 4 summarizes the regression results across three different project samples. Similarly, Models C1 use the data of all 8 sampled projects and thus represent whole sample regression results. In contrast, Models C2 use the data of 4 non-company projects, while Models C3 use the data of 4 company projects, providing subsample regression results. The explanatory power of the regression models is substantial, as evidenced by the high values of the multiple R^2 and adjusted multiple R^2, and the significance of the F-statistic. Below we describe what these models indicate.

Whole Sample Regression Results. In Model C1, issue entropy ($\mathcal{H}_{issue}$) is positively and significantly associated with the number of new commits, as denoted by second largest coefficient of 178.629. This suggests that more external contributions correlates with a higher number of commits in OSS projects.

For the model's control factors, all variables except for "ProjAge" and "NumExt" are positively correlated with the number of commits in OSS projects. This suggests that OSS projects that are younger, have more active core developers, fewer external contributors, more forks, and a greater number of files tend to receive more commits.

For the model's interaction terms, the interaction term $\mathcal{H}_{issue}$:NumCore is significantly positive (Coeff = 87.461), which indicates that the presence of a larger core team amplifies the effect of external contributions on the number of new commits. The interpretation of the results shall be similar to the above in RQ1. Similarly, the interaction term $\mathcal{H}_{issue}$:NumForks also shows a positive correlation (Coeff = 28.196). This suggests that projects with more forks, which may indicate a higher level of community interest or activity, see an increased effect of external contributions on new commits. Moreover, the interaction term $\mathcal{H}_{issue}$:NumPRs is positive (Coeff = 75.110), signifying that more external contributions correlates with more pull request activity resulting in new commits.

On the other hand, the negative interaction term $\mathcal{H}_{issue}$:ProjAge (Coeff = -23.208) implies that the positive impact of external contributions on new commits diminishes as projects mature. This could be due to the fact that core developers contribute the majority of the code and, over time, their contributions to the codebase may diminish as they shift their focus towards maintenance efforts. The negative interaction term $\mathcal{H}_{issue}$:NumExt (Coeff = -47.646) could reflect the complexity of coordinating and integrating work from a wider and possibly less experienced or less familiar contributor base, which might add pressure to code integration efforts. Lastly, the negative interaction term $\mathcal{H}_{issue}$:NumFiles (Coeff = -58.708) suggests that, in larger codebases, high issue entropy may lead to fewer new commits, possibly due to the increased difficulty of managing and understanding a more extensive and potentially complex system.

Subsample Regression Results. The observed significant negative correlation between issue entropy and the number of new commits (Coeff = -506.913) in Model C2, as opposed to the positive correlation in Model C3 (Coeff = 117.514), might suggest that non-company projects could be struggling to manage and translate complex events from the issue tracker into code changes, a task possibly more efficiently handled by company projects with their more structured support systems.

For the model's control factors, ProjAge and NumExt in Model C2 fail to show statistical significance. Otherwise, there's a consistent correlation for all factors between Model C2 and C3, except for NumForks (-601.792 vs. 65.131).

Table 2. Regression Models for New Commits

	Whole Sample Model C1		Noncompany Sample Model C2		Company Sample Model C3	
	Coeffs	Std. Error	Coeffs	Std. Error	Coeffs	Std. Error
$\mathcal{H}_{issue}$	178.629***	19.147	−506.913***	143.160	117.514**	41.612
ProjAge	−128.931***	11.810	13.677	43.329	−115.521***	27.355
NumCore	177.265***	23.220	232.562***	62.118	120.925***	31.409
NumExt	−83.619***	18.517	−20.203	36.526	−57.047*	28.495
NumFiles	189.622***	18.708	133.368**	44.946	199.551***	31.480
NumForks	69.865***	14.393	−601.792***	179.886	65.131*	31.795
NumPRs	115.260***	13.713	140.515***	18.833	152.697***	26.133
$\mathcal{H}_{issue}$:ProjAge	−23.208*	11.729	247.862***	51.570	−18.818	17.444
$\mathcal{H}_{issue}$:NumCore	87.461***	14.531	2.157	27.143	126.929***	28.016
$\mathcal{H}_{issue}$:NumExt	−47.646***	13.374	−19.935	27.479	−50.842*	20.888
$\mathcal{H}_{issue}$:NumFiles	−58.708***	10.121	−86.485***	20.774	−108.649**	40.347
$\mathcal{H}_{issue}$:NumForks	28.196*	12.889	−936.015***	186.377	52.308	28.540
$\mathcal{H}_{issue}$:NumPRs	75.110***	9.444	63.725***	13.844	64.600*	29.456
Multiple R^2	0.829		0.868		0.760	
Adjusted Multiple R^2	0.825		0.864		0.752	
F-statistic	377.606***		272.528***		112.904***	

***p<0.001, **p<0.01, *p<0.05

For the model's interaction terms, the positive interaction term $\mathcal{H}_{issue}$:ProjAge (Coeff = 247.862) in Model C2 implies that the negative impact of external contributions on new commits diminishes in non-company project as it matures. While the negative interaction term $\mathcal{H}_{issue}$:NumForks (Coeff = -936.015) in Model C2 implies that the negative impact of external contributions on new commits would be further exacerbated in non-company projects as its forks increase. All interaction terms in Model C3 exhibit a consistent correlation with Model C1.

In summary, we answer RQ2 as below.

(1) External contributions have a distinct correlation with the number of new commits: it's inversely related in non-company projects but shows a positive correlation in company projects.
(2) For non-company projects, the negative effect of external contributions on new commits decreases with project maturity and an increasing number of opened pull requests, but it grows as the number of forks and files rises.
(3) For company projects, a larger active core team and more opened pull requests appear to amplify the positive influence of external contributions on new commits. However, this positive effect is lessened by a greater number of active external contributors and a larger codebase.

