Off to a Good Start: Dynamic Contribution Patterns and Technical Success in an OSS Newcomer's Early Career

Yang Yue, Yi Wang, and David Redmiles, *Member, IEEE*

Abstract—Attracting and retaining newcomers are critical aspects for OSS projects, as such projects rely on newcomers' sustainable contributions. Considerable effort has been made to help newcomers by identifying and overcoming the barriers during the onboarding process. However, most newcomers eventually fail and drop out of their projects even after successful onboarding. Meanwhile, it has been long known that individuals' early career stages profoundly impact their long-term career success. However, newcomers' early careers are less investigated in SE research. In this paper, we sought to develop an empirical understanding of the relationships between newcomers' dynamic contribution patterns in their early careers and their technical success. To achieve this goal, we compiled a dataset of newcomers' contribution data from 54 large OSS projects under three different ecosystems and analyzed it with time series analysis and other statistical analysis techniques. Our analyses yield rich findings. The correlations between several contribution patterns and technical success were identified. In general, being consistent and persistent in newcomers' early careers is positively associated with their technical success. While these correlations generally hold in all three ecosystems, we observed some differences in detailed contribution patterns correlated with technical success across ecosystems. In addition, we performed a case study to investigate whether another type of contributions, i.e., documentation contribution, could potentially have positive correlations with newcomers' technical success. We discussed the implications and summarized practical recommendations to OSS newcomers. The insights gained from this work demonstrated the necessity of extending the focus of research and practice to newcomers' early careers and hence shed light on future research in this direction.

Index Terms—Dynamic contribution pattern, early career, newcomer, open source, technical success

1 Introduction

Open-source software (OSS) development has been a prevalent practice to build software products nowadays [1]. One of the critical success factors for OSS projects is attracting and retaining newcomers [2]. When newcomers decide to join an OSS project, they may face various barriers, including technical barriers [3], as well as social barriers [4]. Researchers have proposed theories and techniques to help newcomers to identify and overcome these barriers [5], [6]. However, successful onboarding (in terms of making the first code contribution [7]) does not necessarily guarantee newcomers' survival and advancement in their newly-joined projects. Extant literature has shown that most newcomers, though successfully onboarded, eventually dropped out in a short period [8]. In other words, many newcomers fail in their "early careers."

Literature in many disciplines has shown that individuals' early careers are critical to their long-term career success [9], [10], [11]. Particularly, supporting newcomers' early careers brings many benefits to both OSS projects and newcomers themselves. For an OSS project, good support to newcomers' early career would help retain its most valuable asset, and motivate newcomers to make sustainable contributions, hence reducing the high turnover problem [12]. For newcomers, having support during their early career helps them better migrate from the periphery to the core in the onion model of OSS project teams [13]. Moreover, supporting their early career could avoid newcomers' early dropout from the projects, then increases their chances of developing expertise in the technologies used by their projects.

However, the state-of-the-art SE research into OSS newcomers mainly focuses on helping newcomers overcome barriers in their onboarding [6], thus has not yet fully recognize the challenges of their early career after successful onboarding. To provide effective support to newcomers' early career, the first yet most crucial step should be developing an understanding of newcomers' activities in their early careers. We would need to identify which types of behaviors would benefit their success, and which might be detrimental. Moreover, investigating such behaviors requires a dynamic perspective rather than static snapshots, because an individual's career is a developmental process, and has been proven to be determined by dynamics of behaviors over an extended period by vocational

- *Yang Yue and David Redmiles are with the Department of Informatics, University of California, Irvine, CA 92697 USA. E-mail: y.yue@uci.edu, redmiles@ics.uci.edu.*
- *Yi Wang is with the School of Computer Science (National Pilot Software Engineering School) and Key Laboratory of Trustworthy Distributed Computing and Service (MoE), Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: wang@cocolabs.org.*

Manuscript received 25 February 2021; revised 1 February 2022; accepted 15 February 2022. Date of publication 3 March 2022; date of current version 13 February 2023.
The work of Yang Yue and David Redmiles was supported in part by the Donald Bren School of Information and Computer Sciences, University of California, Irvine. The work of Yi Wang was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62076232 and 62172049, in part by the Fundamental Research Funds for the Central Universities, and in part by the Beijing University Posts and Telecommunications. (Corresponding author: Yi Wang.)
Recommended for acceptance by A. M. Moreno.
Digital Object Identifier no. 10.1109/TSE.2022.3156071

behavior researchers [14], [15], [16]. Without such an understanding, researchers could not develop effective mechanism designs and technical interventions to encourage early career behaviors that have long-term benefits and discourage harmful early career behaviors.

In this paper, we sought to develop such an understanding by analyzing dynamic contribution behaviors in newcomers' early careers, and correlating these patterns with their technical success (quantitatively measured by one's "eigenvector centrality" [17] at the source code file level). Thus, we have our first research question:

RQ$_1$ *Which dynamic contribution patterns in newcomers' early career would be positively or negatively associated with their technical success in newly joined projects?*

Moreover, OSS development has been increasingly adopting the ecosystem paradigm, which is a collection of software projects that have some given degree of symbiotic relationships [18]. Different ecosystems may have varying norms and processes [19], [20], which could introduce variances to the dynamic contribution patterns identified in **RQ$_1$**. Hence it is necessary to investigate if there are ecosystem-specific correlations between contribution patterns and technical success. So we ask the second research question:

RQ$_2$ *Are there any differences regarding dynamic contribution patterns across different OSS ecosystems?*

To answer the above research questions, we compiled a dataset consisting of individual members' early career contributions with fine-grained data from 54 large and popular OSS projects in three OSS ecosystems, Apache, OpenStack, and Python Data Science. The rationale to choose these ecosystems is that they are popular ecosystems in open source community that attract lots of contributors, which could provide a diverse sample of newcomers to study. Moreover, many projects in these ecosystems tend to have long development lifecycles, thus, enabled us to study the dynamics of newcomers' early career in a longitudinal fashion. Our findings confirmed that some specific contribution patterns were correlated with newcomers' technical success. In general, these early career contribution patterns included maintaining consistency and persistence, as well as avoiding prolonged inactivity between code contributions. Also, these patterns might have some slight variances that were contingent on different ecosystems.

Indeed, a newcomer's contribution could be beyond technical ones represented by contributing code. Another category of typical contributions is performing documentation tasks, which has been identified as a potential starting point in the pathway towards a successful career in an OSS project [21], [22]. Hence, our third research question aims to explore newcomers' early career documentation contribution and their technical success:

RQ$_3$ *Are newcomers' documentation contributions helpful in achieving technical success?*

Thus, we conducted a complementary case study with six projects selected from the original sample of 54 projects (two projects were selected from each ecosystem, based on the number and percentage of documentation files in repository). The results showed that only a very small set of newcomers made substantial contributions to both documentation and source code. Thus, we could not find sufficient direct evidence to support the positive effects of documentation contributions in newcomers' early career technical success. However, it seems that making code and documentation contributions intensively could help newcomers' technical success; and, in natural settings, it is an unlikely occurrence that a newcomer experiences the role shift from "documentation contributor" to "code contributor." Meanwhile, compared with documentation contributions associated with code contributions, documentation contributions unrelated to code contributions seem to be correlated more likely with technical success measured by the centrality in one's early career. Further research would be necessary to investigate such correlations.

To the best of our knowledge, our work is the first empirical study focusing on newcomers' dynamic contribution patterns in their early careers beyond their onboarding. It extends the prior work that mostly focused on newcomers' careers before they make their first accepted contribution. Specifically, our work makes the following four major contributions:

- A fresh perspective focusing on the dynamic patterns of newcomers' contributions to OSS projects during their early career beyond onboarding processes.
- An empirical study that identifies the correlations between the dynamic patterns of newcomers' contributions and their technical success in terms of code centrality, with consideration of ecosystems.
- A case study that reveals several dynamic trajectories of code and documentation contributions, and some potential correlations with technical success in newcomers' early career success, though not explicitly establishing these relationships due to lack of sufficient data.
- A set of practical guidelines for newcomers on how to adjust their early career contribution behaviors to increase the chance of achieving technical success in OSS projects.

The rest of this paper proceeds as follows: Section 2 briefly introduces the related work, focusing on research into supporting newcomers, as well as behavioral analysis towards OSS newcomers. Section 3 presents our research methods, including projects sampling, data collection, and preparation. Section 4 introduces the analysis strategies we used to find the answers to **RQ$_1$** and **RQ$_2$**. Section 5 presents the results that answer these two **RQs**, and Section 6 presents the case study on documentation contribution, answering **RQ$_3$**. Section 7 discusses implications, limitations, and future work. Section 8 concludes the article.

2 Related Work

2.1 The Common Pathway From a Newcomer to a Fully-Integrated Member

In general, it takes several phases for developers to go through the process from a newcomer to a fully-integrated project member. Fig. 1 describes these phases. At the very beginning, a newcomer needs to decide which project to join. Many intrinsic and extrinsic factors may impact such a decision [23], [24], [25], [26], [27]. Once making up their mind, the onboarding phase starts, and it is a non-trivial phase. A newcomer may face various barriers in this phase [3], [4], [28], related to code issues, project process, etc. The

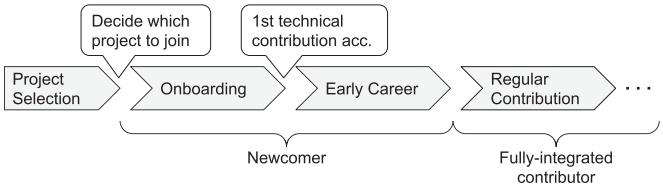

Fig. 1. Key phases and milestones in an OSS developer's career in a project.

tensions on how to start the contribution exist in the literature, e.g., one suggestion is that newcomers could start with documentation [22], while other studies indicate that the newcomers who start with documentation tend to only contribute to documentation, and they never touch source code [21]. Usually, successfully making the first accepted technical contribution concludes this phase. Then, a newcomer begins the early career phase. In this phase, a newcomer continues making contributions; meanwhile, getting familiar with the project, and being known by other members still take a substantial amount of their time. In this phase, a newcomer's potential often may not be fully realized because developing project fluency still takes time [29]. Usually, this phase lasts for several months for most newcomers [29]. However, there is no specific event signifying the finish of this phase. After this phase, a newcomer usually will achieve a similar level of productivity with most of the other regular contributors, and moreover, behave like a regular contributor. Thus, they should be viewed as fully integrated contributors. Of course, in their later careers, a few of them may eventually advance to core contributors, while some others may leave the project.

