The dynamics of leadership and success in software development teams

Lorenzo Betti,[1,*] Luca Gallo,[1] Johannes Wachs,[2,3,4] and Federico Battiston[1,†]

[1] *Department of Network and Data Science, Central European University, Vienna, Austria*
[2] *Institute for Data Analytics and Information Systems, Corvinus University of Budapest, Budapest, Hungary*
[3] *Institute of Economics, HUN-REN Centre for Economic and Regional Studies, Budapest, Hungary*
[4] *Complexity Science Hub, Vienna, Austria.*

From science to industry, teamwork plays a crucial role in knowledge production and innovation. Most studies consider teams as static groups of individuals, thereby failing to capture how the micro-dynamics of collaborative processes and organizational changes determine team success. Here, we leverage fine-grained temporal data on software development teams to gain insights into the dynamics of online collaborative projects. Our analysis reveals an uneven workload distribution in teams, with stronger heterogeneity correlated with higher success, and the early emergence of a lead developer carrying out the majority of work. Moreover, we find that a sizeable fraction of projects experience a change of lead developer, with such a transition being more likely in projects led by inexperienced users. Finally, we show that leadership change is associated with faster success growth, in particular for the least successful projects. Our work contributes to a deeper understanding of the link between team evolution and success in collaborative processes.

I. INTRODUCTION

The production of innovation and knowledge increasingly relies on collective efforts. For example, teamwork is crucial in scientific research, where teams have demonstrated their effectiveness in fostering groundbreaking discoveries [1–3], or in industry, where teamwork is essential for the rapid development of innovative solutions [4]. The effectiveness of teamwork depends on the continuous integration of specialized knowledge of individuals [5, 6], and the ability to leverage constructive conflicts to generate novel insights [7, 8]. Extensive research has investigated characteristics of team members that are associated with performance, from the effect of team size [3], the interdependencies among team members [9], and their diversity in terms of gender [10–12], expertise [13], prior experiences [2, 14], and ethnicity [15, 16]. In this way, the results of such collaborative efforts often go beyond the sum of individual contributions thanks to the emergence of synergies among team members [17, 18], conditional on the team's organization.

In a team, not all members are equal [19]. Successful teamwork often relies on the management of tensions by leaders [20, 21], who organize work through the division of complex tasks into sub-tasks among different team members [22]. Even in self-organizing teams, lacking a predefined hierarchical structure, specific team members can emerge as leading figures who carry out a sizeable fraction of the work and are eventually responsible for the project advancement [23–25]. The organizational dynamic and evolving nature of teams impact team structure over time, as roles and responsibilities shift to adapt to new challenges [26, 27]. For instance, adjustments in the distribution of labor, the emergence and shift of leadership, and the turnover of team members all hinder coordination in teams, explaining why social and coordination skills are growing in value as teams get larger and teamwork more prevalent [28].

Indeed, various conceptual frameworks have highlighted the need to explicitly consider team dynamics to properly understand how teams function and their success [27, 29, 30]. Nevertheless, most of the insights we have consider teams as static entities, with little empirical research accounting for their temporal evolution. This is largely due to difficulties in accessing or collecting temporal data tracking and measuring team activities across time [31, 32]. In science, benefiting from the availability of large-scale curated publication data, a few studies have recently explored the concept of "persistent teams" [33] – researchers who consistently collaborate together over time – showing that "fresh" teams made of new collaborators produce more impactful research [34]. However, publication data only records the outcome of teamwork, lacking information about the process of the collaborative effort, such as the specific contributions and activities of the authors of each paper. In organization theory, team activity is typically tracked through multi-period observations, where team members are surveyed at various intervals. Yet this approach suffers from inherent limitations of low temporal resolution because it would necessitate frequently surveying team members. Such frequent surveys can disrupt teamwork and lead to survey fatigue, eventually compromising data quality and preventing the effective investigation of team dynamics at a microscopic scale [31]. While fine-grained data about team processes can be collected through other methods like sensors that generate real-time data [35] or in controlled laboratory experiments [36], these approaches typically cover only a limited number of teams and are difficult to scale.

Open-source software development offers an ideal opportunity to investigate collaborative dynamics at a large

*Electronic address: betti_lorenzo@phd.ceu.edu
†Electronic address: battistonf@ceu.edu

scale and with a fine-grained resolution [37–39]. Software developers track every change to the software codebase using version control systems (e.g., Git) and store their software in online public repositories (e.g., GitHub). For instance, by analyzing the contribution history of a project via its commit log (a record of every edit made to code), it is possible to obtain information about who made which modification to the software and when. The analysis of this source of data has identified patterns where few developers often perform the majority of contributions in a project and most developers make few [40–42]. This tendency towards centralization, widespread across collaborative software development, has been employed to develop heuristics to identify the core members of teams [43]. Changes in the activity of core developers, as well as changes in its composition, can offer novel insights about how teams function and the relationship to project success [44]. The dynamics of the main developers may vary significantly across projects. For instance, Linus Torvalds, who created the Linux kernel in 1991, is nowadays continuing to dominate the development effort of the project [45] and defines himself as a "benevolent dictator of Planet Linux" [46]. Differently, Pandas' creator Wes McKinney has stopped to be actively involved in the development of Pandas after five years [47] and other developers have succeeded as the main developers of the project [48]. While both Linux and Pandas library are examples of successful projects, their contrasting leadership dynamics highlight a gap in our understanding of the causes and consequences of the emergence of key team members, their potential turnover, and the effect of such changes on project success.

To advance the understanding of team dynamics in open-source software projects, we leverage a curated dataset tracking the activity of over 40 000 developers in more than 6000 projects from 2014 to 2022, along with their success metrics [49, 50]. Our dataset contains the entire development history of the Rust programming language, allowing us to study projects and contributors' activity over time from their inception. First, we examine the distribution of work as reflected by the distribution of commits among team members, study the activity patterns of the lead developers – users responsible for the largest share of commits – and correlate it with project success. Then, we identify repositories where the lead developer changes during the project's lifespan, identifying profound redistribution of workload and a potential reorganization of the team. Finally, we investigate the association between the change of lead developer and the success growth of the project after the transition by comparing those repositories to similar ones that did not change the lead developer. Our findings demonstrate the interplay of team dynamics and performance in open-source software projects, suggesting that changes in team organization have implications for the success of the project.

II. RESULTS

A. Emergence of a lead developer

Software developers track their changes to the software codebase through commits, whose distribution across team members can measure their work contributions [42] and be informative about their roles in a project [43]. We begin by analyzing the distribution of commits among team members to characterize the distribution of work in software development teams. We define a team as the set of developers who make at least one commit to a repository (i.e., the developers who can autonomously modify the software codebase). By ranking the developers of a repository by their total number of commits, we measure the fraction of commits authored by developers as a function of their rank. Fig. 1a shows that the most active developer (rank = 1) authors more than half of the total number of commits. In contrast, the second most active developer typically accounts for only around 10-20% of the total, while the rest is done by the other team members. This observation highlights the presence of a "lead developer" who carries the majority of the workload in a repository, alongside other developers contributing to a lesser extent. Those properties are consistent across teams of different sizes as shown in Fig. 1a (see Fig. S1a for the distribution of repositories across different team sizes) and through the lifetime of repositories (see Fig. S2). This persistent nature of the workload distribution suggests a potential advantage from such a centralization.

One potential aspect where this advantage may manifest is projects' success. To test for a relationship between heterogeneous workload distribution and success, we first quantify the heterogeneity of the workload distribution using the relative effective team size [42]. This metric, defined as $2^H/N$, where H is the binary entropy of the distribution of commits among team members and N is the size of the team, ranges from $1/N$ (i.e., one single developer makes all commits) to 1 (i.e., the workload is evenly distributed among all members). Consequently, a smaller relative effective team size indicates a more uneven distribution of commits among team members. Then, we employ the number of stars and downloads as metrics of repositories' success. Such metrics track two different dimensions of success: stars can be considered as a proxy for repositories' popularity, similar to likes in social media [51], whereas the number of downloads reflects the software usage, thus tracking technical quality. Our analysis reveals an inverse relationship between repositories' success and their relative effective team size, as displayed in Fig. 1b for the number of stars and Fig. 1c for the number of downloads (see Fig. S1b for the distribution of the number of repositories for different relative effective team sizes). The relationship is supported by Spearman's correlation test, yielding a correlation coefficient of at least $\rho = -0.28$ for stars and $\rho = -0.11$ for downloads across all team sizes ($p < 0.001$; see Table I

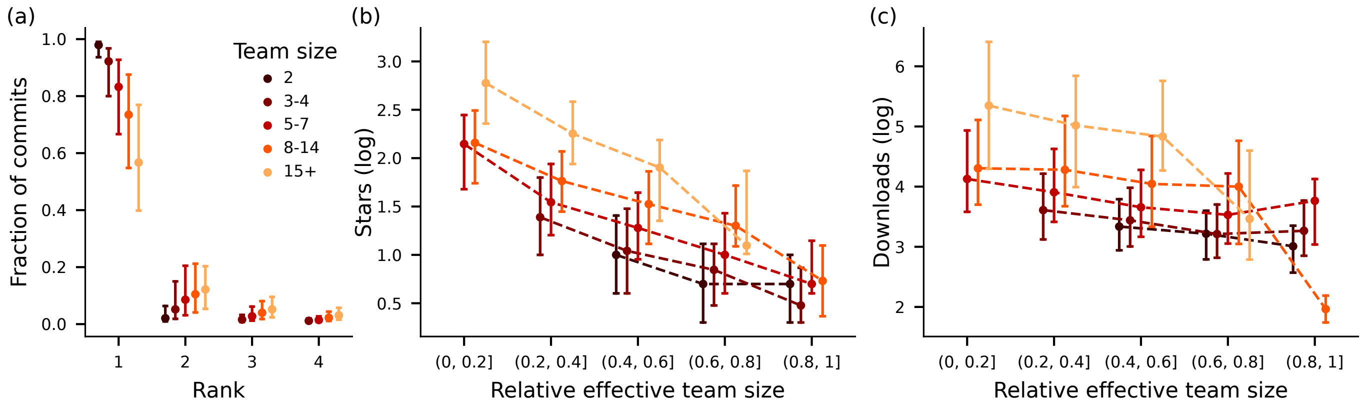

FIG. 1: **Workload distribution within teams and relationship with success.** (**a**) Median fraction of commits authored by the r-th most active developer of a repository stratified by team size. The most active developer makes more than half of the total number of commits while other developers contribute substantially less, regardless of the size of the team. (**b-c**) Median number of stars (**b**) and downloads (**c**) as a function of the relative effective team size stratified by team size. The more heterogeneous the workload distribution in the team, the higher the success. The Spearman's rank test returns $p < 0.001$ for all team sizes. The number of stars and downloads are incremented by one unit. Error bars range from the 25th to the 75th percentile of the distributions.

for details). To check if the age of repositories affects the results of the correlation, we repeated the analysis considering the relative effective team size and success at different moments of repositories' lifetime, finding consistent results (see Fig. S2 and Table I).