5.3 RQ3: To what extent do external contributions correlate with bug fix time in OSS projects, and what factors could strengthen or weaken this association?

To answer RQ3, we model project's average bug fix time as a function of issue entropy. Our model accounts for a variety of control factors, including project age, the number of active core developer, the number of active external contributor, the number of files, the number of forks, the number of opened pull-requests and the number of issue subscribed. Additionally, we have included interaction terms between issue entropy and each of the control factors.

Table 3 summarizes the regression results across three different project samples. Similarly, Models T1 use the data of all 8 sampled projects and thus represent whole sample regression results. In contrast, Models T2 use the data of 4 non-company projects, while Models T3 use the data of 4 company projects, providing subsample regression results. The explanatory power of the regression models is substantial, as evidenced by the high values of the multiple R^2 and adjusted multiple R^2, and the significance of the F-statistic. Below we describe what these models indicate.

Whole Sample Regression Results. In Model T1, issue entropy ($\mathcal{H}_{issue}$) is negatively and significantly associated with the bug fix time, as denoted by third largest coefficient of -23.585. This suggests that more external contributions correlates with a less time for solve bugs in OSS projects.

For the model's control factors, all variables except for ProjAge and NumCore are positively correlated with the number of commits in OSS projects. This suggests that OSS projects that a greater number of files, forks and issue subscribed tend to increase bug fix time.

For the model's interaction terms, both $\mathcal{H}_{issue}$:NumCore (Coeff = -18.041) and $\mathcal{H}_{issue}$:NumForks (Coeff = -10.184) show significant negative coefficients. This indicates that a larger active core team and a higher number of forks intensify the impact of external contributions on reducing bug fix time. Contrarily, the interaction term $\mathcal{H}_{issue}$:NumFiles shows a positive relationship (Coeff = 11.618), suggesting that a higher number of forks actually weaken the influence of external contributions on reducing bug fix time.

Subsample Regression Results. In Model T2 and T3, the correlation between issue entropy and bug fix time is consistent, which indicates a robust link between them.

For the model's control factors, there's a consistent correlation for all factors between Model T1 and T2 except for NumFiles and NumForks. Notably, the correlation with NumFiles is not statistically significant in Model T2. In Model T3, NumFiles (Coeff = 60.607) and NumPRs (Coeff = 51.215) are positively correlated with bug fix time, whereas ProjAge (Coeff = -29.314) shows a negative correlation.

For the model's interaction term, the positive interaction term $\mathcal{H}_{issue}$:ProjAge (Coeff = 71.763) in Model T2 implies that the impact of external contributions on reducing bug fix time diminishes in non-company projects as they mature. The interaction term $\mathcal{H}_{issue}$:NumSubscr with a coefficient of -13.259 in Model T2 indicates that for non-company projects, the effect of external contributions on reducing bug fix time is increased as the number of issue subscribers increases.

In models T2 and T3, the negative coefficient associated with $\mathcal{H}_{issue}$:NumForks indicates that although external contributions are generally associated with shorter bug fix time, this relationship is amplified as the number of forks increases. It suggests that with more forks in a project, the impact of external contributions to reducing bug fix time becomes more significant. This may reflect that a greater number of forks brings about broader developer participation and resource sharing, enhancing the project's internal problem-solving capabilities and efficiency, rather than solely attributing the collaborative effects to the increase in external contributions[14].

In the context of non-commercial projects, the observed external contributions are inversely related to bug resolution time, particularly in conjunction with an increasing number of forks, can be more pronounced. This phenomenon might be explained by the nature of collaboration and

Table 3. Regression Models for bug fix Time

	Whole Sample Model T1		Noncompany Sample Model T2		Company Sample Model T3	
	Coeffs	Std. Error	Coeffs	Std. Error	Coeffs	Std. Error
$\mathcal{H}_{issue}$	−23.585**	8.015	−149.195**	51.424	−48.639**	14.694
ProjAge	2.809	5.042	56.846**	18.184	−29.314**	10.129
NumCore	5.685	9.715	12.403	26.878	−15.234	13.713
NumFiles	36.170***	9.065	−13.887	31.387	60.607***	13.496
NumForks	12.132*	6.136	−144.759*	71.642	16.218	14.225
NumPRs	10.625	5.696	6.140	7.546	51.215***	11.364
NumSubscr	40.159**	12.476	56.053**	20.888	45.378	23.930
$\mathcal{H}_{issue}$:ProjAge	3.715	5.052	71.763***	20.860	−0.632	7.611
$\mathcal{H}_{issue}$:NumCore	−18.041**	6.233	−45.942***	13.318	3.069	11.026
$\mathcal{H}_{issue}$:NumFiles	11.618*	5.754	20.749	11.477	25.275	15.384
$\mathcal{H}_{issue}$:NumForks	−10.184*	5.093	−206.529**	71.660	−35.095***	10.119
$\mathcal{H}_{issue}$:NumPRs	−3.464	4.178	−3.986	5.328	−21.068	11.512
$\mathcal{H}_{issue}$:NumSubscr	−5.425	3.707	−13.259*	5.947	−9.189	17.755
Multiple R^2	0.340		0.409		0.322	
Adjusted Multiple R^2	0.326		0.390		0.298	
F-statistic	37.514***		25.728***		16.474***	

***p<0.001, **p<0.01, *p<0.05

contribution in non-commercial or open-source environments. In such settings, a large number of forks often indicates a high level of community engagement and a diverse contributor base. As a result, complex issues might be addressed more rapidly due to the collective expertise, voluntary participation, and parallel problem-solving that typify these projects. Unlike in company settings, where processes and decision-making can be more centralized and bureaucratic, non-commercial projects benefit from a more decentralized, meritocratic approach that can often lead to more efficient issue resolution, especially for complex problems

In summary, we answer RQ3 as below.