2.2 Supporting OSS Newcomers

2.2.1 Barriers Faced by OSS Newcomers

Researchers have identified various barriers a newcomer may face when onboarding an OSS project [28]. Such barriers can be divided into three categories: technical barriers [28], social barriers [4], and process barriers [30]. The technical barriers could be classified into six categories, i.e., issues to build/set up a workspace, code issues, problems with documentation, newcomer behaviors, newcomer technical knowledge, and finding a way to start [3]. Social barriers are mainly related to social interaction issues [3] and include issues such as reception, newcomers' communication, finding a mentor, and cultural differences [4]. Moreover, newcomers also face some barriers related to process [30], e.g., long project processes, lack of knowledge about procedures and conventions. While most of these barriers happen during their onboarding process [31], there are also several barriers to prohibit newcomers from long-term contribution and success even after successful onboarding. Their willingness and the climate of the project might be the primary reasons for being long-term contributors [32] or abandoning the project [33]. For example, newcomers often tend to receive more negative feedback [34]. Researchers also found that some groups of newcomers may encounter some group-specific barriers. Female newcomers often experience gender biases in OSS projects [35], [36], and they tend to have lower self-efficacy than male newcomers [30], [37]. What's more, the systematic negative biases towards females widely exist in many aspects of OSS, which discourage female newcomers [38].

2.2.2 Overcoming These Barriers

Various methods, guidelines, and tools have been proposed to help newcomers overcome these barriers. Technical training is a common method to address the technical barriers [30], [31], which helps newcomers get the necessary expertise and accelerate their onboarding [39], [40]. Assigning mentors to newcomers is also a common practice [40]. Newcomers could learn from their mentors about how to learn and develop knowledge structures in the projects [41]. The projects could ease onboarding efforts by providing clear and concise project documentation [30]. Such "how to start" documentation is pretty valuable for newcomers, and the whole community [42]. The project should also encourage newcomers to get involved in interactions with other members [41], which has been proven to be beneficial, regardless of gender [36]. Regarding the code commitment, division of tasks could help newcomers start from easy tasks [41], or those of their interests [30].

Researchers also built computational tools for newcomers' onboarding. Recommenders connecting newcomers with projects fitting their experiences and potential mentors were designed and evaluated [43], [44]. Web portals serving multiple purposes were also proposed. Steinmacher *et al.* [45] designed such a portal, named FLOSScoach, to address a number of technical, social, or process barriers. Researchers also tried gamification, to orient and motivate newcomers [46], [47]. Regarding the contribution tasks, BugExchange could provide newcomers tasks, improve newcomers' learning experience, reduce dropouts, and foster community building, so that newcomers could familiarize themselves with both technical and social aspects of OSS projects [48]. If newcomers already get familiar with projects, they could choose the bug-fixing tasks based on their interests as the start point [49].

2.3 Analyzing OSS Newcomers' Activities

Investigating developers' activities could yield rich SE insights, including those related to newcomers. Meanwhile, the popularity of collaborative software development platforms such as GitHub makes developers' activities in OSS projects readily accessible. Researchers have used traditional and open data sources to study newcomers' activities. We briefly summarize some representative work.

2.3.1 Selecting Projects

For any OSS contributors, their first activity would be selecting a project to join. Project selection activities are decision-making processes involving many factors, to name a few, contributors' own motivations [23], [24], project characteristics [25], [26], and social/technical signals [27]. Some factors' impacts may reach later phases far beyond the onboarding. Prior experience and social connection could also influence a contributor's productivity [23], and motivation would also influence a contributor's long-term career [41], [50]. Anyway, getting off to a good start is half of success.

2.3.2 Social Activities

Social activities, such as communication and social interaction, are prevalent throughout a contributor's tenure. For a

newcomer joining the project, they would perform a set of social activities, e.g., communicating through the mailing list and expressing the interests. They are summarized into "joining script" [51], and researchers found that the social activities during the onboarding process could be slightly different due to the diverse characteristics of newcomers, for example, student vs. non-student [52]. It is not surprising that social interactions could help newcomers get familiar with their projects and facilitate newcomers' evaluation of the fit between themselves and projects [8]. Since many of such social interactions are transparent to project members, these social activities could signal rich information about contributors, projects, and communities [53]. Thus, helping newcomers understand the consequences of their social activities may need further attention.

2.3.3 Technical Activities

Technical activities are major activities in OSS projects. Steinmacher *et al.* [54] analyzed newcomer's first contribution to the project and found that the initial contribution is not necessarily a code contribution. Newcomers may start from tasks such as translation, reporting bugs, documentation, and so on. Regarding the code contribution, Gousios *et al.* studied the developers' activities in the pull request based contribution solicitation model [55]. They concluded that such a model is often not newcomer-friendly. Contributors' social and technical activities are often mixed with and mutually impact each other. Sarker *et al.* revealed that different activities could influence each other and affect contributors' performance [56]. Calefato *et al.* found the activities tend to be more organized, more dutiful, and more cooperative, along with the involvement [57]. Role migration, social learning, and pursuing self-interest could be the main factors causing such changes in activities [58], [59], [60].

2.4 Early Career

The importance of the early career in determining both the short-term and long-term career outcome has been consistently confirmed by numerous studies in many occupations, mostly in the areas of management and vocational behavior. It is fair to conclude that individuals' career paths are shaped by both their behaviors and many other personal and environmental (e.g., interpersonal, institutional, cultural, etc.) factors during their early careers, which outcome to a significant degree [11] in many professions, to name a few, university faculty members, school teachers, business professionals, engineers. For instance, in a study with 193 newly-admitted doctoral students, Bauer & Green found that the involvement in socialization in the early phase (9 months) of their doctoral career would be strongly associated with their success in terms of academic productivity [61]. In another study of 247 employees in their first six months after joining a company in France, researchers found that their proactive behaviors in early career, i.e., information seeking and socialization, positively influenced the success in terms of integration and job satisfaction, moderated by their tendencies on servant leadership [62].

While the early career might be a relatively short period in one's entire career, it still cannot be viewed as a static snapshot. There might be quite diverse behavioral dynamics over the course of a specific period over the course of one's career in organizational settings [63], [64], [65]. Such dynamics are often valuable in understanding and predicting many phenomena [66]. Think about a frequently occurring scenario in open source development: a developer might contribute a lot in the first several days but then leave the project suddenly. In this scenario, if we only look at the number of files that developers contribute in total, we may not be able to recognize the potential problem in the project's sustainability caused by that developers' abrupt turnover. The above scenario grounds the necessity of taking dynamic, time-dependent perspectives in investigating one's early career. As Shipp & Cole [67] pointed out in their literature review: "*t*ime is an integral part of the human experience, particularly at work" and thus, the adoption of a temporal lens is essential for the advancement of organizational and vocational behaviors [68], including studying open source developers' early careers.

2.5 Our Contributions

The above literature review shows that most extant work on OSS newcomers has focused on the newcomers' onboarding process ended at the success of the first technical contribution [6], [54]. However, successful onboarding does not equal a successful career. The next phase, i.e., a newcomer's early career (see Fig. 1), could also be critical but less investigated. The primary purpose of our work is to fill this by shifting the focus to newcomers' early careers (a few months after the first commit). In addition, a newcomer's early career is a dynamic process. Thus, it has to be studied from a dynamic perspective with fine-grained activity data. Our research design, which is going to be introduced in the next section, serves this purpose precisely. Besides technical contribution, we also explored other types of contribution, i.e., documentation contribution, to gain more understanding of newcomers' early careers.

3 Research Methods

3.1 Sampled Projects

We first needed to draw a sample of projects for the study. We selected projects under three popular OSS ecosystems, Apache, OpenStack, and Python Data Science. Apache is a collection of various open source projects used as web servers, cloud platforms, and data management, following the same project governing practices. OpenStack is an open-source cloud computing platform developed with Python. Python Data Science ecosystem contains many tools and libraries to help data scientists work efficiently in machine learning and analytic tasks. These three ecosystems represent a wide range of tech stacks. For example, many programming languages are used in these ecosystems, including Java, Python, C/C++, Scala, etc. They crosscut multiple domains, including internet infrastructures, cloud computing, scientific computing, data science, and machine learning, which brings a diverse population of developers. Then all of them have over three thousands projects using GitHub as the major platform for repository hosting and project collaboration, thus making it possible to collect data under the same ontological schema of defining "contributions" of a developer. Moreover, many projects in these three ecosystems are large

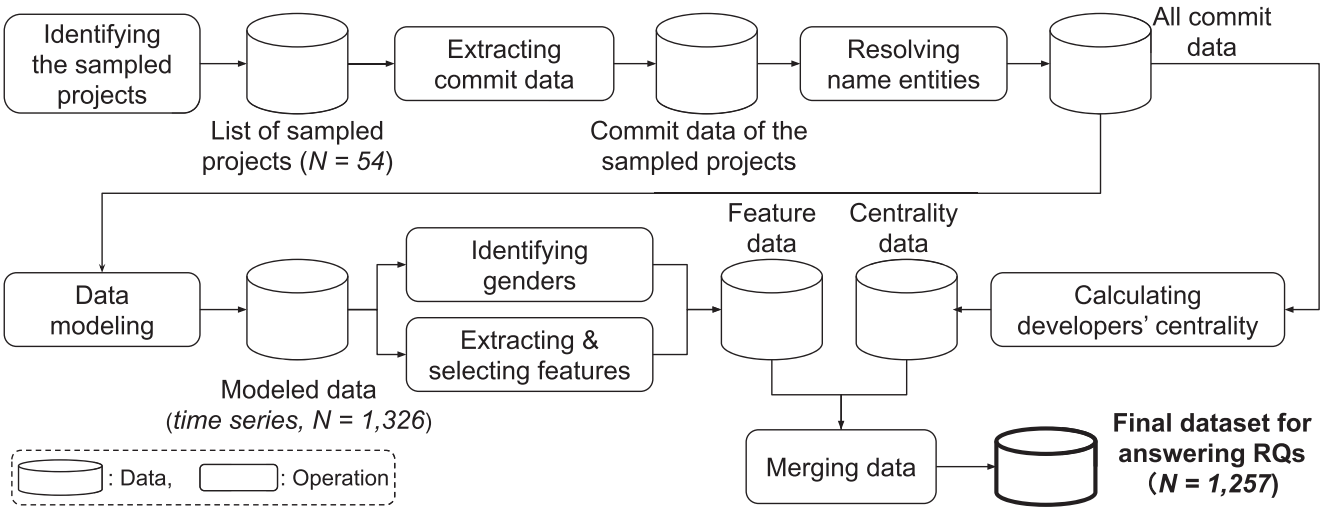

Fig. 2. Overview of the data collection and preparation process.

in size and have a long development lifecycle. Such characteristics enable us to study the dynamic of newcomers' early careers in a longitudinal fashion while keeping the reliability of the data analyses. Based on the above considerations, we chose these three ecosystems and selected 20 projects from Apache, 20 projects from OpenStack, and 14 projects from Python Data Science. These projects were selected based on the popularity (number of stars), as they could attract more developers. Note that we selected fewer projects from Python Data Science ecosystem. Such a decision was not randomly made. The rationale is that the other projects were not comparable with the selected ones, no matter the popularity (the number of stars) or community size (the number of contributors). Thus, we only chose 14 projects, making the total sample contain 54 OSS projects. On average, each selected project has been established for 7 years, with 7,778 stars, has 616 contributors, 13,871 commits, and 2,843 files in repository.