Our observation aligns with prior research on software development teams [41], confirming that even in a relatively new programming language such as Rust, the most successful teams of software developers have an uneven workload distribution [42]. Furthermore, as already found in [40] where three large open-source projects were studied, our results show that the workload distribution becomes heterogeneous already within the first year of activity. Differently from previous studies, in the following we provide a characterization of the behaviors of the lead developers of each repository. Beyond static analyses, we focus in particular on team dynamics, revealing changes in workload distribution and how this impacts a project's success.

B. Characterization of lead developers' activity

Here, we describe the activity patterns of lead developers compared to non-lead developers. To begin, we compare the distribution of their inter-commit time, defined as the time elapsed between two consecutive commits authored by the same user (regardless of the repository in which the commit is made). Fig. 2a displays the distributions for lead and non-lead developers separately, showing distinct characteristics for those two sets of users. In particular, lead developers exhibit a higher frequency of commits. Indeed, their inter-commit time distribution is left-skewed compared to that of non-lead developers, displaying a prominent peak around 30 minutes (the peak in non-lead developers' inter-commit time ranges between one day and one week). This difference turns into lead developers being more likely to have longer streaks of consecutive commits, as depicted in the inset of Fig. 2a. The inset shows the probability distribution of making more than E consecutive commits whose inter-commit time is smaller than $\Delta t = 30$ minutes (the observation is robust across other values of Δt as can be seen in Fig. S3). Lead developers are indeed three times more likely to make streaks of at least 10 commits ($P(\geq 10) = 0.0097$) compared to non-lead developers ($P(\geq 10) = 0.0030$).

Another aspect of the activity pattern of developers is the number of repositories they contribute to. We first show in Fig. 2b the distribution of the number of repositories in which lead and non-lead developers author at least one commit. In addition to being the most active developers in their repositories, lead developers contribute to a median of 4 repositories (interquartile range IQR 2-8) while non-lead developers are active in only a median of 1 repository (IQR 1-2). To gauge insights on the extent to which lead developers work on multiple projects simultaneously, we introduce the repository switch time of developers. This is defined as the time elapsed between a developer's first commit on a repository and their first commit on a different one, namely the time elapsed between a developer initiating work on one repository and transitioning to another. Fig. 2c shows the distribution of repository switch time for lead and non-lead developers separately. The difference with the inter-commit time distribution in Fig. 2a suggests that the trains of consecutive commits shortly separated in time (i.e., inter-commit time of 30 minutes) are mostly done on the same repository, as lead developers switch from one repository to another on a daily to weekly basis. The rapid decay of the repository switch time after one week, in addition to the number of repositories in which lead developers are active, implies that they work

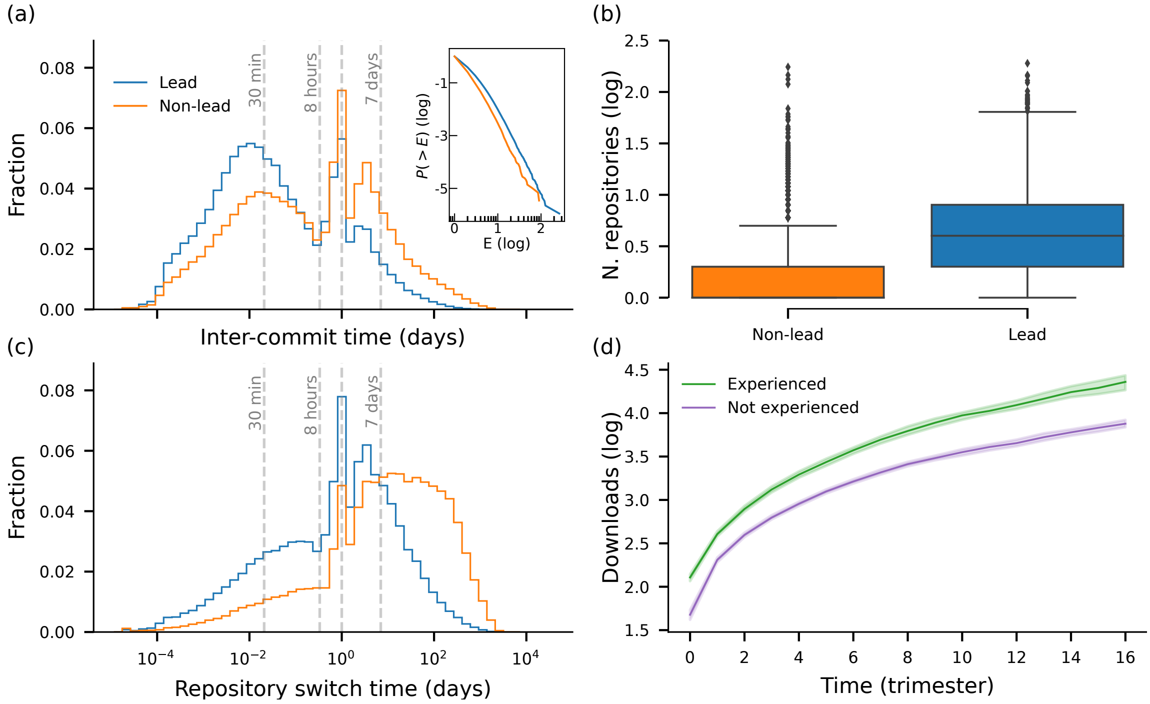

FIG. 2: **Characterization of lead developers' activity.** (**a**) Distribution of the inter-commit times and cumulative distribution of the number of commits close in time (inset) for lead and non-lead developers. Lead developers exhibit higher frequency of commits and longer streaks of consecutive commits. (**b**) Distribution of the number of repositories in which lead and non-lead developers are active. Lead developers are involved in a larger number of repositories. (**c**) Distribution of the repository switch time of lead and non-lead developers. Lead developers tend to switch from one project to another on a daily to weekly basis. (**d**) Number of downloads across repositories' lifetime stratified by lead developers' experience (median and 95% confidence interval). Repositories led by experienced developers are downloaded more compared to those led by inexperienced ones. Time is binned into trimesters.

concurrently on multiple projects and focus their efforts within the Rust ecosystem. Differently, non-lead developers move between repositories over longer periods. This may be a consequence of non-lead developers being active in fewer projects and possibly having lesser involvement in the Rust ecosystem.

Lead developers often contribute to multiple repositories, and their experience as core members in previous projects might prove beneficial for new endeavours. For this reason, we ask: Does an experienced lead developer impact the success of a repository? We define the lead developer of a repository i as experienced if they have been the lead developer of other repositories before starting to contribute to repository i. We identify 2744 (43%) repositories led by experienced lead developers and show in Fig. 2d the median number of downloads these repositories receive across their lifetime, compared to repositories whose lead developer never led a repository before. We observe that repositories led by developers with previous experience receive more downloads over their lifetime. Since downloads can be interpreted as an indicator of the technical quality of the software, we speculate that developers who led repositories in the past may leverage their previous experience in Rust projects to develop better software. By contrast, it is worth noticing that lead developers' previous experience does not affect the number of stars, indicating a lack of advantage in terms of popularity (see Fig. S4). We verified that the observations are not influenced by the inclusion of repositories with varying ages, as this could introduce bias due to the advantage older repositories may have in accumulating stars and downloads (see Fig. S4).

C. Lead developers of repositories can change

Lead developers emerge early in a project and display distinct activity patterns. Does the same individual consistently maintain such a role, or does this figure change in time? To identify such changes, we first aggregate developer activities (i.e., cumulative number of commits) and the success of repositories (i.e., cumulative number of stars and downloads) by trimesters. Then, we consider the lead developer of a repository at trimester t as the one who has made the largest share of commits up to that time (we excluded 206 repositories from the subsequent analyses to ensure the inclusion of repositories displaying meaningful changes; see details in the Methods section). We find that 618 repositories (10%) undergo a change of lead developer, with the majority of such changes occurring between the second and third year of a repository's lifetime, as shown in Fig. 3a. Most repositories undergoing a change in lead developer experience only one transition (92%). Therefore, we focus our subsequent analyses on the first and second lead developers, hereafter referred to as the old and new lead developers, respectively.

To further investigate how the transition from the old to the new lead developer occurs, the first aspect we examine is the dynamics of the transition . We show in Fig. 3b the relative number of new commits authored by the old and new lead developer around the time when the transition occurs (i.e., the time when the lead developer changes), averaged across repositories (repositories with no activity in a trimester are omitted). The result indicates a rapid transition with a drop in the activity of the former lead developer. One year before the transition, old and new lead developers make a comparable amount of new contributions on average, i.e., 30% of the total number of new commits. However, while the activity of old lead developers sharply declines, dropping to levels below 10%, the relative contribution of new leads largely increases, ultimately making a heavy portion of new commits (around 65%) in the subsequent year after the change. In particular, in 297 repositories, corresponding to 48% of projects that undergo a change, the old lead developer ceases to contribute entirely through commits. This stop in commit activity does not necessarily indicate a complete departure from the project, as contributors can remain involved in other activities within projects. Indeed, we observe that in 48 of such repositories (16%), old lead developers remain active by engaging in other tasks such as addressing issues and reviewing pull requests. This transition of core team members to administrative roles has been previously observed in open-source software development as a response to an abrupt increase of external attention [52].