(1) External contributions are associated with a significant reduction in bug fix time.
(2) For non-company projects, the effect of external contributions on reducing bug fix time grows with an increasing number of active core developers, forks, and issue subscribers, while it decreases as the project ages.
(3) For company projects, more forks appear to amplify the effect of external contributions on reducing bug fix time.

6 DISCUSSION

6.1 Discussion of the Finding

The results and findings of **RQ1** reveal relationships between a OSS project's external contributions and the its number of new bugs. It suggests that an increase of external contributions correspond to a significant rise in the number of new bugs within OSS projects, with non-company projects being

Table 4. Regression Models for NewCommits

	Whole Sample Model 1		Noncompany Sample Model 1		Company Sample Model 1	
	Coeffs	Std. Error	Coeffs	Std. Error	Coeffs	Std. Error
H_{conv}	123.482**	35.197	138.759*	56.348	95.321***	25.417
ProjAge	−57.235*	25.471	−74.921**	18.952	−41.513**	16.105
NumForks	76.832***	19.857	83.472*	27.135	67.367**	22.987
NumIssues	−23.475**	7.345	−19.405**	5.764	−15.672*	7.123
NumPrs	47.052*	11.287	44.803*	13.689	39.028*	10.523
NumFiles	71.368***	15.264	68.592**	14.382	59.712***	12.786
NumCores	24.590**	7.321	22.031*	8.905	20.847*	8.213
NumExternals	31.023**	7.122	28.934**	8.053	27.722*	8.433
H_{conv}:ProjAge	42.532*	13.581	38.742**	11.016	26.315**	8.734
H_{conv}:NumCores	−19.544*	8.901	−17.182*	7.948	−11.753**	5.992
H_{conv}:NumFiles	12.243*	5.243	10.891*	4.950	8.523**	3.987
H_{conv}:NumForks	−31.782**	9.431	−28.035**	8.874	−22.513*	7.339
H_{conv}:NumPrs	−9.127*	4.670	−8.013*	4.123	−7.013**	3.418
H_{conv}:NumExternals	−11.201**	3.984	−10.562*	3.794	−9.122*	3.413
Multiple R^2	0.638		0.631		0.592	
Adjusted Multiple R^2	0.615		0.537		0.565	
F-statistic	14.356***		12.853***		11.275***	

***p<0.001, **p<0.01, *p<0.05

especially sensitive to this effect. Additionally, this relationship diminishes as projects mature, yet intensifies with an increase in fork counts, particularly for non-company projects.

Building upon the insights from our research, it appears that the heightened influx of bugs in non-company OSS projects with more external contributions may be a result of the considerable strain on maintainers. Maintainers are tasked with navigating the barrage of contributions ranging from feature suggestions to bug reports, which escalates their workload and decision-making responsibilities [8, 60]. The typically lower technical skill level of external contributors may inadvertently lead to more bugs being introduced, particularly when maintainers are swamped [1]. Moreover, an increase in project forks correlates with a surge in contributions [65], further intensifying the strain on maintainers. This phenomenon is pronounced in non-company projects due to less support and fewer resources than their company-backed counterparts [1]. However, as projects mature, they often cultivate a supportive user community, which can help mitigate some of these challenges for maintainers [1].

Turning to **RQ2**, the contrasting results between non-company and company projects are particularly revealing. Non-company projects experience a negative correlation between external contributions and new commits, implying that these projects may struggle to efficiently convert the effort in the issue tracker into code contributions. This could be due to a lack of resources or structured processes that are more commonly found in company projects [1], which show a positive correlation. However, as these projects become mature, they seem to develop mechanisms to better assimilate external inputs, seen in the decreased negative impact of external contributions. Interestingly, the significant negative interaction between external contributions and the number of forks in non-company projects suggests that the energy and attention of external contributors

Table 5. Regression Models for NewBugs

	Whole Sample Model 2		Noncompany Sample Model 2		Company Sample Model 2	
	Coeffs	Std. Error	Coeffs	Std. Error	Coeffs	Std. Error
H_{conv}	5.832*	2.583	7.186**	2.084	4.052***	1.076
ProjAge	22.357**	7.239	30.214***	6.321	15.932**	5.098
NumForks	−17.124**	5.156	−19.425**	4.967	−13.190*	5.027
NumIssues	9.218***	2.178	10.823***	1.765	8.102**	2.358
NumPrs	−6.892*	2.123	−7.032*	2.148	−4.879*	1.987
NumFiles	11.719*	4.173	14.803*	3.615	9.672*	3.712
NumCores	2.987**	0.943	3.418**	1.105	1.729**	0.864
NumExternals	−1.231*	0.612	−1.536*	0.565	−0.843*	0.469
H_{conv}:ProjAge	−4.953*	2.215	−6.432**	1.867	−3.185*	1.548
H_{conv}:NumCores	1.045*	0.514	1.106*	0.481	0.675*	0.432
H_{conv}:NumFiles	3.891**	1.453	4.517**	1.107	2.845**	1.029
H_{conv}:NumForks	−5.112**	1.784	−6.321**	1.378	−3.789**	1.236
H_{conv}:NumPrs	1.302*	0.538	1.453*	0.514	0.902*	0.456
H_{conv}:NumExternals	−0.964*	0.412	−1.089*	0.387	−0.674*	0.363
Multiple R^2	0.705		0.699		0.670	
Adjusted Multiple R^2	0.630		0.623		0.594	
F-statistic	19.652***		18.043***		15.124***	

***p<0.001, **p<0.01, *p<0.05

may be dispersed across many independently developed [51] or divergent forks [13, 34, 50] that are not intended for merging back into the original project. This dispersion could lead to a dilution of constructive issue tracker activities, as efforts are spread out over multiple, separate development paths rather than being concentrated on enhancing the main codebase.