3.2 Data Collection and Preparation

Once selected the sampled projects, the next step was collecting and preparing data for the analysis. Fig. 2 provides an overview of the data collection and preparation process.

3.2.1 Extracting Commit Data

We first collected all developers' commit data from the sampled projects' repositories. These commit data would be used to derive their contribution behaviors. We used PyDriller [69] to extract the details of code commits, including authors, timestamp, SHA, list of source code files in the commit, and the modification type defined by PyDriller. PyDriller defines six types of modifications at the source code file level, including ADD, COPY, RENAME, DELETE, MODIFY, and UNKNOWN. It is relatively straightforward to understand what they stand for. Note that MODIFY refers to a type of composite modification, which means adding and deleting some code in the same source code file.

Since we were interested in developer's code contributions, the data extraction focused on the commits containing source code files, determined by their filename extensions, i.e., `.java`, `.py`, `.cpp`, `.c`, `.h`, and `.cc`. Regarding other files in repositories, such as `.xml` and `.yaml`, they are mainly configuration files that are not frequently touched by developers, thus, these files were not included in our dataset. In this paper, commits without any source code files were discarded.

3.2.2 Resolving Name Entity

Developers may use multiple names when submitting commits, for example, full name, email, or login name. To faithfully extract a developer's complete contribution behavior traces in a project, these different names must be resolved first. For each developer, we used GitHub API to retrieve the full name, login name, and email, if available. Then we used the login name as the identifier to resolve these name entities in the collected data. When the login name was unavailable, the full name was treated as the identifier. Thus, we could compile an individual developer's full contribution history during his or her early career in a project.

Besides, some projects might use bots to automate development processes [70]. We excluded these bot accounts through a two-step procedure. We first searched the accounts containing words "build," "exporter," or "bot," in their login names, full names, or emails [56], then manually checked them and removed the bot accounts according to their GitHub profile descriptions. Furthermore, we manually checked the remaining 1,326 developers in our dataset and did not find other existing bots.

3.2.3 Data Modeling

Since we focus on dynamic contribution patterns, it is straightforward to model newcomers' early career contribution activities as time series, using collected data. As we mentioned above, there are six activities based on modification types of the source code files. In this paper, we focus on the behavior of **MODIFY**. Because it is the central behavior among the six behaviors, as shown in Fig. 3, over 80% source code contributions are MODIFY, which could reflect most of a developer's technical contributions. Besides, other types of activities cannot help us better understand a newcomer's contribution. For example, DELETE means deleting some lines of code or some source code files, which newcomers are only allowed to do that under certain circumstances, also

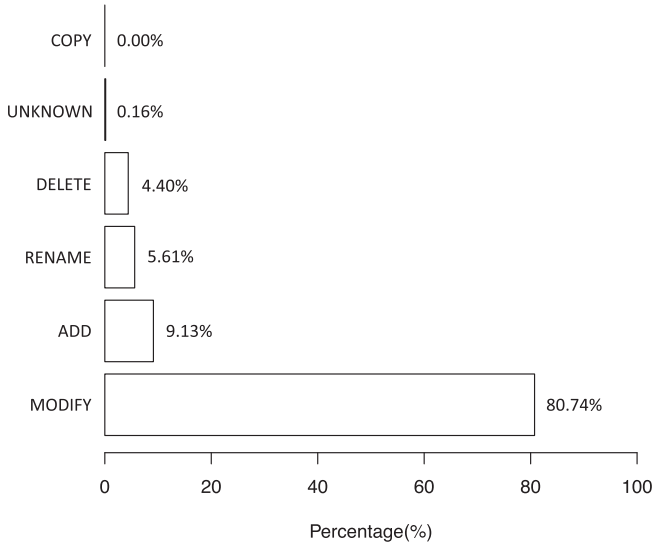

Fig. 3. The percentage of six modification types in our dataset.

due to the limited number of this type of contribution, it is impossible to model time series of DELETE with detailed contribution activities. Moreover, UNKNOWN is the activities that cannot be categorized into other types, which could not provide clear information about newcomers. Therefore, MODIFY behavior could be more related to newcomers, comparing with other behaviors.[1]

According to a developer's common pathway described in Section II-A, anyone who newly joins the project and successfully submits the first commit would be considered as a newcomer in one's early career, and they are identified at the project level in this paper. Then we operationalized the term "**early career**" as the first 180 days (≈ six months) since a newcomer submitted their first code commit and got accepted. According to the survival analysis reported in [8], developers often stop contributing to a project after a while (fail to survive), usually several months. Besides, Zhou and Mocus revealed that it often takes 2-6 months for newcomers to become productive and achieve fluency in the projects [29]. Therefore, we set the 180 days limit to allow their contribution patterns to become stable.

Newcomers' MODIFY behavior was measured at the source code file level, because git is designed as a file-based version control system, and commits are tracked at the file level [71], which is the data we extracted from repositories. We modeled each day's count of MODIFY behavior as a data point, creating a 180 data points time series for each newcomer in each project. An individual's time series enabled us to characterize a newcomer's dynamic contribution patterns from a longitudinal perspective. For each newcomer of each project, the time series was in the the following form:

$$< 1, x_1 >, < 2, x_2 >, \ldots, < i, x_i >, \ldots, < 180, x_{180} > \quad (1)$$

where i indexes days since the first commit, and x_i is the total number of source code files newcomer modified on day i.

1. In the supplementary material, we also compiled two datasets for ADD and DELETE, but this yielded little valid insight.

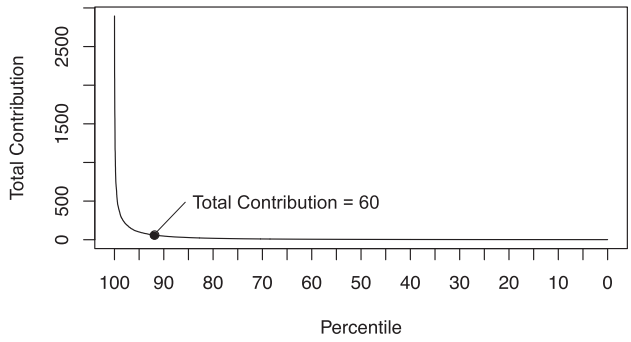

Fig. 4. The distribution of newcomers' total contribution in their early career.

Literature has shown that many OSS developers only make a handful of contributions in their entire tenure [72]. Our data confirmed such observations, as shown in Fig. 4, the majority of newcomers only had a limited amount of contribution throughout their early careers. Based on our observation on the dataset, most of contributors only had contribution in the first few weeks, then left the projects, which could be considered as causal contributors who were occasionally involved in the development. Therefore, if we included all contributors, most of the time series in the sample would be dominated by "0 s," which introduces too much noise to extract any meaningful features from them, and ultimately threats the validity of the results. Hence, we filtered out the developers who made less than 60 MODIFY in their first 180 days, based on several empirical observations and practical considerations mentioned above. The filtering resulted in 8.14% developers from sampled projects. In total, our sample had 1,326 time series. By setting such a threshold for total contributions, we also guaranteed that the included individuals had a certain level of motivation to succeed in their projects rather than contribute in an ad hoc manner, as their contributions are much higher than the casual contributors, i.e., they average touch at least 10 source code files every month during their early career.

3.2.4 Extracting and Selecting Time Series Features

For each time series in our sample, we used tsfresh [73], a Python library, to extract a set of time series features. tsfresh calculates a large number of time series features in many types (756 in total). However, not all these features are useful; for example, some of them capture high-order characteristics of time series, thus were hard to understand and link to human behaviors. Therefore, we further removed features based on three criteria. The first criterion is that an included feature must be human-interpretable. High-order features beyond the time domain were removed. For example, *c*wt_coefficients, which calculates a continuous wavelet transform for the Ricker wavelet, could not be interpreted at the behavioral level, hence should be removed. The second criterion is that an included feature must have some variances. We hence removed features whose values are almost identical across data cases. The third criterion is that an included feature should not be significantly correlated with an already-included one. E.g., *r*ange_count(1, -1) is significantly correlated with *c*ount_below(0). Because we have included the latter, *r*ange_count(1, -1) has to be excluded to

TABLE 1
Selected Time Series Features. Some Intuitive Interpretations of Them in Our Context are Provided in Braces ()

Features		Description
Dynamic Fluctuations		
(1)	*a*gg_linear_trend(max, 5, slope)	Linear trend estimated using linear regressions with the aggregated (max) over 5 days (positive → increasing trend, negative → decreasing trends).
(2)	*a*gg_linear_trend(max, 10, slope)	Linear trend estimated using linear regressions with the aggregated (max) over 10 days (positive → increasing trend, negative → decreasing trends).
(3) – (5)	*c*3[(1), (2), (3)]†	Metrics of time series' non-linear trends with lag as 1, 2, 3, respectively (Higher → more non-linear).
(6)	*e*nergy_ratio_by_chunks(10, 0)	The ratio of the sum of squares of the first 1/10 of the whole series (higher → more contributions in the initial days).
(7)	*f*irst location of maximum	The first relative location of the max (higher → max appears later in time series).
(8)	*m*ean change	The mean of the differences between adjacent datapoints (higher → less consistent‡).
(9)	*r*atio value number to time series length	The ratio of unique values to the length of the time series (higher → less consistent‡).
(10)	*s*ymmetry_looking(0.05)	If the distribution of x looks symmetric with parameter as 0.05 (higher → less dynamic).
(11) – (13)	*t*ime_reversal_asymmetry_statistic[(1), (2), (3)]†	Metrics of the time-reversal asymmetry with lag as 1, 2, 3, respectively, (higher → less dynamic).
Complexity		
(14)	*a*pproximate_entropy(2, 0.1)	Approximate entropy measuring time series' complexity (higher → more complex).
(15)	*c*id_ce(False)	The complexity without normalization (higher → more complex).
(16)	*s*ample entropyy	Sample entropy assessing the complexity of time series (higher → more complex).
Contribution Frequency		
(17)	*c*ount_below(0)	The percentage of values that are ≤ 0 (higher → more days without contribution).
(18)	*l*ongest strike above mean	The length of the longest subsequence $> mean$ (higher → more consecutive days with contribution).
(19)	*l*ongest strike below mean	The length of the longest subsequence $\leq mean$ (higher → more consecutive days without contribution).
(20)	*v*ariance	The variance of time series (higher → less consistent‡).
Amount of Contribution		
(21)	*l*arge_standard_deviation(0.2)	If the standard deviation of time series is higher than $0.2 \times (max - min)$ (higher → less consistent‡).
(22)	*p*ercentage of reoccurring datapoints	The percentage of the data points occurring more than once (higher → more consistent‡).
(23)	*q*uantile(0.9)	The value greater than 90% of the ordered values from time series (higher → less consistent‡).
(24)	*s*um of reoccurring data points	The sum of all data points that occur more than once (higher → more consistent‡).