Next, we ask if previous experience can explain changes to the role of the lead developer. To account for the temporal dimension, we consider the lead developer of a repository i as experienced if they have been the lead developer of other repositories before becoming the lead developer of repository i. We find that 11% of repositories initiated by inexperienced lead developers undergo a transition of lead developer (388 out of 3519), while this happens to 9% of those initiated by experienced lead developers (230 out of 2646). Although this difference looks small, it corresponds to a 30% increase in the odds of changing the lead developer when the initial lead lacks previous experience and it is deemed significant according to Fisher's exact test (odds ratio at 1.30, p-value $= 0.003$). We checked the robustness of the result against the year in which repositories were initiated by conducting additional tests restricted to repositories initiated after specific years. The results, shown in Fig. 3c, confirm that the association remains significant except for the last two years, possibly because of the reduction of the sample size ($N = 2339$ for 2019 and $N = 1027$ for 2020). In short, the experience of the initial lead developer is associated with a lower likelihood of change.

Finally, we investigate if lead developers changes may be explained by the success of the project before the change. For instance, existing literature on startups suggests a U-shape relationship between founder departure and growth rate, indicating that founders of startups experiencing either slow or rapid growth are more likely to depart than those of startups with intermediate growth rates [53]. Interestingly, we find no evidence supporting an association between previous success and the likelihood of changing the lead developer in the future within the Rust ecosystem (see Fig. S5 for details).

D. Repositories that change the lead developer perform better after the change

With the previous analyses, we have shown that a sizeable fraction of Rust repositories undergo a change of their lead developers, describing the dynamics and factors associated with this turnover. How does such a turnover relate to the repositories' future success?

To answer this question, we employ a matching approach to compare the success trend of those repositories (named "lead-change repositories") against the success of similar repositories whose lead developer did not change (named "lead-remain repositories"). Specifically, we design a stringed matching procedure to identify pairs of lead-change and lead-remain repositories that are similar in terms of team composition and success prior to the change of the lead developer (see Methods section for details). To ensure a reasonable fit, we restrict the analysis to a subset of 151 lead-change repositories (24%) displaying meaningful activity and success one year before the change of leader (more than 50 commits, 10 stars, and 100 downloads). We then monitor the difference in the success growth at time t (Δ_t) associated to the lead developer change as

$$\Delta_t = Y_t - \tilde{Y}_t$$

where $Y_t = S_t/S_{t_0}$ is the success of the lead-change repository (i.e., either number of stars or downloads) relative to the last pre-treatment period ($t_0 = -1$) and $\tilde{Y}_t$ is that of the matched repository. The value of Δ_t should be close to zero for $t \leq t_0$, indicating that the lead-change repository and its match exhibit similar success trajectory before the change. After t_0, Δ_t quantifies how the success growth varies in relation to the change of lead developer. We consider Δ_t for t ranging from $[-4, 4]$, namely, one year before and after the time in which the change occurs ($t = 0$).

We find that the change of lead developer is positively related to a stronger growth in success. Fig. 4a shows the success difference Δ_t for stars averaged across the 135 repositories with a suitable match (89% of the lead-change repositories). Notably, the difference in success growth is already positive during the first trimester in which the new lead developer takes over ($\Delta_{t=0} = 0.05$

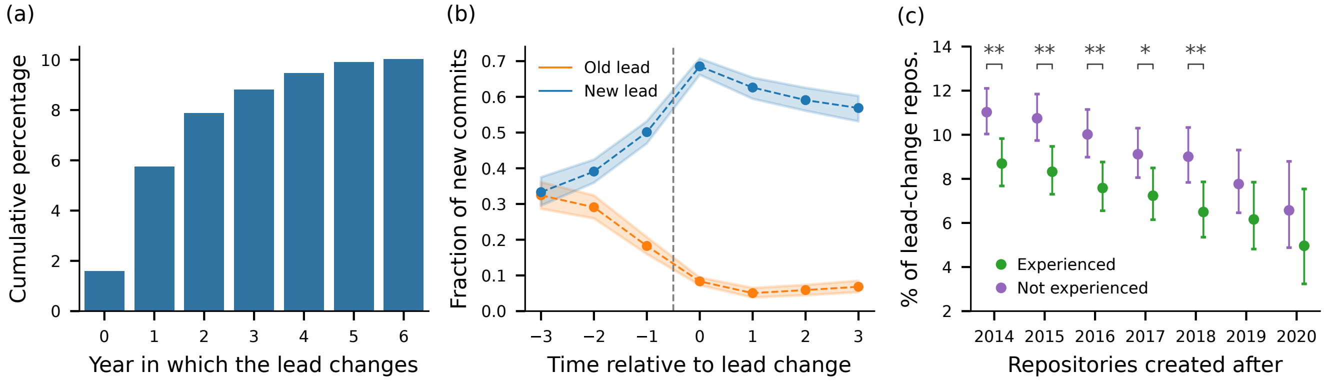

FIG. 3: **Lead developers can change across the lifetime of repositories.** (**a**) Cumulative percentage of repositories undergoing a lead developer change as a function of the number of years since their creation. Around 10% of repositories change their lead developer throughout their lifetime, with the majority occurring within the second and third year of activity. (**b**) Fraction of new commits authored by the old and new lead developer before and after the lead developer transition (mean and 95% confidence interval). After the transition (vertical dashed line), contributions from the old lead developer diminish rapidly. (**c**) Percentage of lead-change repositories stratified by the previous experience of the old lead developer. Each point refers to repositories created at a specific year or later. Repositories led by inexperienced lead developers exhibit a significantly higher likelihood to change their lead developer compared to those led by experienced ones, according to Fisher's exact test. Significance levels are denoted as follows: * for $p < 0.05$, ** for $p < 0.01$, and *** for $p < 0.001$. Error bars refer to 95% confidence intervals (Wilson score interval).

on average, 95% CI: 0.02 - 0.09) and keeps increasing during the whole year after the change (one year later, $\Delta_{t=4} = 0.23$ on average, 95% CI: 0.06 - 0.40). Same results hold for downloads (see Fig. S7), thus suggesting that the change of lead developer is associated with faster success growth with respect to both popularity and technical quality. The values of Δ_t before the change ($t < 0$) are close to zero, meaning that the success trends between the lead-change and their matched repositories are close on average before the transition. This underlines the good quality of the matching procedure. To check the robustness of the result, we also consider the median of the success difference Δ_t, showing that the observed positive association is not due to the skewness of the distribution of Δ_t (see Fig. S7).

Finally, we investigate how success before the transition and lead developer experience affect success growth. As depicted in Fig. 4b, we find that lowly successful repositories (i.e., those in the bottom 30% of the success distribution) have larger Δ_t in terms of number of stars when compared to average (i.e., those between the 50th and the 80th percentile) and top (i.e., top 10%) repositories. This indicates that repositories gaining relatively more visibility after the change of lead developers are the least popular. We found no significant differences in the case of downloads (see Fig. S8). Regarding the previous experience, we observe in Fig. 4c that repositories initiated by an experienced lead developer have a significant growth in terms of stars, whereas repositories initiated by inexperienced lead developers do not (see Fig. S9 for a discussion of the skewness of Δ_t). Finally, in Fig. S10, we consider the combined effect of both the old and the new lead developer's experience on success growth.

III. DISCUSSION

In this study, we used fine-grained data about software development teams to characterize the temporal dynamics of online collaborative projects. By tracking the activity of the most active contributor, we unveiled the emergence of a lead developer who exhibits a distinctive pattern of activity compared to non-lead developers, with lead developers emerging early in a project lifetime. Moreover, we show how an uneven distribution of workload among team members is positively associated with a repository's success. We identified a sizeable fraction of projects that undergo a change of their lead developer and revealed an association between such transitions and faster success growth.

Our analysis of open-source Rust projects reports that most of the work is carried out by one or a few developers and that this correlates with higher success. This evidence is well-documented in the literature and spans across different programming languages [40–42, 54, 55]. One possible explanation for such a correlation is the increase in efficiency due to the concentration of workload among few developers, which is likely to reduce the cost of coordination [42, 56]. At the same time, this distribution of work may result in a concentration of knowledge about the functioning of the software around a few developers, thus posing the project at risk should the main developers leave. Drawing on a well-known concept in the software development literature, projects have a small "truck factor", meaning that the number of key developers who would need to be incapacitated, i.e., hit by a truck, to prevent further development of the project is small [57–61]. The tension between efficiency and project

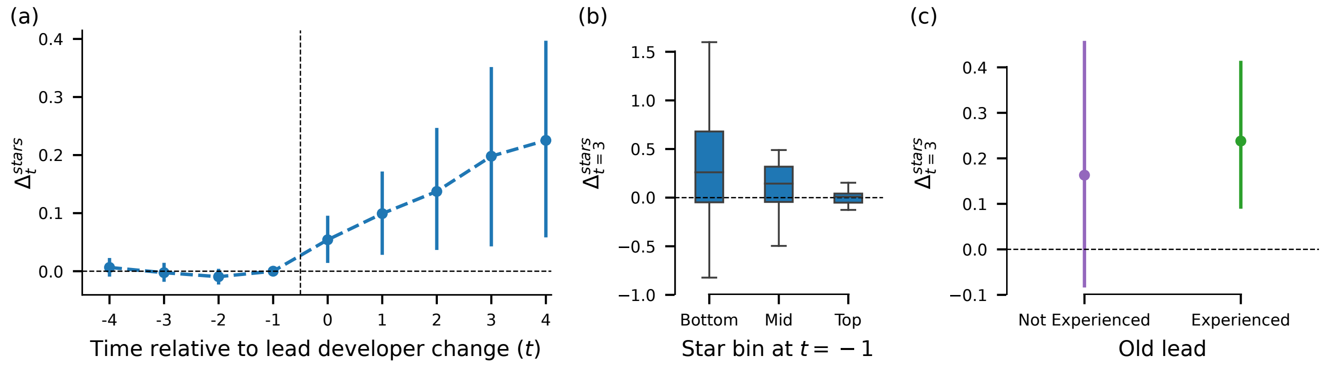

FIG. 4: **Lead developer changes are associated with faster success growth.** (**a**) Average effect of lead developer change Δ_t for stars. Repositories' success grows faster compared to similar repositories that did not undergo such a change. (**b-c**) Success growth Δ_t for stars stratified by (**b**) the success before the change of lead developer and (**c**) the experience of the old lead developer. Worst-performing repositories exhibit a large positive effect following the change, whereas top-performing ones are minimally affected. Repositories started by an experienced lead developer benefit more from the change than those initiated by an inexperienced one. Error bars in (**a**) and (**c**) correspond to 95% confidence intervals.

sustainability highlights a "high risk, high reward" strategy in open-source software development.