Turning to **RQ3**, the negative correlation between external contributions and bug fix time implies that increased involvement from external contributors might lead to faster bug resolution. The visibility and attention that come with more external contributors mean that bugs are more likely to be spotted and addressed promptly [53]. This may be attributed to the wide array of ideas and skills that external contributors bring to the table, enhancing the detection and addressing of bugs [2]. In non-company project, the positive interaction with project age suggests that as non-company projects mature, the initially negative impact of external contributions on bug fix time diminishes [25]. Mature projects likely develop more efficient workflows and stronger community practices that can handle the influx of contributions without significant delays in bug resolution. In company projects, the positive coefficients for the number of files and pull-requests might reflect a tipping point where structural and procedural complexity begins to outweigh the benefits of more contributors and hinders quick bug resolution.

Interestingly, a strong interaction with external contributions is associated with a decrease in commits, yet it is also correlated with a reduction in bug fix time. This suggests a nuanced dynamic wherein enhanced discussion and engagement around issues from external contributors—implied by substantial external contribution—may not directly translate into a greater volume of code contributions, but does seem to positively affect the efficiency of the bug resolution process. Different from number of commits, the same factors—substantial external contribution and an

Table 6. Regression Models for bug fix Time

	Whole Sample Model 3		Noncompany Sample Model 3		Company Sample Model 3	
	Coeffs	Std. Error	Coeffs	Std. Error	Coeffs	Std. Error
H_{conv}	42.735***	12.649	48.023**	13.751	29.435**	8.731
ProjAge	−39.751**	15.213	−44.512*	17.109	−21.195*	9.642
NumForks	60.453*	17.432	62.341**	18.583	48.321**	14.127
NumIssues	−29.785**	8.419	−31.642*	9.201	−22.874*	7.832
NumPrs	25.731**	6.842	27.953*	7.136	16.920*	5.321
NumFiles	48.572**	12.103	51.328**	13.905	41.324**	11.203
NumCores	19.231**	6.342	20.490*	7.083	15.421*	5.874
NumExternals	−13.214*	4.198	−14.728*	4.791	−10.865*	3.671
H_{conv}:ProjAge	−24.175*	7.832	−26.432*	8.216	−16.742*	6.509
H_{conv}:NumCores	−13.012*	4.117	−14.451*	4.795	−10.034*	3.765
H_{conv}:NumFiles	20.236*	5.413	21.857*	6.062	16.239*	4.987
H_{conv}:NumForks	−29.451*	7.419	−30.876*	7.932	−20.875*	6.342
H_{conv}:NumPrs	−11.753*	3.496	−12.832*	3.892	−8.976*	2.964
H_{conv}:NumExternals	−9.234*	2.989	−10.125*	3.254	−6.785*	2.632
Multiple R^2	0.582		0.573		0.552	
Adjusted Multiple R^2	0.477		0.464		0.435	
F-statistic	11.763***		11.102***		9.657***	

***p<0.001, **p<0.01, *p<0.05

abundance of forks—can create an environment ripe for quick bug identification and resolution. With more contributors working independently across various forks, the collective problem-solving capacity is amplified [51]. This decentralization enables a broader and more rapid response to bugs as they arise, often leading to a swifter bug resolution process. The enhanced diversity and redundancy within such a vibrant development ecosystem mean that bugs are likely to be tackled from multiple angles, increasing overall efficiency in resolving issues.

6.2 Recommendation

The observations and analyses derived from our study provoke a number of actionable recommendations for practitioners and maintainers of open-source software (OSS) projects. Accordingly, we provide a set of guidelines which aims to unlock the potential of external contributions, thereby enhancing the sustainability of OSS projects.

- **Optimizing Issue Tracker Management:** Given the increased workload associated with substantial external contributions, as seen in the rise of bug reports in non-company projects, optimizing issue tracker management is crucial. Strategies that streamline issue tracking and prioritize reports can help maintainers manage this workload more effectively. For example, a dynamic prioritization strategy could use event logs to gauge contributors' interest in specific issues, allowing the system to adjust priorities based on real-time engagement. This approach not only helps maintainers focus on the most critical issues but also encourages active participation by demonstrating to contributors that their input directly influences project priorities.

- **Balancing Contribution and Workload:** The finding that increased involvement of external contributors correlates with both a rise in new commits and faster bug resolution suggests the need for balancing the advantages of these contributions with the accompanying workload. Clear contribution guidelines and a structured onboarding process can help maintainers manage this influx, ensuring that new contributions are high-quality and align with the project's needs. This aligns with our discussion that a well-structured community can reduce the negative impacts of high issue entropy while enhancing the benefits.
- **Leveraging the Community in Bug Resolution:** Our study shows that more external contributions is linked to faster bug resolution times. This indicates the importance of fostering a supportive and engaged community, where the active participation of external contributors can expedite the identification and resolution of bugs. Encouraging community-driven bug solutions not only speeds up problem-solving but also alleviates the workload of the core team, while helping to cultivate potential new members for the core team from within the community.
- **Mitigating Fork Dispersion Effects:** The negative effects of fork dispersion on the concentration of contributions emphasize the necessity for better integration of efforts across different forks. As we discussed, forks can compete for contributors' attention on the primary project. Encouraging contributors to merge their changes back into the main repository can help maintain momentum and maximize the benefits of community involvement while minimizing the fragmentation of effort.