†‡ *denotes the family of three features with different lags;* ‡ *"consistent" refers to "consistent daily contributions."*

avoid multicollinearity in later analysis. Finally, we have a set of 24 features. Table 1 introduces them with short descriptions and intuitive interpretations in our context, and we grouped them into four categories, i.e., *d*ynamic fluctuations, *c*omplexity, *c*ontribution frequency, and *a*mount of contribution, based on the characteristics of time series they reflected.

Note that the above features are not very intuitive. To help readers better understand them, we use examples in Fig. 5 to visually explain how these abstract features reflect the time series' characteristics. We use examples 1 (top) and 2 (bottom) to refer to them. At first glance, example 1's time series shows some regular patterns and is quite stable over the entire period; but, example 2's time series is much more irregular. Obviously, they should have quite different values on the 24 selected features, which are supposed to characterize the corresponding time series. Table 2 presents the values of these features for the two examples. Now, let us combine them with the two time series plotted in Fig. 5 to have a close look at how these abstract features describe the corresponding time series.

To keep the paper concise, we are going to focus on three features (*a*gg_linear_trend(max, 5, slope), *c*ount_below(0), and *s*ample entropy) instead of looking all 24 selected features. First, *a*gg_linear_trend(max, 5, slope) reflects the general linear trend of the time series. If the maximum value in the consecutive 5 days decreases over the entire period, the value of this feature would be negative, and vice versa. In example 1, the max value of 5 days is always 10. Thus, example 1's *a*gg_linear_trend(max, 5, slope) is "0." In example 2,

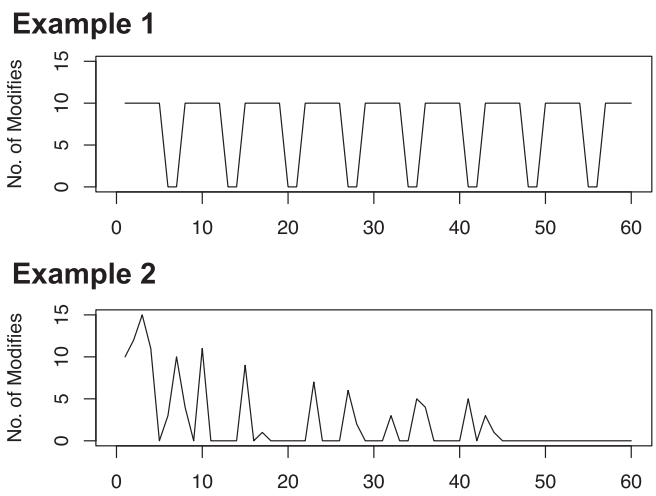

Fig. 5. Two examples of time series. Note that they are not from our empirical data, instead, we artificially created them to exemplify the differences in time series features between regular and irregular time series.

TABLE 2
Selected Time Series Features of the Examples Show in Fig. 5

Features		Example 1	Example 2
Dynamic Fluctuations			
(1)	*a*gg_linear_trend(max, 5, slope)	0.00	-1.11
(2)	*a*gg_linear_trend(max, 10, slope)	0.00	-2.54
(3)	*c*3(1)	448.28	67.24
(4)	*c*3(2)	142.86	11.79
(5)	*c*3(3)	296.30	42.78
(6)	*e*nergy_ratio_by_chunks(10, 0)	0.11	0.55
(7)	*f*irst location of maximum	0.00	0.03
(8)	*m*ean change	0.00	-0.17
(9)	*r*atio value number to time series length	0.03	0.22
(10)	*s*ymmetry_looking(0.05)	0.00	0.00
(11)	*t*ime_reversal_asymmetry_statistic(1)	0.00	-24.55
(12)	*t*ime_reversal_asymmetry_statistic(2)	0.00	-55.89
(13)	*t*ime_reversal_asymmetry_statistic(3)	-18.52	-32.69
Complexity			
(14)	*a*pproximate_entropy(2, 0.1)	0.33	0.57
(15)	*c*id_ce(False)	40.00	30.72
(16)	*s*ample entropyy	0.44	0.55
Contribution Frequency			
(17)	*c*ount_below(0)	0.27	0.68
(18)	*l*ongest strike above mean	5.00	4.00
(19)	*l*ongest strike below mean	2.00	17.00
(20)	*v*ariance	19.56	14.07
Amount of Contribution			
(21)	*l*arge_standard_deviation(0.2)	1.00	1.00
(22)	*p*ercentage of reoccurring datapoints	1.00	0.54
(23)	*q*uantile(0.9)	10.00	9.1
(24)	*s*um of reoccurring data points	440	71

while there are some increases in the early period, the overall trend of maximum values in the consecutive 5 days exhibit some decreases. Thus this feature has a negative value (-1.11). Second, regarding *c*ount_below(0), which counts the percentage of values that are less than or equal to 0, example 1 has a much smaller value than example 2 does (0.27 vs. 0.68). It fits what we can easily learn from the two time series. Lastly, although there are more "0"s in example 2, it looks more complex than example 1 in its visual form. The values of the feature *s*ample entropy (0.44 vs. 0.55), which captures the complexity of the time series, do confirm this.

In the context of a newcomer's early career, the two examples form sharp visual contrasts in different contribution patterns. We may view the first example represents a developer who regularly contributes in a stable fashion, while the second represents a developer who makes contributions in a less consistent and persistent manner. To sum up, these features capture multiple detailed dynamic characteristics of a newcomer's contribution patterns from multiple perspectives. Thus, using them to describe, summarize, and analyze individual newcomers' contribution patterns is proper since this study aims to identify the correlations between OSS newcomers' dynamic contribution patterns and technical success in early career.

3.2.5 Calculating Developers' Centrality

A newcomer's technical success was measured by his or her "eigenvector centrality" at the source code file level. In social network analysis, eigenvector is a robust metric that could deal with various network structures, such as disconnected networks. Then eigenvector centrality represents an individual's influence on a network [74], in our case, it represents a developer's influence on the network of source code files, which is the technical influence in project; and has been widely adopted in multi-disciplines. From an organizational behavioral perspective, members' influence in a community signifies their success, which is also held in open source development [75], [76], [77]. Furthermore, eigenvector centrality could avoid non-technical factors' impact, such as the impact of commercial sponsors, or one's ability in socialization with other contributors. Comparing with other metrics, it tends to be more appropriate in our analysis. For example, closeness would maximize the impact of certain nodes in the network, degree centrality is limited to reflect the impact on the whole network, and PageRank is a variant of eigenvector centrality in certain scenarios. Based on these considerations, we chose eigenvector centrality to measure the technical success. Newcomers with a high eigenvector centrality indicate that they have been likely to touch the core part of the project and technically influential. It usually evidences their expertise, skills, and importance to the project. Therefore, a higher centrality means a higher possibility of becoming core developers and achieving technical success in an OSS project [17], [78].

We applied the two-step method developed in Jergensen *et al.* [17] to calculate each newcomer's eigenvector centrality. The first step used each project's commit data to build a network of source code files based on the concept of logical commits [79]. This network provided us each source code file's eigenvector centrality [80]. In the second step, we first calculated the centrality of a newcomer's every commit by averaging all files' eigenvector centrality in the commit. Then, one's overall eigenvector centrality was computed as the sum of the eigenvector centrality of all commits. Since we did not deal with any other types of centrality, we would simply use "centrality" instead of "eigenvector centrality" in the rest of the article.

3.2.6 Identifying Genders

We had two control variables in the statistical analysis, one of them is gender, thus we need to identify developers' genders (male vs. female). It is challenging to infer the gender automatically and the results are not accurate enough [81]. Thus, we manually identified the gender in this paper. A developer's gender was manually inferred based on their profile information on GitHub and external sources, particularly social media sites. Some developers would provide cues of their gender identities on GitHub profiles, i.e., profile photos of their real faces, or pronouns. For those who did not make their gender identities explicit, we used the external sources, such as LinkedIn, personal websites, to infer their genders. Two researchers independently identified sampled developers' gender identities. The inter-rater reliability [82] was 0.92 (Cohen's kappa), indicating excellent agreements. They jointly resolved the disagreements and removed the individuals (69) whose gender could not be determined as either female or male. Finally, we reached a sample containing data from 62 females and 1195 males.

4 Data Analysis Strategies

We are going to introduce how we analyze the data in this section. In general, we use Hierarchical Linear Modeling (HLM) to answer **RQ$_1$** and **RQ$_2$**. All statistical analyses were performed using R version 3.6.1. The data was made available at: bit.ly/3mMNCDN.

4.1 Analysis for RQ$_1$

RQ$_1$ is about establishing correlations between time series features and centrality. It is straightforward to employ regression techniques to fulfill the task. However, note that our data was nested in nature, i.e., a project often contained multiple newcomers, which violates the independence of observations assumption. Thus, the correlations to be established might be potentially impacted by the nesting between a project and its newcomers. Such a nesting must be accounted for in the model to avoid misrepresenting the effects [83]. Hierarchical Linear Modeling (HLM) enabled us to deal with multiple-level regression modeling. Hence, we built an HLM model with all 1257 data cases in our sample to fulfill this task. According to literature, gender could be a variable related to developer's contribution [37], [84], and different ecosystems might also have influence [20], thus, we also included these two variables as control variables, to better understand newcomers' contribution patterns. In total, the HLM model had 27 variables: 24 aforementioned time series features as *i*ndependent variables, and two control variables, i.e., a binary variable for gender (0: female, 1: male), and a categorical variable represent the three ecosystems (coded as two binary variables in the model). The *d*ependent variable is the level of technical success a newcomer achieved in their early career, measured by the aggregated centrality over 180 days (see Section III-B5). A project's impacts were modeled as random effects. We followed standard procedures to build the regression model and performed the standard model diagnosis. To measure the model's explanation power, we adopted the pseudo conditional and marginal R^2s proposed in Xu [85].