More broadly, our study contributes to the ongoing discourse within the field of team science, particularly by addressing the unresolved question of how changes in team composition affect teamwork [31]. Although the literature presents mixed findings, where team changes can be either beneficial [62, 63] or detrimental [64, 65], there is consensus that changes involving core team members deeply affect team functionality [19]. Our results are not only in line with these general predictions, but lean towards a positive effect of such changes on team outcomes, especially for the worst-performing teams before the transition. In addition, the stronger effect observed for experienced lead developers leaves room to speculate about the importance of experienced developers in setting the stage for successful projects regardless of potential changes in such a fundamental role. This observation raises further questions about the characteristics of the old and new lead developers that may influence such a relationship. Drawing on previous studies, the change of a team member is more beneficial the higher the relative skills of the team members involved in the turnover [66]. Further research is needed to identify factors that make the turnover more beneficial for team outcomes.

While our findings relate the change of the lead developer with a significantly higher success growth, the mechanism behind this association remains unclear. Indeed, this relationship highlights a deep connection between the dynamics of teams and their performance, which leaves room for speculation on whether success drives leadership change or vice versa. For instance, as a result of a rapid increase of attention, projects can attract new contributors, and core developers can shift to organizational roles [52]. This suggests that an increase in success brings more developers to join the team, thus increasing the probability of changing the lead developer. A similar mechanism may occur due to Rust's rapid increase in public attention, which likely increases the number of new contributors to projects and may require teams to adapt and undergo structural changes [67]. Conversely, leadership turnover can unleash creative forces previously bound up in an organization or team, which drive the increase in the project's success [68]. At the same time, in our context we saw that new leaders tend to ramp up their activity in a project before taking over, suggesting that there may be value in a balance of previous collaborative ties and new connections in an evolving team [21]. In short, further research is needed to shed light on the causality between lead developer change and success.

Our work relies on the analysis of a curated dataset and benefits from considering the whole development history of one single programming language, providing a controlled environment to characterize team dynamics in open-source software development. Indeed, the inclusion of multiple programming languages may affect the results because coding practices differ depending on the programming language. For instance, it has been shown that different programming languages have varying levels of productivity and require unequal effort to write the same code [69, 70]. Extending our study to a larger sample of programming languages is not trivial. For instance, the success metrics we considered (i.e., GitHub's stars and downloads) are platform-specific and may not be available for all programming languages. This applies in particular to programming languages that are older than software-development platforms, as we may not have data on their repositories' entire development history. Keeping these challenges in mind, future studies on multiple programming languages might broaden our understanding of how teams coordinate in software development projects beyond the Rust ecosystem.

Beyond open-source software development, our work provides a fresh perspective on team evolution, leadership dynamics and their relationship to project success, contributing to a deeper understanding of the successful dynamics of collaborative processes.

IV. METHODS

A. Data and selection of repositories

We used data sourced from [49], consisting of a curated dataset containing the activities of developers across 39 671 repositories hosting Rust packages on various online platforms (e.g., GitHub, GitLab). Developers' activities are tracked through commits, providing a comprehensive record of changes to the project's codebase, in addition to other activities pertaining more to project management tasks (e.g., pull requests, Q&As). Developers' user names are disambiguated, and flags identifying bot accounts are provided. The dataset includes platform-specific features that can be used as a proxy for repositories' success over time. We chose to use the number of stars, which can be considered the most reliable measure of popularity [51], and number of downloads as a proxy for technical quality. Since stars are only available for repositories stored on GitHub, we discarded those hosted in other platforms (6% of the projects in the datasets).

Inspired by [71], we filtered repositories to ensure the inclusion of repositories suitable to study collaborative software development. After discarding the activity of bots, we selected the repositories satisfying the following conditions: (1) first commit with no deletions, (2) total number of lines of code positive across the whole lifetime, (3) first commit in 2014 or later, (4) at least 100 lines of code in total, (5) lifetime of at least one year, (6) at least one commit per month on average, (7) at least one package associated to the repository (since a repository can host more than one package [49]). Conditions (1) and (2) make us more confident that we study repositories for which their whole history is tracked. Indeed, no repository can be initiated by deleting any line, nor can it have a negative number of lines at any point in its lifetime. Conditions (4-7) select repositories hosting software developed over time and likely discard very small projects. After discarding repositories developed by one single developer, and repositories displaying activity on less than four trimesters, we ended up with a total of 6165 repositories.

B. Detecting lead developer changes

The lead developer of a repository is defined as the developer with the highest number of commits. However, different developers may be leading the repository at different points in time. To identify changes in lead developers over time, we counted the cumulative number of commits authored by team members at each time period (i.e., trimesters). Then, we defined the lead developer of the repository at time t as the team member that made the largest share of commits up to that time. To avoid the inclusion of spurious changes, such as those in the initial phases of the project when the activity is relatively low, we excluded the repositories that have changed the lead developer within the first three trimesters of activity. Additionally, cases where a lead developer change is followed by the former lead developer re-assuming their role were excluded as well (total repositories discarded $N_{\text{spurious}} = 206$). This criterion ensures that our analyses rely on repositories in which a meaningful change happened, with the initial lead developer maintaining their position for a significant duration before being succeeded by a new lead developer.

C. Matching procedure

To investigate the performance of repositories after the change of their lead developer ("lead-change repositories"), we compared their success trajectory with that of similar ones whose lead developer did not change ("lead-remain repositories"). Specifically, we implemented a matching procedure that, for each lead-change repository, identifies a set of lead-remain repositories that are similar to the lead-change repository in terms of temporal patterns of activity, team composition and success before the change of lead developer happened. Then, we selected the matched repository among those candidates as the one with the most similar success trajectory during the year preceding the change of lead developer.

The details of the matching are as follows. Initially, for each lead-change repository, we identified lead-remain repositories whose first commit date differs by at most six months from that of the lead-change repository. In addition, we required the lead-remain repositories to have lifetime as long as the age at which the lead-change repository changed the lead developer. This ensures that the lead-change repository and its candidates developed within a comparable timeframe. Such a requirement contributes to controlling for the average status of Rust's ecosystem, thus avoiding potential biases due to the rapid growth of the programming language. In this setting, $t = 0$ designates the time in which the new lead developer takes over. Then, we refined our set of candidates to select repositories that closely matched the lead-change repository in terms of team composition and prior success for $t < 0$. The selected set of candidates met the following criteria: (1) similar team size at $t = -1$ (see Fig. S6), (2) absolute difference in relative effective team size, averaged across $t \in [-4, -1]$, smaller than 0.20, and (3) absolute relative difference of (log) number of stars and downloads at $t = -1$ smaller than 0.50.

As the last step, we selected the candidate that most closely matched in terms of success growth before the change. To do that, we first defined the success growth as $Y_t = S_t/S_{t_0}$, where $t_0 = -1$ refers to the last time period before the change of the lead developer and S_t is either the number of stars or downloads at time t. In other words, we considered the success growth relative to the success of the repository at the last trimester before the lead developer change. We then chose the matched repository as the one that exhibited the smallest maximum relative difference in success growth during the period $t \in [-4, -1]$, considering both stars and downloads. If the maximum difference is larger than 0.50, we discarded the lead-change repository due to the lack of a sufficiently similar repository among the lead-remain ones. Our results are robust against the choice of those thresholds.

Acknowledgements

L.G. and F.B. acknowledge support from the Air Force Office of Scientific Research under award number FA8655-22-1-7025. J.W. acknowledges support from the Center for Collective Learning (101086712-LearnData-HORIZON-WIDERA-2022-TALENTS-01 financed by EUROPEAN RESEARCH EXECUTIVE AGENCY (REA).)