6.3 Implications

Forum [23], issue tracker[39], and pull request [59] are considered the primary avenues for external contributors to engage in the development of open-source software (OSS), as typically outlined in the contribution guidelines of OSS projects(e.g., facebook/react [54]). While, In our experiments, we have focused solely on capturing the activities of external contributors within the issue tracker. There has been study linking forum discussions to issues issue[61] and correlating issues with pull requests [5]. Combined with their work, centering on issues allows us to organize the activities of external contributors systematically and to create a more comprehensive view of their engagement. By capturing the complexity of these activities through issue entropy, we can gain crucial insights into the participation of external contributors and the sustainability of OSS [51]. Additionally, some studies define the degree of external contributor engagement in terms of the number of contributors [57] or the quantity of commits submitted [39]. However, we argue that issue entropy provides a more nuanced capture of external contributors' involvement.

Further research into the patterns of change in issue entropy can meaningfully enhance open-source software (OSS) development. It is essential to employ pattern mining and time series analysis technologies to excavate the evolutionary patterns and future trends of issue entropy over time, to identify and assess the different factors and events that influence the fluctuation of project issue entropy, and to analyze their impact on the sustainability and prosperity of OSS projects. With a thorough understanding of the patterns and factors that contribute to issue complexity, we can develop monitoring tools and establish guidelines to promote efficient collaboration during the development and maintenance of OSS projects.

Moreover, communities like CHAOSS [16] has collected a vast range of qualitative and quantitative metrics to understand and evaluate the health of open-source communities, projects, and ecosystems, including 8 metrics related to issue tracker [15]. The concept of issue entropy presented in this paper offers a new perspective on the health of the OSS project ecosystem and has the potential to be integrated into the existing collection of metrics.

Importantly, the concept of event sequences and their complexity, which we have applied to issue tracking, can be extended to various collaborative contexts. In these settings, contributions often take the form of interactions, discussions, decisions, and task resolutions, all of which can be captured through event sequences. For instance, in academic collaborations, issue trackers or similar tools can monitor progress and feedback, while in business or research environments, event sequences may represent project milestones or communication exchanges, revealing the depth of collaboration. By applying information theory in these domains, we can uncover intricate patterns of contributions in diverse collaborative contexts.

6.4 Threats to Validity

Construct validity. This study introduces issue entropy as a measure of complexity within issue trackers, derived from the activity logs. Shannon entropy, a comprehensive metric of system complexity, has been utilized across various studies [3, 32, 38]. We elucidate how Shannon entropy captures system complexity by describing it in the context of data emission from a device and employ the same model to explain the mechanism by which issue entropy captures the complexity of issue trackers. Consequently, we are confident that there is no significant threat to the construct validity from this perspective.

Internal validity. We took multiple measures to ensure that data collection process avoided most of the perils summarized in [10, 35]. All projects under analysis were substantial in size, with established governing structures and practices, utilizing both issue trackers and pull-requests to manage tasks and contributions. We leveraged objective human activity records collected from GitHub, ensuring an unbiased analysis process. Mature, widely recognized analysis techniques were employed, and the use of fixed effects models in panel regressions was empirically justified.

In this study, we examined the correlation between issue entropy and OSS projects' productivity, following the recommendations set forth in [25], employing new bugs, new commits, and bug fix time as productivity metrics to capture diverse dimensions of project performance.

External validity. As for the external validity, we are confident that there is no significant threat in the design of issue entropy. Although our study focuses only on the activities of external contributors, it is a general model for measuring complexity within issue trackers. This is due to the fact that we do not make any assumptions about the software project or issue tracker in our definition and calculation. However, we admit that our result of empirical study may not generalizable to all open source software projects. One potential limitation is that all 8 projects are large-scale which ensures adequate traceable records of issue-level activities. We suggest caution for applying our findings in the context of small-scale or inactive open source projects. It is our future work to include more projects in our studies.

7 CONCLUSION

In the realm of open-source software (OSS) development, the resolution of issues is not just a technical task but a pivotal activity that drives ongoing enhancement and secures a project's endurance. Although the significance of external contributors is recognized, the level their involvement in issue tracker still lack full quantification and clarity, and the relationship between their involvement and project productivity remains unclear.

In this work, we propose issue entropy, a metric applies principles of information theory to scrutinize granular details within a project's issue-related activities, providing a unique lens to understand and assess external contributions. We deploy issue entropy to measure the complexity of events sequence in issue tracker quantitatively and examine its relationship with the introduction of new bugs, commit frequency, and bug resolution times—metrics that serve as proxies for OSS project productivity. Our results indicate a strong positive correlation between external contributions and

the influx of new bugs, a fluctuating correlation with commit activity contingent on whether a project is company, and a marked negative correlation with bug resolution timeframes. Furthermore, we observed notable interactions between external contributions and additional variables, such as project maturity and file quantity.

In light of these findings, we discuss the implications and offer a suite of recommendations designed to unlock the potential of external contributions. Additionally, we contemplate the broader implications of issue entropy, which provides novel insights into the vitality of the OSS ecosystem and could support subsequent scholarly inquiry and practical endeavors.