4.2 Analysis for RQ$_2$

Similarly, we used HLM in analyses for **RQ$_2$**. We built three HLM models for the three ecosystems, respectively, using the subsample corresponding to them. There were 513 data cases in the Apache subsample, 369 in the OpenStack subsample, and 375 in the Python Data Science subsample. The models' independent variables are the same set of 24 time series features used for **RQ$_1$**, and a binary variable for gender as a control variable. The dependent variable is centrality. As we did before, random effects were used to model projects' impacts. Doing so enabled us to compare the effects of independent variables across ecosystems.

5 Results and Findings

This section reports the results and findings, and answers **RQ$_1$** and **RQ$_2$**. We first present the descriptive statistics and then organize the results according to the two **RQ**s. While presenting and interpreting the results of **RQ$_1$** and **RQ$_2$**, we may occasionally give some propositions implying possible but not definitive causalities, which is a common practice in data-driven research related to human and social factors. However, please bear in mind that all regression models establish correlations only, rather than causalities.

TABLE 3
Descriptive Statistics of Time Series Features

Features		Mean	SD
Dynamic Fluctuations			
(1)	*a*gg_linear_trend(max, 5, slope)	-0.14	0.55
(2)	*a*gg_linear_trend(max, 10, slope)	-0.51	2.01
(3)	*c*3(1)	143.13	2,114.34
(4)	*c*3(2)	41.65	510.33
(5)	*c*3(3)	21.68	222.44
(6)	*e*nergy_ratio_by_chunks(10, 0)	0.25	0.34
(7)	*f*irst location of maximum	0.38	0.31
(8)	*m*ean change	-0.11	0.43
(9)	*r*atio value number to time series length	0.06	0.03
(10)	*s*ymmetry_looking(0.05)	0.84	0.37
(11)	*t*ime_reversal_asymmetry_statistic(1)	-2,048.87	128,673.40
(12)	*t*ime_reversal_asymmetry_statistic(2)	-1,879.51	46,823.27
(13)	*t*ime_reversal_asymmetry_statistic(3)	-392.02	10,248.40
Complexity			
(14)	*a*pproximate_entropy(2, 0.1)	0.36	0.21
(15)	*c*id_ce(False)	102.45	161.96
(16)	*s*ample entropyy	0.26	0.21
Contribution Frequency			
(17)	*c*ount_below(0)	0.87	0.10
(18)	*l*ongest strike above mean	3.06	1.90
(19)	*l*ongest strike below mean	67.45	43.58
(20)	*v*ariance	129.74	758.28
Amount of Contribution			
(21)	*l*arge_standard_deviation(0.2)	0.02	0.13
(22)	*p*ercentage of reoccurring datapoints	0.49	0.18
(23)	*q*uantile(0.9)	1.85	3.48
(24)	*s*um of reoccurring data points	72.92	111.26

5.1 Descriptive Statistics

Table 3 summarizes the basic descriptive statistics of time series features. For example, we can find that newcomers' contributions decreased on average because the first two features representing linear trends have negative values. Due to the restriction of space, we do not explain them one by one. However, we may refer back to them in discussing and interpreting the regression results later.

5.2 RQ$_1$: Dynamic Patterns Vs. Technical Success

*M*odel 1 in Table 4 summarizes the results of the HLM built on the data of the whole sample (all three ecosystems). In general, *M*odel 1 has a 0.610 conditional R-squared and 0.445 marginal R-squared, thus could explain a substantial amount of variances of the dependent variable (*c*entrality). Table 4 also marks significant independent variables in regression models. In this section, we will discuss these significant features based on their categories. For reader convenience considerations, we provide the row index of each feature in Table 4 along with its name when referring to it.

5.2.1 Dynamic Fluctuations

The first category is features related to time series' dynamic fluctuations at different temporal scales. In this category, there are 6 significant features in the HLM model. They are: *a*gg_linear_trend(max, 5, slope) (Row 1), *a*gg_linear_trend(max, 10, slope) (Row 2), *c*3(3) (Row 5), *e*nergy_ratio_by_chunks(10, 0)

TABLE 4
Regression Model for $\mathbf{RQ_1}$ (Whole Sample) and $\mathbf{RQ_2}$ (Subsamples)

		Subsamples		
	Whole Sample	**Apache**	**OpenStack**	**Python Data Science**
	*M*odel 1-Centrality	*M*odel 2-Centrality	*M*odel 3-Centrality	*M*odel 4-Centrality
Dynamic Fluctuations				
(1) *a*gg_linear_trend(max, 5, slope)	4.471∗∗∗	1.762∗	-5.365	4.817
(2) *a*gg_linear_trend(max, 10, slope)	-1.315∗∗∗	-0.516∗	0.889	-1.292
(3) *c*3(1)	0.000	0.000	-0.014∗	-0.000
(4) *c*3(2)	-0.000	-0.000	-0.008	-0.004
(5) *c*3(3)	-0.003∗∗∗	-0.001∗	0.040	-0.003
(6) *e*nergy_ratio_by_chunks(10, 0)	-0.972∗	0.141	-0.228	-1.847
(7) *f*irst location of maximum	-0.449	-0.271	0.587	-0.392
(8) *m*ean change	-1.908∗∗∗	-0.099	1.416	-1.611
(9) *r*atio value number to time series length	17.790	2.258	36.110∗	28.980
(10) *s*ymmetry_looking(0.05)	0.499	1.664∗∗∗	0.278	1.151
(11) *t*ime_reversal_asymmetry_statistic(1)	0.000∗	0.000	-0.001	0.000
(12) *t*ime_reversal_asymmetry_statistic(2)	-0.000	-0.000	-0.002	0.000
(13) *t*ime_reversal_asymmetry_statistic(3)	0.000∗∗∗	0.000	0.001	-0.004∗
Complexity				
(14) *a*pproximate_entropy(2, 0.1)	-8.109∗∗∗	-8.308∗∗∗	-6.370∗	-10.170∗∗
(15) *c*id_ce(False)	-0.006∗∗∗	-0.001	-0.008	-0.030∗∗∗
(16) *s*ample entropyy	-6.974∗∗	5.466∗∗	-15.470∗	-8.558
Contribution Frequency				
(17) *c*ount_below(0)	-39.680∗∗∗	-9.217∗	-66.690∗∗∗	-46.230∗∗∗
(18) *l*ongest strike above mean	-0.002	-0.275∗∗	-0.052	-0.059
(19) *l*ongest strike below mean	-0.010∗	-0.011∗∗	0.006	-0.006
(20) *v*ariance	0.001∗	0.000	0.002	0.017∗∗∗
Contribution Amount				
(21) *l*arge_standard_deviation(0.2)	-5.873∗∗∗	-6.174∗∗∗	-0.673	-4.767∗∗
(22) *p*ercentage of reoccurring datapoints	-1.002	-2.272∗∗	0.729	-1.051
(23) *q*uantile(0.9)	-0.422∗∗∗	-0.178	0.486 ∗	-0.565∗
(24) sum of reoccurring data points	0.030∗∗∗	0.028∗∗∗	0.005	0.035∗∗∗
Binary Variables				
(25) *g*ender	0.272	0.354	0.055	0.431
(26) *e*cosystem_d1	2.950∗∗	-.-	-.-	-.-
(27) *e*cosystem_d2	-0.320	-.-	-.-	-.-
*N*o. of Obs.	1,257	513	369	375
Conditional R^2	0.610	0.759	0.641	0.673
*M*arginal R^2	0.445	0.655	0.545	0.407

[1]∗ $p < 0.05$, ∗∗ $p < 0.01$, ∗∗∗ $p < 0.001$.

(Row 6), *m*ean change (Row 8), and *t*ime_reversal_asymmetry_statistic with parameters 1 and 3 (Row 11 & 13).

Let us focus on the linear trend first. The features *a*gg_linear_trend(max, 5, slope) and *a*gg_linear_trend(max, 10, slope) are related to the linear trend of time series. First of all, most of their values in our data cases are negative (see Table 3), indicating the *m*ax in each chunk (5 or 10) is decreasing. Now, let us have a look at their coefficient. The first's coefficient is positive (Row 1). It basically means that: the smaller the decrease (in a relatively short period, 5 days) trend, the larger the centrality. Meanwhile, the second's coefficient is negative (Row 2), indicating some opposite effect (in a relatively long period, 10 days). Nevertheless, such an opposite effect could not offset the short-term decrease's negative effect, as the time series is decreasing in its global trend. Besides, the *e*nergy_ratio_by_chunks(10, 0)' s and *m*ean change's coefficients are negative (Row 6 & 8), which also confirms that the decreases in contributions were negatively correlated with a newcomer's centrality. Thus, regarding the linear trend, we could conclude that *s*hort-term, drastic decreases on a newcomer's contributions in early career could be negatively associated with their centrality in general.

Regarding the non-linear trend, *c*3(3)'s negative coefficient (Row 5) clearly indicates that *l*arge, non-linear fluctuations had a negative correlation with a newcomer's centrality. Meanwhile, the coefficients of another two metrics of non-linearity, *time_reversal_asymmetry_statistic(1)* and *time_reversal_asymmetry_statistic(3)* are positive but very small (≈ 0) in the regression model (Row 11 & 13), which also confirm the above negative correlation.

5.2.2 Complexity

The second category is about time series' complexities. The significant features in this category include *a*pproximate_entropy(2, 0.1) (Row 14), *c*id_ce(False) (Row 15), and *s*ample entropy (Row 16). The features *a*pproximate_entropy(2, 0.1)

and *s*ample entropy have coefficients that are both negative. Thus, the less complex a newcomer's time series is, the higher their centrality is. Besides, *c*id_ce(False) measures complexity by counting peaks and valleys in a time series. Therefore, its negative coefficient means more peaks and valleys in a newcomer's time series would negatively correlate to the centrality (Row 15), consistent with the entropy feature. Therefore, we can conclude that *t*he complexities of newcomer's contribution patterns could be negatively associated with their centrality. Being consistent in making contributions would inevitably reduce complexity. Thus, these results emphasize the importance of **keeping consistent contributions** again.

5.2.3 Contribution Frequency

The third category is about newcomer's contribution frequency. The significant features include *c*ount_below(0) (Row 17), *l*ongest strike below mean (Row 19), and *v*ariance (Row 20).