[1] S. Wuchty, B. F. Jones, and B. Uzzi, Science **316**, 1036 (2007), https://www.science.org/doi/pdf/10.1126/science.1136099 .
[2] R. Guimerà, B. Uzzi, J. Spiro, and L. A. N. Amaral, Science **308**, 697 (2005), https://www.science.org/doi/pdf/10.1126/science.1106340 .
[3] L. Wu, D. Wang, and J. A. Evans, Nature **566**, 378 (2019).
[4] M.-L. Liu, N.-T. Liu, C. G. Ding, and C.-P. Lin, Technological Forecasting and Social Change **91**, 295 (2015).
[5] S. M. Ben-Menahem, G. von Krogh, Z. Erden, and A. Schneider, Academy of Management Journal **59**, 1308 (2016), https://doi.org/10.5465/amj.2013.1214 .
[6] J. A. Grand, M. T. Braun, G. Kuljanin, S. W. J. Kozlowski, and G. T. Chao, J Appl Psychol **101**, 1353 (2016).
[7] X.-Y. Xie, W.-L. Wang, and K. Luan, Group Processes & Intergroup Relations **17**, 240 (2014), https://doi.org/10.1177/1368430213502559 .
[8] N. J. A. Thomas A. O'Neill and S. E. Hastings, Human Performance **26**, 236 (2013), https://doi.org/10.1080/08959285.2013.795573 .
[9] F. Baumann, A. Czaplicka, and I. Rahwan, Scientific Reports **14**, 2491 (2024).
[10] Y. Yang, T. Y. Tian, T. K. Woodruff, B. F. Jones, and B. Uzzi, Proceedings of the National Academy of Sciences **119**, e2200841119 (2022), https://www.pnas.org/doi/pdf/10.1073/pnas.2200841119 .
[11] J. B. Bear and A. W. Woolley, Interdisciplinary Science Reviews **36**, 146 (2011), https://doi.org/10.1179/030801811X13013181961473 .
[12] L. Wallrich, V. Opara, M. Wesołowska, D. Barnoth, and S. Yousefi, "The relationship between team diversity and team performance: reconciling promise and reality through a comprehensive meta-analysis registered report," (2024).
[13] M. de Vaan, B. Vedres, and D. Stark, American Journal of Sociology **120**, 1144 (2015), https://doi.org/10.1086/681213 .
[14] N. Pobiedina, J. Neidhardt, M. d. C. Calatrava Moreno, and H. Werthner, in *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13 Companion (Association for Computing Machinery, New York, NY, USA, 2013) p. 1185–1194.
[15] R. B. Freeman and W. Huang, Journal of Labor Economics **33**, S289 (2015), https://doi.org/10.1086/678973 .
[16] S. Hoogendoorn and M. Van Praag, (2012).
[17] A. W. Woolley, C. F. Chabris, A. Pentland, N. Hashmi, and T. W. Malone, Science **330**, 686 (2010), https://www.science.org/doi/pdf/10.1126/science.1193147 .
[18] L. H. Ungar, B. A. Mellers, V. A. Satopaa, P. E. Tetlock, and J. Baron, in *AAAI Fall Symposium: Machine Aggregation of Human Judgment* (2012).
[19] H. Arrow and J. E. Mcgrath, Small Group Research **24**, 334 (1993), https://doi.org/10.1177/1046496493243004 .
[20] T. Saeed, S. Almas, M. Anis-ul Haq, and G. S. K. Niazi, International Journal of Conflict Management **25**, 214 (2014).
[21] B. Vedres and T. Cserpes, in *Academy of Management Proceedings*, Vol. 2020 (Academy of Management Briarcliff Manor, NY 10510, 2020) p. 18879.
[22] S. J. Zaccaro, A. L. Rittman, and M. A. Marks, The Leadership Quarterly **12**, 451 (2001).
[23] S. O'Mahony and F. Ferraro, The Academy of Management Journal **50**, 1079–1106 (2007).
[24] R. Thapa and S. Vidolov, ECIS 2020 Research Papers (2020).
[25] S. TAGGAR, R. HACKEW, and S. SAHA, Personnel Psychology **52**, 899 (1999), https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1744-6570.1999.tb00184.x .
[26] H. A. Simon, *Administrative behavior* (Simon and Schuster, 2013).
[27] H. Arrow, J. McGrath, and J. Berdahl, *Small Groups as Complex Systems: Formation, Coordination, Development, and Adaptation* (Thousand Oaks, California, 2000).
[28] D. J. Deming, The Quarterly Journal of Economics **132**, 1593 (2017).
[29] F. Delice, M. Rousseau, and J. Feitosa, Frontiers in Psychology **10** (2019), 10.3389/fpsyg.2019.01324.
[30] J. C. Gorman, T. A. Dunbar, D. Grimm, and C. L. Gipson, Frontiers in Psychology **8** (2017), 10.3389/fpsyg.2017.01053.

[31] S. E. Humphrey and F. Aime, The Academy of Management Annals **8**, 443 (2014), https://doi.org/10.1080/19416520.2014.904140 .
[32] M. Santolini, L. Blondel, M. J. Palmer, R. N. Ward, R. Jeyaram, K. R. Brink, A. Krishna, and A.-L. Barabasi, "igem: a model system for team science and innovation," (2023), arXiv:2310.19858 [cs.SI] .
[33] S. Chowdhary, L. Gallo, F. Musciotto, and F. Battiston, "Team careers in science: formation, composition and success of persistent collaborations," (2024), in preparation.
[34] A. Zeng, Y. Fan, Z. Di, Y. Wang, and S. Havlin, Nature Human Behaviour **5**, 1314 (2021).
[35] M. Kolbe and M. Boos, Frontiers in Psychology **10** (2019), 10.3389/fpsyg.2019.01478.
[36] R. O. Szabo, S. Chowdhary, D. Deritei, and F. Battiston, Scientific Reports **12**, 10498 (2022).
[37] I. Scholtes, P. Mavrodiev, and F. Schweitzer, Empirical Software Engineering **21**, 642 (2016).
[38] C. Gote, I. Scholtes, and F. Schweitzer, Empirical Software Engineering **26**, 75 (2021).
[39] D. Sornette, T. Maillart, and G. Ghezzi, Plos one **9**, e103023 (2014).
[40] M. Goeminne and T. Mens, in *the Joint Porceedings of the 1st International workshop on Model Driven Software Maintenance and 5th International Workshop on Software Quality and Maintainability* (Citeseer, 2011) pp. 74–82.
[41] K. Yamashita, S. McIntosh, Y. Kamei, A. E. Hassan, and N. Ubayashi, in *Proceedings of the 14th International Workshop on Principles of Software Evolution*, IWPSE 2015 (Association for Computing Machinery, New York, NY, USA, 2015) p. 46–55.
[42] M. Klug and J. P. Bagrow, Royal Society Open Science **3**, 160007 (2016), https://royalsocietypublishing.org/doi/pdf/10.1098/rsos.160007 .
[43] T. Bock, N. Alznauer, M. Joblin, and S. Apel, ACM Trans. Softw. Eng. Methodol. **32** (2023), 10.1145/3593803.
[44] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, in *2013 6th international workshop on cooperative and human aspects of software engineering (chase)* (IEEE, 2013) pp. 49–56.
[45] "Linuk kernel commit distribution on github," https://web.archive.org/web/20240324122230/https://github.com/torvalds/linux/graphs/contributors, accessed: 2024-04-17.
[46] G. Rivlin, "Leader of the free world," https://www.wired.com/2003/11/linus/, accessed: 2024-04-17.
[47] M. Turck, "In conversation with wes mckinney, ceo, ursa computing," https://mattturck.com/mckinney/, accessed: 2024-04-17.
[48] "Pandas commit distribution on github," https://web.archive.org/web/20240417080730/https://github.com/pandas-dev/pandas/graphs/contributors, accessed: 2024-04-17.
[49] W. Schueller, J. Wachs, V. D. P. Servedio, S. Thurner, and V. Loreto, Scientific Data **9**, 703 (2022).
[50] W. Schueller and J. Wachs, Collective Intelligence **3**, 26339137241231912 (2024).
[51] H. Borges and M. Tulio Valente, Journal of Systems and Software **146**, 112 (2018).
[52] D. Maldeniya, C. Budak, L. P. Robert Jr., and D. M. Romero, in *Proceedings of The Web Conference 2020*, WWW '20 (Association for Computing Machinery, New York, NY, USA, 2020) p. 2055–2065.
[53] W. Boeker and R. Karichalil, Academy of Management Journal **45**, 818 (2002), https://doi.org/10.5465/3069314 .
[54] A. Mockus, R. T. Fielding, and J. D. Herbsleb, ACM Transactions on Software Engineering and Methodology **11**, 309–346 (2002).
[55] T. Chełkowski, P. Gloor, and D. Jemielniak, PLOS ONE **11**, 1 (2016).
[56] Y. Ye, K. Nakakoji, and Y. Yamamoto, in *Software Engineering Approaches for Offshore and Outsourced Development*, edited by B. Meyer and M. Joseph (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007) pp. 152–169.
[57] G. Avelino, L. Passos, A. Hora, and M. Valente, in *2016 IEEE 24th International Conference on Program Comprehension (ICPC)* (IEEE Computer Society, Los Alamitos, CA, USA, 2016) pp. 1–10.
[58] R.-H. Pfeiffer, in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)* (2021) pp. 41–45.
[59] L. Williams and R. R. Kessler, *Pair programming illuminated* (Addison-Wesley Professional, 2003).
[60] M. Bowler, "Truck Factor," https://web.archive.org/web/20210429072452/http://www.agileadvice.com/2005/05/15/agilemanagement/truck-factor/ (2005), [Accessed 27-03-2024].
[61] M. Ferreira, T. Mombach, M. T. Valente, and K. Ferreira, Software Quality Journal **27**, 1583 (2019).
[62] J. C. Gorman, P. G. Amazeen, and N. J. Cooke, Nonlinear dynamics, psychology, and life sciences **14**, 265 (2010).
[63] H.-S. Choi and L. Thompson, Organizational Behavior and Human Decision Processes **98**, 121 (2005).
[64] J. N. Pedro J. Ramos-Villagrasa and A. L. García-Izquierdo, European Journal of Work and Organizational Psychology **21**, 778 (2012), https://doi.org/10.1080/1359432X.2012.669525 .
[65] K. Lewis, M. Belliveau, B. Herndon, and J. Keller, Organizational Behavior and Human Decision Processes **103**, 159 (2007).
[66] J. K. Summers, S. E. Humphrey, and G. R. Ferris, Academy of Management Journal **55**, 314 (2012), https://doi.org/10.5465/amj.2010.0175 .
[67] M. Galesic, D. Barkoczi, A. M. Berdahl, D. Biro, G. Carbone, I. Giannoccaro, R. L. Goldstone, C. Gonzalez, A. Kandler, A. B. Kao, R. Kendal, M. Kline, E. Lee, G. F. Massari, A. Mesoudi, H. Olsson, N. Pescetelli, S. J. Sloman, P. E. Smaldino, and D. L. Stein, Journal of The Royal Society Interface **20**, 20220736 (2023), https://royalsocietypublishing.org/doi/pdf/10.1098/rsif.2022.0736 .
[68] D. Tzabbar and R. R. Kehoe, Journal of Management **40**, 449 (2014).
[69] L. Lavazza, S. Morasca, and D. Tosi, in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16 (Association for Computing Machinery, New York, NY, USA, 2016) p. 1434–1439.
[70] L. Prechelt, Computer **33**, 23–29 (2000).
[71] C. Gote, P. Mavrodiev, F. Schweitzer, and I. Scholtes, in *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22 (Association for Computing Machinery, New York, NY, USA, 2022) p. 262–273.