Acknowledgments

This work is supported by NSFC No.62332005

References

[1] Mark Aberdour. 2007. Achieving Quality in Open Source Software. *IEEE Softw.* 24, 1 (jan 2007), 58–64. doi:10.1109/MS.2007.2

[2] Ritu Agarwal. 2010. The Effects of Diversity in Global , Distributed Collectives : A Study of User Participation in Open Source Projects. https://api.semanticscholar.org/CorpusID:15201970

[3] Edward B. Allen, Sampath Gottipati, and Rajiv Govindarajan. 2007. Measuring Size, Complexity, and Coupling of Hypergraph Abstractions of Software: An Information-Theory Approach. *Software Quality Journal* 15, 2 (jun 2007), 179–212. doi:10.1007/s11219-006-9010-3

[4] Mohammad AlMarzouq, Li Zheng, Guang Rong, and Varun Grover. 2005. OPEN SOURCE: CONCEPTS, BENEFITS, AND CHALLENGES. *Communications of AIS* (2005), 756–784. Issue No.16.

[5] Zakarea Alshara, Hamzeh Eyal Salman, Anas Shatnawi, and Abdelhak-Djamel Seriai. 2023. ML-Augmented Automation for Recovering Links Between Pull-Requests and Issues on GitHub. *IEEE Access* 11 (2023), 5596–5608. doi:10.1109/ACCESS.2023.3236392

[6] Ann Barcomb, Andreas Kaufmann, Dirk Riehle, Klaas-Jan Stol, and Brian Fitzgerald. 2020. Uncovering the Periphery: A Qualitative Survey of Episodic Volunteering in Free/Libre and Open Source Software Communities. *IEEE Transactions on Software Engineering* 46, 9 (2020), 962–980. doi:10.1109/TSE.2018.2872713

[7] Anne Bartel-Radic and Nicolas Lesca. 2011. Do intercultural teams need "requisite variety" to be effective? *Management international* 15 (2011), 89–104. https://api.semanticscholar.org/CorpusID:144654102

[8] Olga Baysal, Holmes Reid, and Michael W. Godfrey. 2014. No issue left behind: reducing information overload in issue tracking. In *FSE 2014: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*.

[9] Dane Bertram, Amy Voida, Saul Greenberg, and Robert Walker. 2010. Communication, Collaboration, and Bugs: The Social Nature of Issue Tracking in Small, Collocated Teams. In *CSCW '10: Proceedings of the 2010 ACM conference on Computer supported cooperative work*.

[10] Christian Bird, Peter C. Rigby, Earl T. Barr, David J. Hamilton, Daniel M. German, and Prem Devanbu. 2009. The promises and perils of mining git. In *2009 6th IEEE International Working Conference on Mining Software Repositories*. 1–10. doi:10.1109/MSR.2009.5069475

[11] Barry Boehm and Victor R. Basili. 2001. Software Defect Reduction Top 10 List. *Computer* 34, 1 (jan 2001), 135–137. doi:10.1109/2.962984

[12] Robert L. Brennan and Dale J. Prediger. 1981. Coefficient Kappa: Some Uses, Misuses, and Alternatives. *Educational and Psychological Measurement* (1981), 687–699. Issue NO.3.

[13] John Businge, Moses Openja, Sarah Nadi, and Thorsten Berger. 2022. Reuse and maintenance practices among divergent forks in three software ecosystems. *Empirical Software Engineering* (2022), 54(1–47). Issue No.2.

[14] Harold Carey and Patrick Laughlin. 2012. Groups perform better than the best individuals on letters-to-numbers problems: Effects of induced strategies. *Group Processes & Intergroup Relations - GROUP PROCESS INTERGROUP RELA* 15 (03 2012), 231–242. doi:10.1177/1368430211419174

[15] CHAOSS. 2017. *All metrics in CHAOSS.* https://chaoss.community/kbtopic/all-metrics/ [Online; accessed 02-January-2024].

[16] CHAOSS. 2017. *CHAOSS - Community Health Analytics in Open Source Software.* https://chaoss.community/ [Online; accessed 02-January-2024].

[17] Chisel. 2024. *What Is Issue Tracking? Definition and Best Practices.* https://chisellabs.com/glossary/what-is-issue-tracking/ [Online; accessed 02-January-2024].

[18] Grzegorz Chrupala. 2012. Learning from evolving data streams: online triage of bug reports. In *EACL '12: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*.

[19] Joseph Ciervo, Shih Chuan Shen, Kristin Stallcup, Abraham Thomas, Michael A. Farnum, Victor S. Lobanov, and Dimitris K. Agrafiotis. 2019. A new risk and issue management system to improve productivity, quality, and compliance in clinical trials. *JAMIA open* (2019), 216–221. Issue No.2.

[20] Josh Cowls and Ralph Schroeder. 2015. Causation, Correlation, and Big Data in Social Science Research. *Policy & Internet* 7 (08 2015), n/a–n/a. doi:10.1002/poi3.100

[21] Yves Croissant and Giovanni Millo. 2008. Panel data econometrics in R: The plm package(Article). *Journal of Statistical Software* (2008), 1–43. Issue No.2.

[22] Kevin Crowston, Hala Annabi, and James Howison. 2003. Defining Open Source Software Project Success. *Proceedings of the International Conference on Information Systems* (06 2003).

[23] Sherae Daniel, Tingting Chung, and Pratyush Sharma. 2020. The Impact of Anonymous Peripheral Contributions on Open Source Software Development. *AIS Transactions on Human-Computer Interaction* (01 2020), 146–171. doi:10.17705/1thci.00133

[24] Wei Li; Wenjun Wu; Huaimin Wang; Xueqi Cheng; Huajun Chen; Zhihua Zhou; Rong Ding. 2017. Crowd intelligence in AI 2.0 era. *Frontiers of Information Technology & Electronic Engineering* (2017), 15–43. Issue No.1.