Given that most individuals in our sample had less than one contribution per day on average (see Section III-B3), *c*ount_below(0) and *l*ongest strike below mean reflect two aspects about days without contribution. The former describes the overall count of days without contribution; the latter often counts the longest strike of days without contribution. Their negative coefficients (Row 17 & 19) show that no contribution days, particularly consecutive no contribution days, would negatively correlate with one's centrality. Meanwhile, the positive coefficient of *v*ariance indicates similar correlations (Row 20). Higher contribution frequency could produce more nonzero data points in time series, resulting in higher variance, which in turn positively correlated to the centrality. To sum up, *b*eing persistent (contributing frequently) is positively correlated with one's centrality.

5.2.4 Amount of Contributions

The fourth category is about the amount of contributions, which is the value of datapoints in time series. The significant features are *q*uantile(0.9), *l*arge_standard_deviation (0.2), and *s*um of reoccurring datapoints.

First, let us have a look at *q*uantile(0.9) (Row 23), which captures the value greater than 90% ordered values in a newcomer's time series data. High *q*uantile(0.9) means a newcomer made an extremely large amount of contributions in some days, which signifies some inconsistencies in their time series. Its negative coefficient reconfirms the *p*ositive correlations between consistent contribution patterns and the centrality. Second, *s*um of reoccurring datapoints is positively correlated to centrality (Row 24). When the number of a newcomer's contributions in a day is the same as that in other days in the time series, *s*um of reoccurring datapoints increases. Thus, *t*he consistent amount of daily contributions could be positively correlated with the centrality. The last feature, *l*argest_standard_deviation(0.2) (Row 21), basically conveys the same message.

5.2.5 Answers to $\mathbf{RQ_1}$

Based on the above discussion, we can answer the $\mathbf{RQ_1}$ as follows:

*O*ur analysis confirms the existence of the correlations between newcomers' contribution patterns and their centrality. Specifically, the centrality is positively correlated with frequent, and consistent daily amounts of contributions; and negatively correlated with short-term, drastic decreases in contributions, the complexity of time series, and long strikes without contributions. In general, being consistent and persistent is positively associated with newcomers' technical success measured by their centrality.

5.3 $\mathbf{RQ_2}$: Patterns Across Ecosystems

In *M*odel 1, the control variable representing ecosystems was significant (*e*cosystem_d1), showing that the correlations between contribution patterns may be contingent on ecosystems. We split the whole sample into three subsamples according to the ecosystem a project belongs to. With each subsample, we built an HLM model to analyze the associations between newcomer's contribution patterns and centrality, and then compared the three models (*M*odel 2–4 in Tables. 4). All models recorded higher conditional R-squared values (0.610, 0.641, and 0.673) than the whole sample model (Model 1), which means the ecosystem-specific model might describe the correlations better. We summarized common patterns and ecosystem-specific ones as follows.

5.3.1 Common Patterns

There are some common patterns shared by regression models based on the whole sample and the three subsamples. The coefficients of *c*ount_below(0) are negative in all regressions (Row 17), and the feature itself is also significant in all models, showing that *c*ontinuous contributions are positively associated with the centrality, no matter in which ecosystem. Besides, complexity-related features, such as *a*pproximate_entropy(2, 0.1), are significant in multiple models, and have negative coefficients (Row 14). Thus, *l*ess complex contribution dynamics are also positively associated with the centrality. Since consistent contribution patterns are apparently less complex, such results indeed reconfirm the importance of consistency in early career contributions.

5.3.2 Ecosystem-Specific Patterns

Now let us have a look at ecosystem-specific patterns. In *M*odel 2 (Apache), the correlations between linear trends and the centrality are significant, and keep accordance with those in *M*odel 1 (see the coefficients of the first two features). Besides, *l*ongest strike below mean has a negative coefficient (Row 19). Both suggest that *d*rastic short-term decreases and noncontinuous contribution patterns are negatively correlated with the centrality. However, we did not observe the significance of such patterns in *M*odel 3 (OpenStack) and *M*odel 4 (Python Data Science). Meanwhile, in *M*odel 2 & 4, newcomer's centrality could be positively correlated to consistent daily contributions (*l*arge_standard_deviation(0.2), and *s*um of reoccurring data points) (Row 21 & 24), however it is not significant in *M*odel 3. In *M*odel 2, the features *a*pproximate_entropy(2, 0.1) (Row 14) and *s*ample entropy (Row 16) have conflicts.

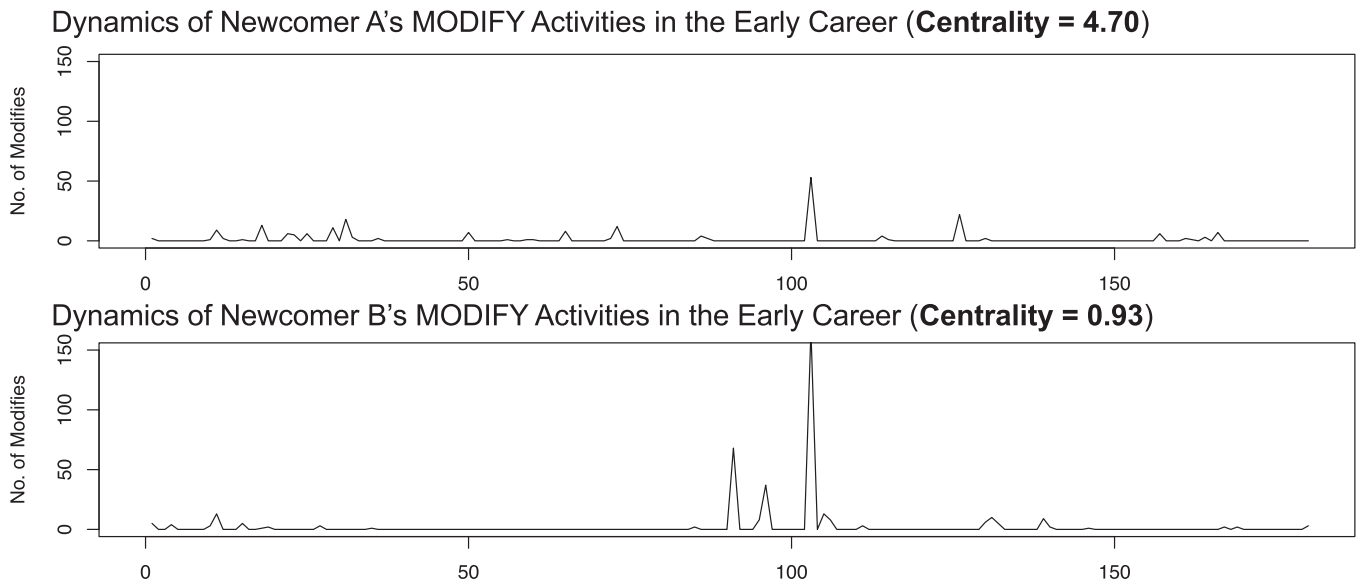

Fig. 6. Newcomer A & B's dynamics of their MODIFY activities in their early careers. It indicates some intuitive connection between their different contribution patterns and centrality.

In *M*odel 3, we observed a negative correlation between non-linear trend and the centrality (*c*3(1)) (Row 3), while this correlation is not significant in the other two models. Another interesting observation is that *r*atio value number to time series length is significant only in *M*odel 3, suggesting an inconsistent contribution pattern, and we were unable to explain the contradiction. In *M*odel 4, *c*id_ce(False) (Row 15) has negative coefficient, indicating potential negative correlations between complexities and the centrality.

5.3.3 Answers to $\mathbf{RQ_2}$

Based on the above discussion, we can answer the $\mathbf{RQ_2}$ as follows:

> *N*o matter which ecosystem a project belongs to, its newcomers' consistent and continuous contribution patterns are generally positively associated with the centrality. However, there are also differences regarding the contribution patterns that are correlated with the centrality across ecosystems.

5.4 Interpreting the Results by Real-World Examples

Based on the results and findings above, we use several examples from our sample to provide some intuitive interpretations of the findings regarding newcomer's contribution patterns and technical success in the projects.

Fig. 6 visualizes the dynamics of two newcomers' MODIFY activities in our sample with two time series. We use A & B to refer to these two newcomers in this section. Obviously, there are significant disparities in their centrality achieved after 180 days of participation: A's centrality is 4.70 while B's is 0.93. Meanwhile, there are some apparent differences in the two time series representing their contribution patterns, (1) A exhibits relatively stable contribution patterns, e.g., low fluctuations, consistencies regarding the daily contribution amount, while B's contributions often change drastically; (2) A contributes more frequently than B. Such differences, if translating to the time series features shown in Table 1 and combining with the regression results in Table 4, would suggest that the features of A's dynamics are coincidentally high on those having positive effects on one's centrality, while the features of B's dynamics are linked with those having negative effects. For example, B's dynamics records much higher value on the features such as ''count_below(0),'' ''longest strike below mean'' than A's does, which are features negatively associated with one's centrality.

6 Contribution Beyond Source Code

The above sections present the analysis and findings on the dynamic patterns of newcomers' source code contribution and their correlations with technical success (measured by the centrality) in their early careers. As we mentioned in the introduction, many newcomers also engage in contributions in documentation, in addition to source code. Such documentation contributions may also potentially have some effects on their technical success. Therefore, understanding the relationships between newcomers' contributions and technical success in early career shall consider such a type of contribution. We hereby designed and conducted a complementary case study to explore newcomers' documentation contributions, which provided answers to the $\mathbf{RQ_3}$. The case study is detailed in this section, including the data preparation process, the analysis strategy, and its results and findings.

6.1 Data Preparation

As shown in Fig. 7, we took several procedures to prepare data for the case study.