Supplementary Material: The dynamics of leadership and success in software development teams

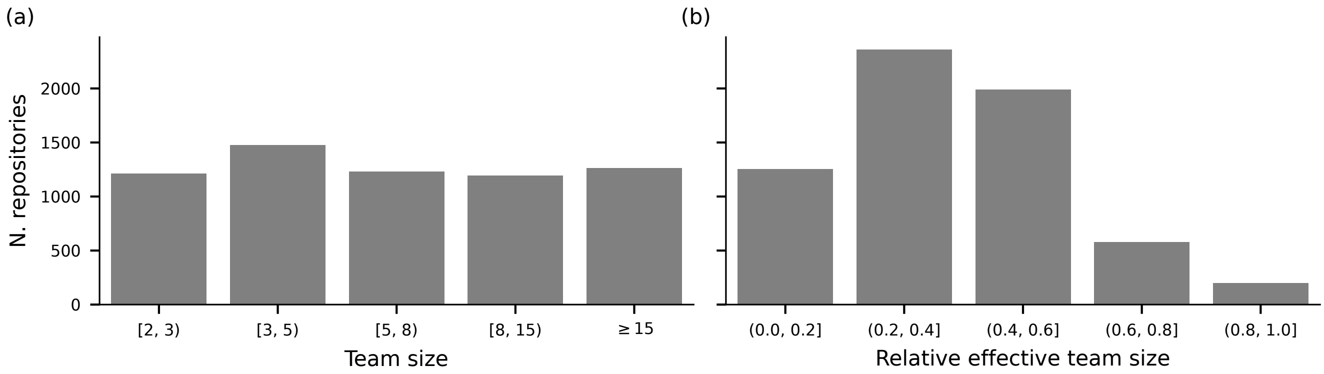

FIG. S1: Distribution of repositories across team size (**a**) and relative effective team size strata (**b**), as defined in Fig. 1.

TABLE I: Spearman's correlation coefficient between the relative effective team size of teams and the success of repositories (stars and downloads) stratified by team size. Correlations are estimated at various stages of the repositories' lifetimes: the last active period in the dataset (All) and after one, two, and three years of activity. Significance levels are denoted as follows: * for $p < 0.05$, ** for $p < 0.01$, and *** for $p < 0.001$.

			Spearman's ρ	
			Stars	Downloads
Success at	Team size	N. repos.		
All	2	1212	-0.28***	-0.16***
	3-4	1475	-0.37***	-0.18***
	5-7	1229	-0.41***	-0.18***
	8-14	1192	-0.41***	-0.11***
	15+	1263	-0.54***	-0.16***
1	2	1563	-0.27***	-0.10***
	3-4	1690	-0.37***	-0.17***
	5-7	952	-0.42***	-0.14***
	8-14	690	-0.42***	-0.12**
	15+	372	-0.49***	-0.15**
2	2	1354	-0.26***	-0.11***
	3-4	1630	-0.37***	-0.14***
	5-7	1220	-0.42***	-0.14***
	8-14	949	-0.44***	-0.06
	15+	728	-0.48***	-0.13***
3	2	929	-0.27***	-0.10**
	3-4	1229	-0.35***	-0.15***
	5-7	1009	-0.41***	-0.14***
	8-14	889	-0.40***	-0.06
	15+	834	-0.48***	-0.13***

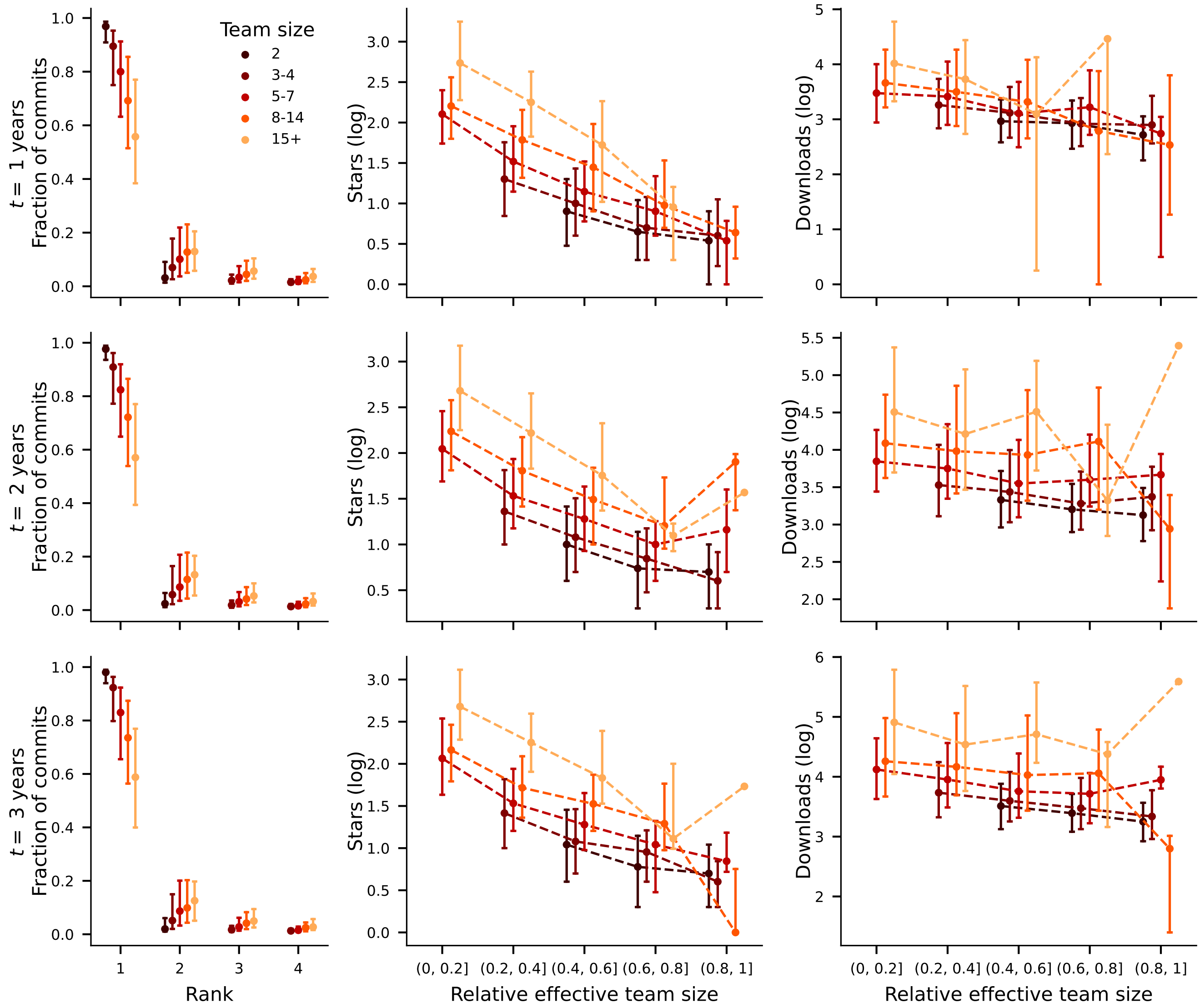

FIG. S2: Workload distribution within teams and relationship with success at different point in time during the lifetime of repositories. Similarly to Fig. 1, the workload distribution is heterogeneous and the success is decreasing in the relative effective team size, pointing to those properties being stable across the lifetime of repositories. The Spearman's correlations between relative effective team size and success remains negative and significant for both stars and downloads. The only exceptions are downloads for team sizes 8-14 at $t = 2$ and $t = 3$ years, cases in which the correlation is not anymore significant at 0.05 level. Table I shows the detailed results of the correlation. The details of the figure are the same as in Fig. 1.

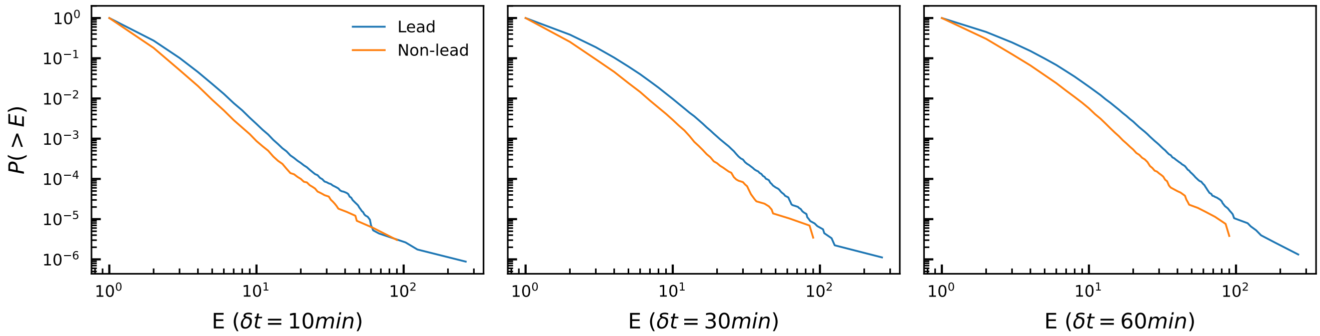

FIG. S3: Complementary cumulative distribution of the number of commits close in time E for different values of δt. Leaders are more likely to make commits in larger event train sizes for all the tested values of δt.