[25] Nicole Forsgren, Margaret-Anne Storey, Chandra Maddila, Thomas Zimmermann, Brian Houck, and Jenna Butler. 2021. The SPACE of Developer Productivity: There's more to it than you think. *Queue* 19 (02 2021), 20–48. doi:10.1145/3454122.3454124

[26] Emanuel Giger, Martin Pinzger, and Harald Gall. 2010. Predicting the fix time of bugs. In *Proceedings of the 2nd international workshop on recommendation systems for software engineering*. 52–56.

[27] Github. 2024. *Permissions for each role.* https://docs.github.com/en/organizations/managing-user-access-to-your-organizations-repositories/managing-repository-roles/repository-roles-for-an-organization [Online; accessed 02-January-2024].

[28] Georgios Gousios, Eirini Kalliamvakou, and Diomidis Spinellis. 2008. Measuring developer contribution from software repository data. In *MSR '08: Proceedings of the 2008 international working conference on Mining software repositories*.

[29] Georgios Gousios, Martin Pinzger, and Arie Deursen. 2014. An exploratory study of the pull-based software development model. doi:10.1145/2568225.2568260

[30] Robert A. Guth. 2006. Trolling the Web for Free Labor, Software Upstarts Are New Force. *Wall Street Journal(Eastern Edition)* (2006), A1–A12. Issue No.114.

[31] M. Hanisch, C. Haeussler, S. Berreiter, and S. Apel. 2018. Developers' Progression from Periphery to Core in the Linux Kernel Development Project. *Academy of Management Annual Meeting Proceedings* 2018, 1 (2018), 14263.

[32] Ahmed E. Hassan. 2009. Predicting faults using the complexity of code changes. In *2009 IEEE 31st International Conference on Software Engineering*. 78–88. doi:10.1109/ICSE.2009.5070510

[33] J. Howison and K. Crowston. 2014. Collaboration through open superposition: A theory of the open source way(Article). *MIS Quarterly: Management Information Systems* (2014), 29–50. Issue 1.

[34] James Howison and James Herbsleb. 2013. Incentives and integration in scientific software production. *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW* (02 2013), 459–470. doi:10.1145/2441776.2441828

[35] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2014. The Promises and Perils of Mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (Hyderabad, India) *(MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 92–101. doi:10.1145/2597073.2597074

[36] Brian C. Keegan, Shakked Lev, and Ofer Arazy. 2015. Analyzing Organizational Routines in Online Knowledge Collaborations: A Case for Sequence Analysis in CSCW. *Computer Science* (2015).

[37] Foutse Khomh, Tejinder Dhaliwal, Ying Zou, and Bram Adams. 2012. Do faster releases improve software quality? An empirical case study of Mozilla Firefox. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. 179–188. doi:10.1109/MSR.2012.6224279

[38] Kapsu Kim, Yeongil Shin, and Chisu Wu. 1996. Complexity Measures for Object-Oriented Program Based on the Entropy. 127 – 136. doi:10.1109/APSEC.1995.496961

[39] Rajiv Krishnamurthy, Varghese Jacob, Suresh Radhakrishnan, and Kutsal Dogan. 2016. Peripheral Developer Participation in Open Source Projects. *ACM Transactions on Management Information Systems* 6 (01 2016), 1–31. doi:10.1145/2820618

[40] Saerom Lee, Hyunmi Baek, and Sehwan Oh. 2020. The role of openness in open collaboration: A focus on open-source software development projects. *ETRI Journal* (2020), 196–204. Issue 2.

[41] Zhixing Li, Yue Yu, Tao Wang, Yan Lei, Ying Wang, and Huaimin Wang. 2023. To Follow or Not to Follow: Understanding Issue/Pull-Request Templates on GitHub . *IEEE Transactions on Software Engineering* 49, 04 (April 2023), 2530–2544. doi:10.1109/TSE.2022.3224053

[42] Aron Lindberg, Nicholas Berente, James Gaskin, and Kalle Lyytinen. 2016. Coordinating Interdependencies in Online Communities: A Study of an Open Source Software Project. *Information Systems Research* (2016), 751–772. Issue 4.

[43] Walid Maalej and Hans-Jörg Happel. 2010. Can development work describe itself?. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*. 191–200. doi:10.1109/MSR.2010.5463344

[44] Christian A. Mahringer. 2021. Analyzing Digital Trace Data to Promote Discovery – The Case of Heatmapping. In *Business Process Management Workshops*.

[45] Savage Mike and Burrows Roger. 2007. The Coming Crisis of Empirical Sociology. *Sociology* (2007), 885–899. Issue No.5.

[46] Marco Ortu, Bram Adams, Giuseppe Destefanis, Parastou Tourani, Michele Marchesi, and Roberto Tonelli. 2015. Are Bullies More Productive? Empirical Study of Affectiveness vs. Issue Fixing Time. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. 303–313. doi:10.1109/MSR.2015.35

[47] Marco Ortu, Giuseppe Destefanis, Mohamad Kassab, Steve Counsell, Michele Marchesi, and Roberto Tonelli. 2015. Would you mind fixing this issue? an empirical analysis of politeness and attractiveness in software developed using agile boards. In *Agile Processes in Software Engineering and Extreme Programming: 16th International Conference, XP 2015, Helsinki, Finland, May 25-29, 2015, Proceedings 16*. Springer, 129–140.

[48] Jason Osborne and Amy Overbay. 2004. The Power of Outliers (and Why Researchers Should Always Check for Them). *Pract. Assess. Res. Eval.* 9 (01 2004).