6.1.1 Sampling Targeted Projects and Identifying Documentation Contributions

To collect necessary evidence for the case study, we first investigated how a newcomer could contribute to documentation in

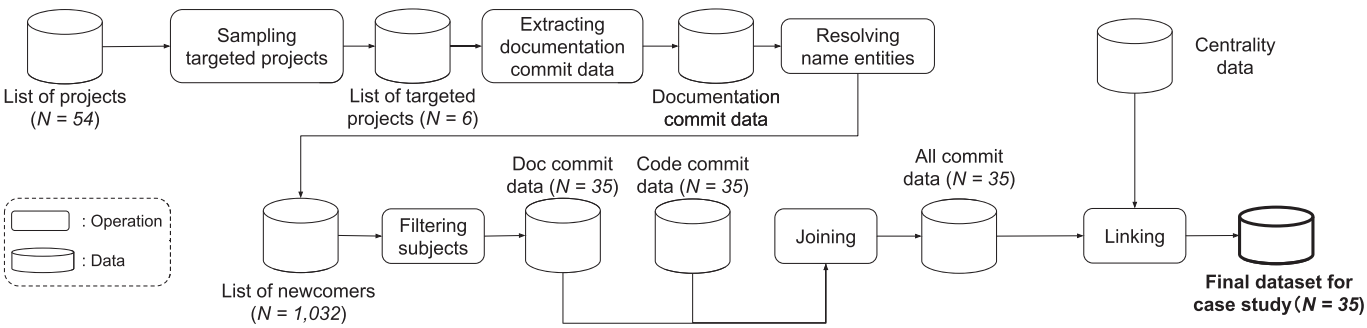

Fig. 7. Overview of the data preparation process for the case study on documentation contribution.

each sampled project, by manually checking their repositories' README files, Wiki pages hosted on GitHub and other relevant websites, e.g., project homepages. Usually, there is a section named "How to contribute," which might specify guidelines about contributing to documentation. We found out that 50 of 54 sampled projects had separated directories to accommodate members' documentation contributions in their official repositories. However, most of them had yet to establish explicit documentation contribution solicitation and management practices as they did for code contributions. Some projects' documentation was quite fragile and lacked continuous contributors.

Thus, we decided to focus on projects with sufficient documentation, rather than studying all 50 projects with separated documentation directories. For each ecosystem, we drew two projects as case study targets following two steps:

1) The project with most documentation files in the documentation directory would be selected;
2) The project with the highest percentage of documentation files in the repository would be selected; if this project had already been selected in the first step, the one with the second-highest percentage would be selected.

The above procedure resulted in six projects, which are: *S*hardingSphere and *J*Meter from the Apache ecosystem, *N*ova and *a*nsible-hardening from the OpenStack ecosystem, and *N*umPy and *B*okeh from the Python Data Science ecosystem. We then extracted the documentation commit data from these six projects' repositories. If a commit contains a file under the documentation directory, this commit would be extracted, and the files under the documentation directory would be considered as documentation files. Following the same data extraction process introduced above, we extracted the documentation commit data, including author, timestamp, SHA, commit message, and list of documentation files. Then we resolved the name entity and removed the bot accounts as we did for identifying source code contributions. Then, we had a sample of 1,032 newcomers having documentation contributions.

6.1.2 Filtering Subjects

In contrast to such a large pool of documentation contributors, we noticed that most newcomers' documentation contributions are quite low (≈ 14 documentation files), and 488 of them (≈ 47%) only edited one documentation file throughout their entire early career. Moreover, as shown in Fig. 8, over 90% of them only touched less than 20 documentation files, while 90% of newcomers who contributed to source code modified less than 60 source code files during their early careers. Therefore, although each project often has a large pool of newcomers working as documentation contributors, lots of them are casual/one-time contributors, which stopped their contribution and dropped out soon. To facilitate fair comparisons with newcomers included in the main study and remove noisy data of casual contributors, we excluded newcomers who made less than 20 documentation contributions during their early career, resulting in a sample of 85 newcomers from the six sampled projects.

Since some newcomers who only contributed to documentation never touched source code during their early career, it was impossible to assess their technical success using the centrality measure. Meanwhile, the case study's scope automatically excluded the newcomers who only made source code contributions. Thus, we should target those who had substantial amounts of both types of contributions during their early career. By applying this rule, the final sample of subjects contains 35 newcomers from four projects, *J*Meter: 3, *N*ova: 18, *N*umPy: 6, and *B*okeh: 8 (Fig. 9).

6.1.3 Compiling Case Study Evidence

For the identified 35 newcomers, we compiled a dataset as the main evidence for the case study. The dataset contained data related to these individuals' all commits made during their early career by joining documentation commit data and source code commit data. The total number is 14,456. Then, these 35 newcomers' centrality data were linked back.

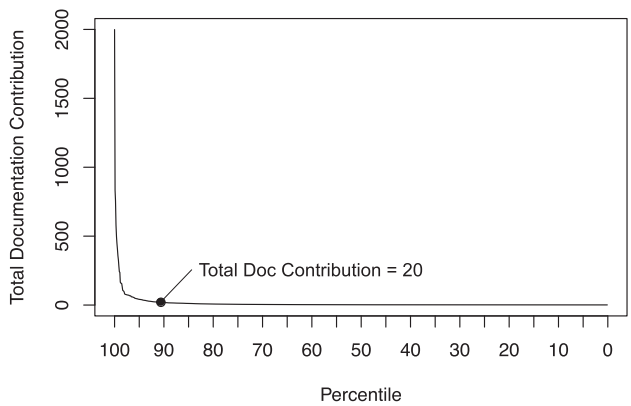

Fig. 8. The distribution of newcomers' total documentation contribution in their early career.

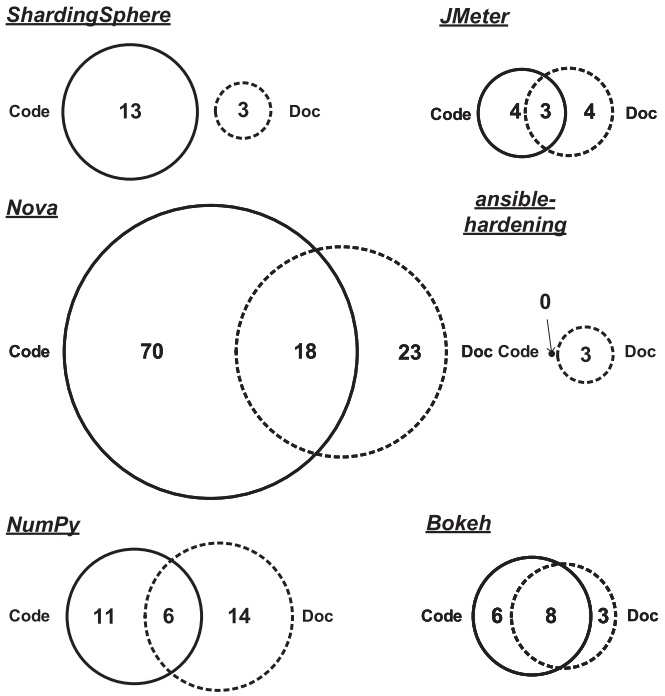

Fig. 9. The number of newcomers with types of contributions in each sampled projects.

The final dataset thus consisted of three components: documentation commit data, code commit data, and centrality data. Each piece of commit data had detailed information, including its author, timestamp, SHA, and commit message. The dataset then served as the main evidence for the case study and would be used to answer $\mathbf{RQ_3}$.

6.2 Analysis for $\mathrm{RQ_3}$

We adopted a mixed (quantitative and qualitative) strategy to analyze the data from the 35 sampled newcomers (the primary evidence of the case study). As we did in the above sections, the quantitative analysis focused on providing descriptive statistics rather than making statistical inferences. Similar to what we did for code contributions, we also modeled each newcomers' documentation contributions as time series, and summarized the characteristics of dynamic patterns of documentation contributions to develop an understanding of this type of contribution, particularly on the differences between the two types of contributions.

Since the documentation contributions usually influence one's early career technical success (measured by the centrality) indirectly through code contributions, the qualitative analysis would focus on the relative temporal relationships and interactions between an individual's code and documentation contribution. We plotted all 35 individuals' dynamic contributions of documentation and source code. Each plot contained a newcomer's contributions of both types during his or her entire early career. And then, we qualitatively identified different trajectories of these plots according to these plots' temporal patterns and the number of contributions represented. Two authors first coded the dynamic trajectories showing in the plots independently, which achieved an initial 0.96 IRR. A few disagreements were resolved through discussions. The coding process resulted in four trajectories. We further linked a newcomer's contribution trajectory with the detailed commit data and the centrality to provide answers to $\mathbf{RQ_3}$ through qualitative reasoning.

TABLE 5
Descriptive Statistics of Doc Contribution Time Series Features

Features		Mean	SD
Dynamic Fluctuations			
(1)	*a*gg_linear_trend(max, 5, slope)	-0.17	0.46
(2)	*a*gg_linear_trend(max, 10, slope)	-0.64	1.80
(3)	*c*3(1)	20.06	98.78
(4)	*c*3(2)	11.47	58.76
(5)	*c*3(3)	10.13	59.44
(6)	*e*nergy_ratio_by_chunks(10, 0)	0.44	0.40
(7)	*f*irst location of maximum	0.24	0.31
(8)	*m*ean change	-0.21	0.64
(9)	*r*atio value number to time series length	0.03	0.02
(10)	*s*ymmetry_looking(0.05)	0.97	0.17
(11)	*t*ime_reversal_asymmetry_statistic(1)	-18.31	72.60
(12)	*t*ime_reversal_asymmetry_statistic(2)	2.60	142.60
(13)	*t*ime_reversal_asymmetry_statistic(3)	23.06	221.00
Complexity			
(14)	*a*pproximate_entropy(2, 0.1)	0.18	0.19
(15)	*c*id_ce(False)	59.70	116.52
(16)	*s*ample entropyy	0.10	0.16
Contribution Frequency			
(17)	*c*ount_below(0)	0.94	0.08
(18)	*l*ongest strike above mean	2.34	1.41
(19)	*l*ongest strike below mean	99.37	49.43
(20)	*v*ariance	86.67	331.73
Amount of Contribution			
(21)	*l*arge_standard_deviation(0.2)	0.00	0.00
(22)	*p*ercentage of reoccurring datapoints	0.48	0.18
(23)	*q*uantile(0.9)	0.20	0.76
(24)	*s*um of reoccurring data points	23.69	39.80

6.3 Results and Findings

6.3.1 Descriptive Statistics

The documentation contributions could also be modeled as dynamic time series. We extracted the same set of 24 time series features for these time series. Table 5 summarizes descriptive statistics of these features. Comparing with the time series of code contribution (Table 3), there are some similarities, as well as differences.

First, regarding the dynamic fluctuations, the time series of both types of contribution are decreasing in general, according to the values of *agg_linear_trend(max, 5, slope)* and *agg_linear_trend(max, 10, slope)* (Row 1 & 2). However, the time series of documentation contribution tend to have a smaller non-linear trend (Row 3 & 4), also less complex in general (Row 14-16). Although the 35 newcomers had substantial documentation contributions compared with the newcomers who were excluded from the case study, their documentation contributions are still far less than code contributions in frequency and total amount. We observed more zeros (Row 17) and longer periods without documentation contribution (Row 19), and the documentation contributions tend to be less consistent (Row 18 & 20) than code contributions.