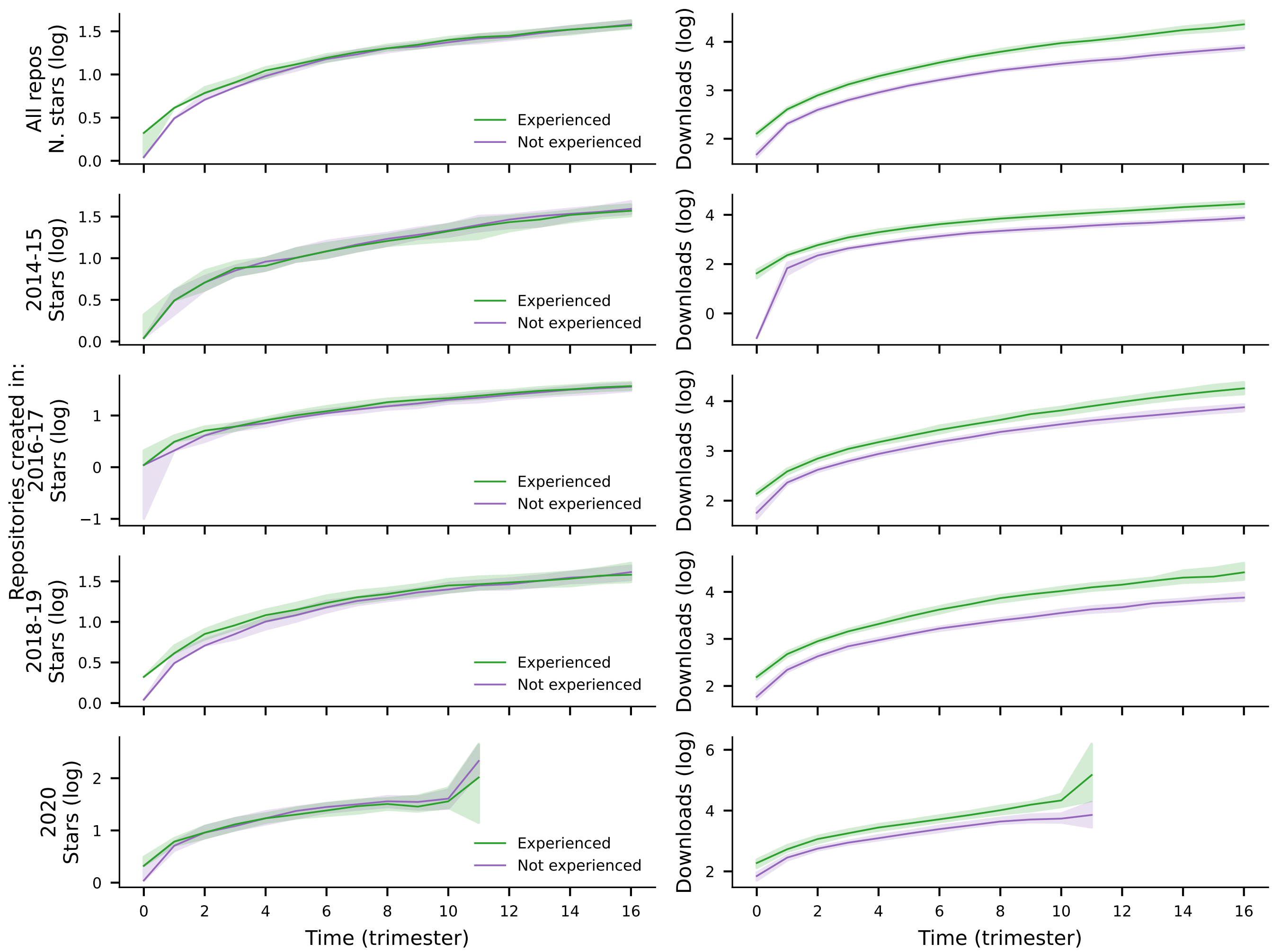

FIG. S4: Number of downloads acquired across repositories' lifetime stratified by lead developer's experience (median and 95% confidence interval). Each row corresponds to repositories initiated in different years, with the first row including all repositories. While the prior experience of lead developers doesn't impact repositories' success in terms of stars, it does influence the number of downloads. Indeed, repositories whose lead developer is experienced receive more downloads across their lifetime than those whose lead developer is not experienced. After accounting for the initiation year of repositories, we find that this observation remains consistent regardless of the repositories' age.

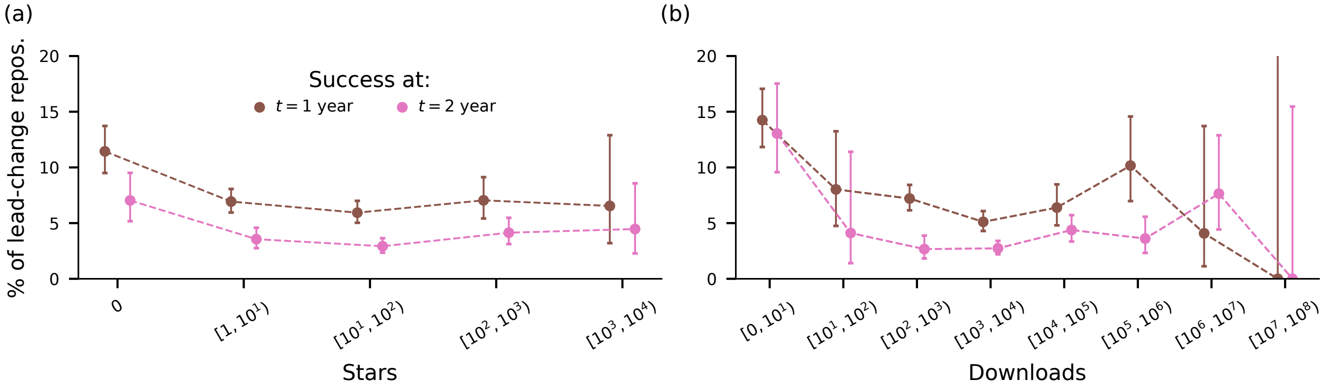

FIG. S5: Percentages of repositories that will change the lead developer after time t as a function of the success of repositories at time t. The figure suggests that repositories with the worst performance in terms of stars (**a**) and downloads (**b**) are the most likely to change their lead developer. We investigated whether there is a connection between future leadership changes and past success, testing both linear and quadratic dependencies using a fixed effect regression model. However, our analysis did not find significant evidence supporting such a relationship. Error bars refer to 95% confidence intervals (Wilson score interval).

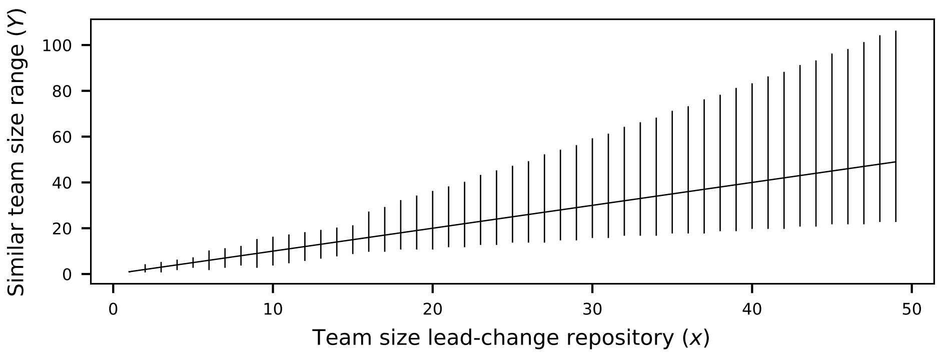

FIG. S6: Graphical representation of the team size similarity used in the matching approach. For each team size of the lead-change repository (horizontal axis), the range of similar team sizes for the candidate repositories is represented (vertical lines). The solid line represents the two team sizes being equal. In plain terms, we require similar team sizes to be strictly close or equal for small team sizes (2 to 15), while being similar in relative terms for larger team sizes (larger than 15). The functional relationship is the following, considering x being the team size of the lead change repository and Y being the range of similar team sizes for the candidate repositories: $Y = \{1\}$ if $x = 1$, $Y = \{n | n \in \mathbb{N} \text{ and } |n - x| \leq 2\}$ if $x \in [2, 5]$, $Y = \{n | n \in \mathbb{N} \text{ and } |n - x| \leq 4\}$ if $x \in [6, 8]$, $Y = \{n | n \in \mathbb{N} \text{ and } |n - x| \leq 6\}$ if $x \in [6, 15]$, $Y = \{n | n \in \mathbb{N} \text{ and } |\log_{10}(n) - \log_{10}(x)| / \log_{10}(x) \leq 0.2\}$ if $x > 15$.