[49] Rohan Padhye, Senthil Mani, and Vibha Singhal Sinha. 2014. A study of external community contribution to open-source projects on GitHub. In *MSR 2014: Proceedings of the 11th Working Conference on Mining Software Repositories*.

[50] Ei Pa Pa Pe Than, Laura Dabbish, and James Herbsleb. 2021. Open Collaborative Writing: Investigation of the Fork-and-Pull Model. *Proceedings of the ACM on Human-Computer Interaction* 5 (04 2021), 1–33. doi:10.1145/3449211

[51] Ayushi Rastogi and Nachiappan Nagappan. 2016. Forking and the Sustainability of the Developer Community Participation – An Empirical Investigation on Outcomes and Reasons. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. 102–111. doi:10.1109/SANER.2016.27

[52] Baishakhi Ray, Daryl Posnett, Premkumar Devanbu, and Vladimir Filkov. 2017. A large-scale study of programming languages and code quality in GitHub. *Commun. ACM* (2017), 91–100. Issue No.10.

[53] Eric S. Raymond and Bob Young. 2001. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly & Associates, Inc., USA.

[54] React. 2014. *contribution guide for react*. https://legacy.reactjs.org/docs/how-to-contribute.html [Online; accessed 02-January-2024].

[55] Gregorio Robles, Jesús M. González-Barahona, Carlos Cervigón, Andrea Capiluppi, and Daniel Izquierdo-Cortázar. 2014. Estimating Development Effort in Free/Open Source Software Projects by Mining Software Repositories: A Case Study of OpenStack. In *MSR 2014: Proceedings of the 11th Working Conference on Mining Software Repositories*.

[56] Anastasia Ruvimova, Alexander Lill, Jan Gugler, Lauren Howe, Elaine Huang, Gail Murphy, and Thomas Fritz. 2022. An Exploratory Study of Productivity Perceptions in Software Teams. In *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*. 99–111. doi:10.1145/3510003.3510081

[57] Pankaj Setia, Balaji Rajagopalan, Vallabh Sambamurthy, and Roger Calantone. 2012. How Peripheral Developers Contribute to Open-Source Software Development. *Information Systems Research* 23 (03 2012), 144–163. doi:10.2307/23207878

[58] C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27, 3 (1948), 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x

[59] Vibha Singhal Sinha, Senthil Mani, and Saurabh Sinha. 2011. Entering the circle of trust: developer initiation as committers in open-source projects. In *MSR '11: Proceedings of the 8th Working Conference on Mining Software Repositories*.

[60] Xin Tan, Minghui Zhou, and Brian Fitzgerald. 2020. Scaling Open Source Communities: An Empirical Study of the Linux Kernel. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. 1222–1234.

[61] James Tizard, Peter Devine, Hechen Wang, and Kelly Blincoe. 2023. A Software Requirements Ecosystem: Linking Forum, Issue Tracker, and FAQs for Requirements Management. *IEEE Transactions on Software Engineering* 49, 4 (2023), 2381–2393. doi:10.1109/TSE.2022.3219458

[62] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. 2013. StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge. In *2013 International Conference on Social Computing*. 188–195. doi:10.1109/SocialCom.2013.35

[63] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark G.J. van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. 2015. Gender and Tenure Diversity in GitHub Teams. In *CHI '15: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*.

[64] Bogdan Vasilescu, Yue Yu, Huaimin Wang, Premkumar Devanbu, and Vladimir Filkov. 2015. Quality and Productivity Outcomes Relating to Continuous Integration in GitHub. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE 2015)*. Association for Computing Machinery, New York, NY, USA,

805–816. doi:10.1145/2786805.2786850

[65] Liang Wang, Zhiwen Zheng, Xiangchen Wu, Baihui Sang, Jierui Zhang, and Xianping Tao. 2023. Fork Entropy: Assessing the Diversity of Open Source Software Projects' Forks. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 204–216. doi:10.1109/ASE56229.2023.00168

[66] Zhendong Wang, Yang Feng, Yi Wang, James A. Jones, and David Redmiles. 2020. Unveiling Elite Developers' Activities in Open Source Projects. 29, 3, Article 16 (jun 2020), 35 pages. doi:10.1145/3387111

[67] Ronald L. Wasserstein and Nicole A. Lazar. 2016. The ASA's Statement on p -Values: Context, Process, and Purpose. *American Statistician* 70, 2 (2016), 129 – 133. https://search-ebscohost-com-s.libyc.nudt.edu.cn:443/login.aspx?direct=true&db=mth&AN=116101602&lang=zh-cn&site=ehost-live

[68] Jeffrey M. Wooldridge. 1999. Introductory Econometrics: A Modern Approach. https://api.semanticscholar.org/CorpusID:56503746

[69] Y. Ye and K. Kishida. 2003. Toward an understanding of the motivation of open source software developers. In *Software Engineering, 2003. Proceedings. 25th International Conference on.*

[70] Shengyu Zhao, Xiaoya Xia, Brian Fitzgerald, Xiaozhou Li, Valentina Lenarduzzi, Davide Taibi, Rong Wang, Wei Wang, and Chunqi Tian. 2024. OpenRank Leaderboard: Motivating Open Source Collaborations Through Social Network Evaluation in Alibaba. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 346–357. doi:10.1145/3639477.3639734

[71] Minghui Zhou and Audris Mockus. 2011. Does the initial environment impact the future of developers. In *2011 33rd International Conference on Software Engineering (ICSE)*. 271–280. doi:10.1145/1985793.1985831

Received January 2024; revised October 2024; accepted February 2025