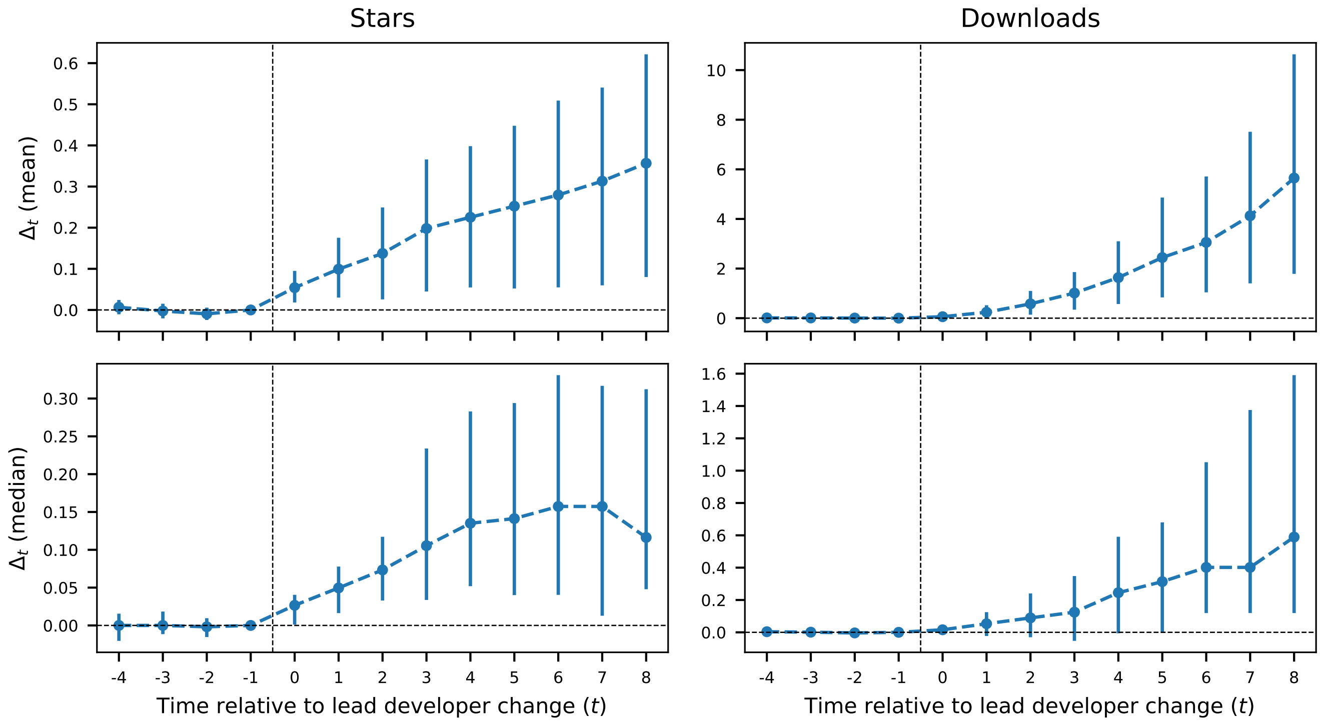

FIG. S7: We checked if the result for the average Δ_t (top row) is influenced by the skewness of its distribution by comparing it with the estimation of the median Δ_t (bottom row). The effect on stars (left column) and downloads (right column) remains positive, although the magnitude decreases, especially for downloads. This suggests that while the skewness of Δ_t may enhance the average Δ_t, it doesn't affect the direction of the effect. Error bars represent 95% confidence intervals.

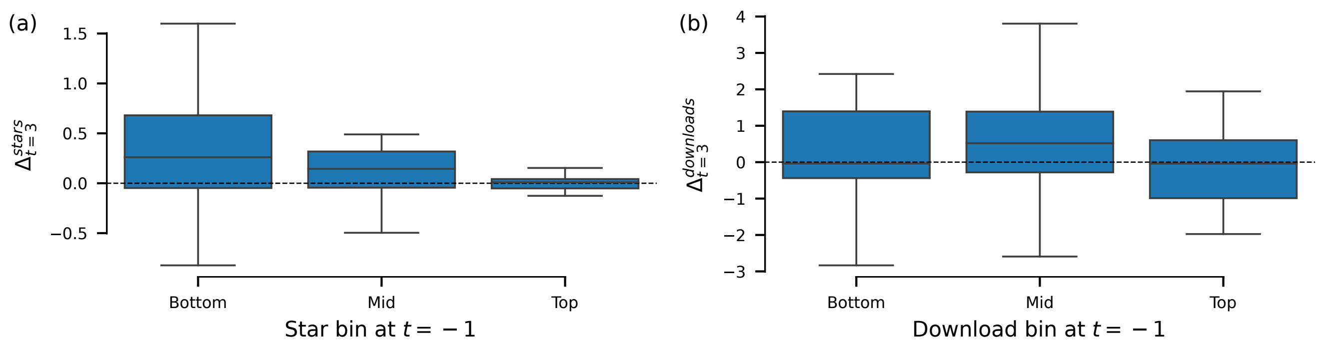

FIG. S8: Descriptive analysis of the heterogeneity of Δ_t considering the success before the change of the lead developer. Repositories are divided into bins depending on the number of stars (**a**) and number of downloads (**b**) in the trimester preceding the change of the lead developer ($t=-1$). Bottom teams: below the 30th percentile of success; Mid teams: between the 50th and 80th percentile of success; Top teams: above the 90th percentile of success. Although worst performing repositories in terms of stars benefit more from the change of lead developer, this is not the same when we consider downloads. Indeed, we observe no difference on the effect depending on the number of downloads before the change of the lead developer.

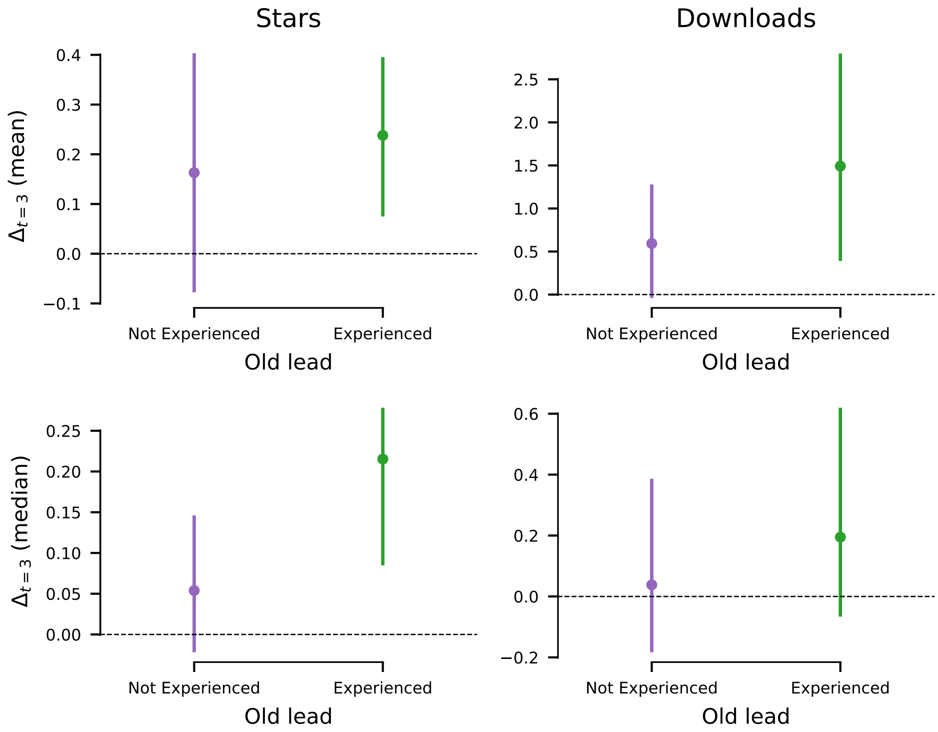

FIG. S9: Descriptive analysis of the heterogeneity of Δ_t at $t = 3$ considering the experience of the old lead developer. Repositories initiated by experienced lead developers have a larger benefit from the turnover. The second row displays the median Δ_t, indicating that the observed result is not affected by the skewness of the distribution. Error bars refer to 95% confidence intervals.

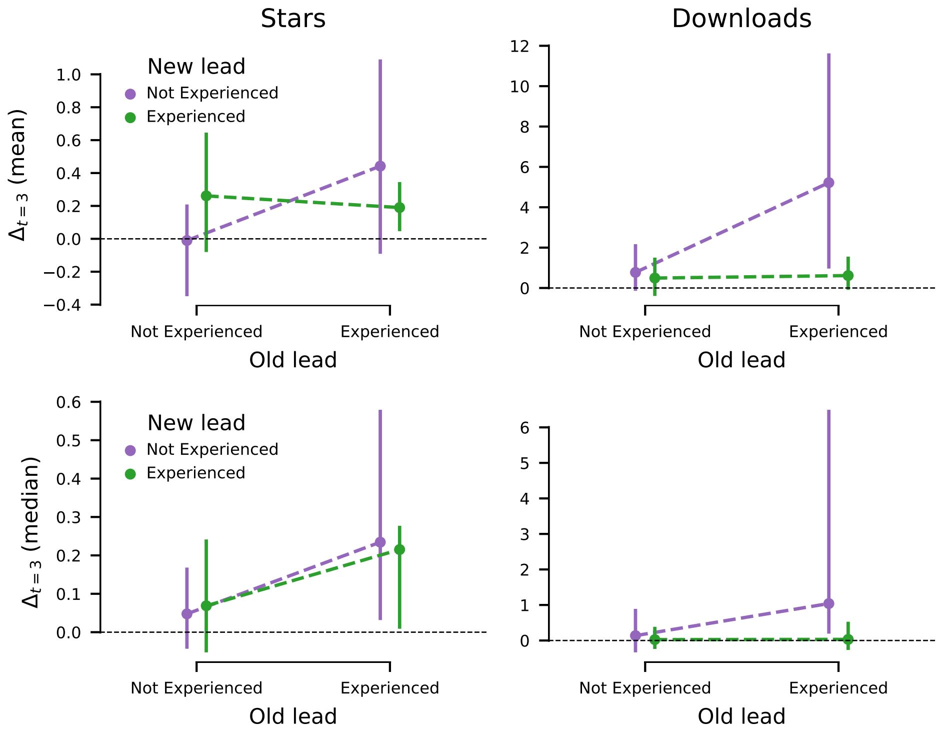

FIG. S10: Descriptive analysis of the heterogeneity of Δ_t considering the experience of the old and new lead developer. The figure suggests that the experience of the new lead developer is not affecting strongly the way repositories benefit from the change. However, this further stratification shrinks the sample size and the large error bars make it difficult to draw conclusions from this analysis. The second row displays the median Δ_t. Error bars refer to 95% confidence intervals.