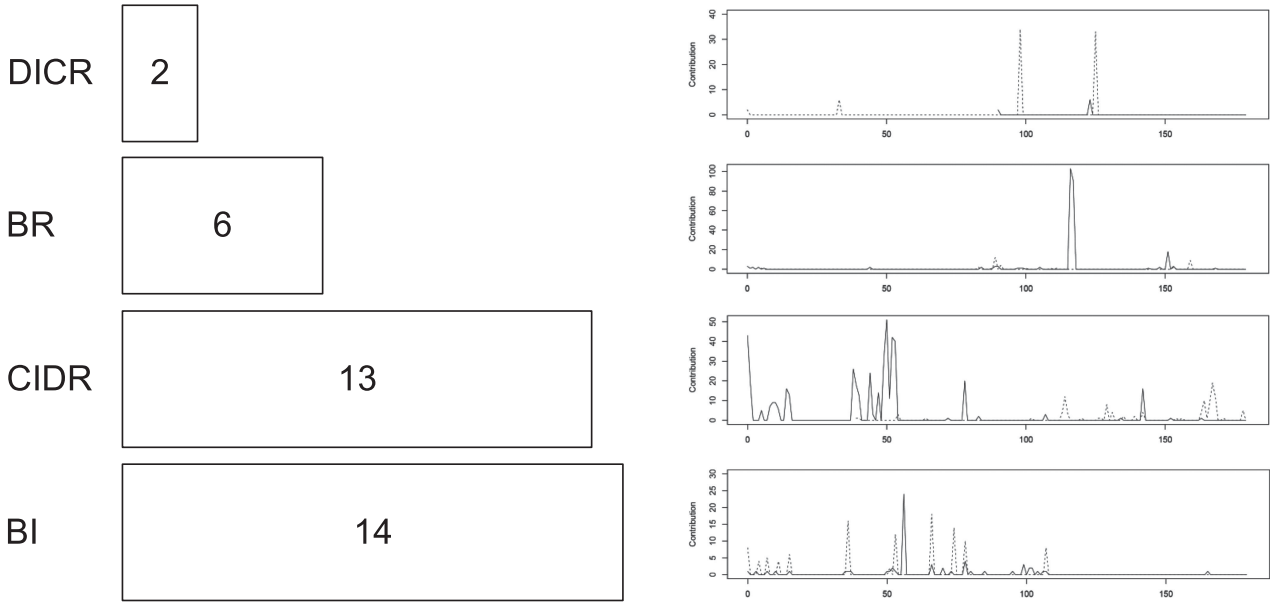

Fig. 10. The number of newcomers and an example in each trajectory. In each example, the solid line represents the dynamics of code contributions, while the dotted line represents the dynamics of documentation contributions.

6.3.2 Dynamic Trajectories of Newcomers' Code and Documentation Contributions

Section 6.3.2 and 6.3.3 report on the qualitative analysis' results and findings. First, Section 6.3.2 introduces the four dynamic trajectories. These identified trajectories are:

1) *C*ode-Intensive & Doc-Random (CIDR): This trajectory means that a newcomer had intensive code contributions but a very random and small amount of documentation contributions; besides, the first code contribution often appeared before the first documentation contribution. We use CIDR to refer to this trajectory. Among the 35 newcomers, 13 newcomers' plots exhibited such a trajectory, accounting for 37% in the whole sample.
2) *D*oc-Intensive & Code-Random (DICR): This trajectory means that a newcomer had intensive documentation contributions but a very random and small amount of code contributions; besides, the first documentation contribution often appeared before the first code contribution. We use *D*ICR to refer to this trajectory. Among the 35 newcomers, only two newcomers' plots exhibited such a trajectory, accounting for 6% in the whole sample.
3) *B*oth-Intensive (BI): This trajectory means that a newcomer had made intensive contributions in both source code and documentation. We use *B*I to refer to this trajectory. This trajectory contained most newcomers' plots (14 out of 35, 40%).
4) *B*oth-Random (BR): In the opposite to the third trajectory, some (6 out of 35, 17%) newcomers' contributions in both source code and documentation were random and limited. We named this trajectory as *B*oth-Random, shortened as *B*R.

The right part of Fig. 10 gives an example for each trajectory. The differences are obvious. Also, note that the newcomers' documentation contributions were very limited and random when they fell into trajectories CIDR or *B*R.

6.3.3 Linking Newcomers' Dynamic Trajectories With Their Centrality

We then linked these newcomers' centrality with their trajectories. In Fig. 11, we plotted all 35 newcomers' centrality, grouped by their trajectories. In addition, we used different color patterns to indicate the quartile distribution of their centrality values.

The first quartile (red dots in Fig. 11) includes nine newcomers who had the highest centrality. Three of them were in the CIDR trajectory, and six were in the *B*I trajectory. Neither of the other two trajectories had any newcomer achieving centrality falling into the first quartile. Besides, a large proportion (16 out of 27, 60%) of newcomers in CIDR and *B*I trajectories recorded above (or equal to) median centrality,

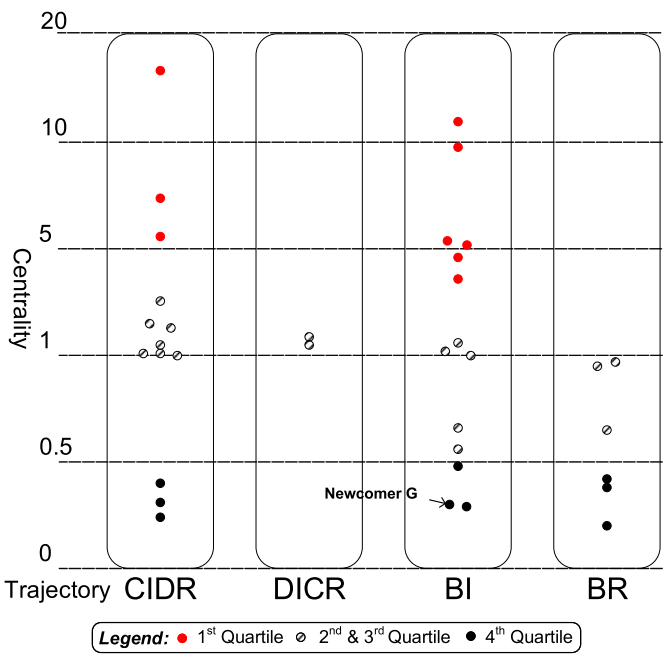

Fig. 11. The number of newcomers in each trajectory.

which accounts for almost 90% of newcomers who had above (or equal to) the median centrality. Thus, it confirms what was revealed in the main study, i.e., intensive code contributions in source code seem to have some positive correlations with high centrality. Then, the fourth quartile (black dots in Fig. 11) includes nine newcomers evenly distributed in three trajectories (*C*IDR, *B*I, *B*R). Particularly, newcomers in the fourth quartile account for half of all individuals in the *B*R trajectory; and none newcomer in this trajectory had above (or equal to) the median centrality. Both suggest that making random contributions does not help too much in achieving technical success in early careers.

Unfortunately, there is no direct evidence to enable us to determine if documentation contributions could be helpful, partially due to the limitation of the data. However, if digging deeper into the nine newcomers' commit information in the *B*I who had centrality in the first and last quartile, we could find an interesting coincidence. That is, for the six in the first quartile, many of their documentation contributions were independent of source code contributions, while for the three in the fourth quartile, most of their documentation contributions and source code contributions often appeared in the same commits. These documentation contributions were mostly about explanation or summary of their code contributions. For example, Newcomer G's all documentation contributions were part of commits containing source code contributions, and her centrality was among the lowest ones (marked in Fig. 11). Furthermore, in the trajectory *D*ICR where the documentation contributions obviously had little relevance with random code contributions, its two newcomers' achieved higher centrality than the median centrality. Therefore, intensive code-independent documentation contributions may be positively associated with high centrality, though we have no explicit evidence due to the restriction of the data. Further research is necessary to examine such relationships.

6.3.4 Answers to $\mathbf{RQ_3}$

Based on the above discussion, we can answer the $\mathbf{RQ_3}$ as follows:

> *M*aking code and documentation contribution intensively does help newcomers technical success in their early careers. However, there is no explicit evidence to support that documentation contribution's positive role. Meanwhile, documentation contributions unrelated to code contributions may potentially have positive correlations but require further investigations.

7 Discussion

7.1 Discussion of Findings

Our study's findings convey several important messages. First, a newcomer's early career contribution behaviors did correlate with their technical success measured in terms of eigenvector centrality. Prior literature has done an excellent job focusing on easing their onboarding process by systematically examining the barriers they faced and designing tools to help them make the first contribution [3], [4], [6], [28], [43], [48], [49]. Our work extends the prior work from the *o*nboarding to *e*arly career and shows that contribution behaviors matter technical success. Thus, our work provides a fresh perspective and potential solutions to the long-lasting problems in OSS projects, i.e., high dropout rates [8] and a large population of casual/one-time contributors [50], [72]. Based on our findings, socio-technical interventions on their contribution behaviors should be designed to empower newcomers in their early careers to help them achieve long-term success and advancement in an OSS project.

Second, regarding the specific contribution behavioral patterns, our findings suggest that a newcomer shall be *c*onsistent and *p*ersistent. $\mathbf{RQ_1}$'s finding clearly demonstrated that decreasing trends, long strikes without contribution would hurt the chance of success. In addition, documentation contribution does not evidently help newcomers with their technical success in their early careers unless they make intensive contributions. These findings give some recommendations to early career OSS contributors on better regulating their contribution activities and improving the chance of technical success.

Third, we found that the ecosystem that a project belongs to does introduce certain variances on the relationships between the contribution patterns and technical success. For contributors who switch their contribution to another project of a different ecosystem, the contribution patterns they used for previous projects might not work in new projects, due to different ecosystems. For example, avoiding long strikes without contribution in Apache projects could increase the chance of success, while it might not work similarly in OpenStack or Python Data Science projects. Hence, when contributors switch to another ecosystem, they should explore the ecosystem-specific contribution patterns to achieve technical success. While a limitation of this paper is that what factors cause such ecosystem-specific differences have not yet been identified, prior literature provide some cues that ecosystem-level norms and practices may play role in determining these differences. Bogart *et al.* [20] found that ecosystems upheld diverse values and behavioral norms which shaped its contributors' activities. For example, compared with the Apache ecosystem, the Python Data Science is more diverse, interdisciplinary and has less constraints. Therefore, its developers may not need to adhere to specific dynamic contribution patterns to achieve high centrality, in contrast to those in the Apache ecosystem (see Table 4).

Lastly, in addition to the source code contributions, newcomers also often contribute to documentation. Our case study results identified some specific trajectories that might positively correlate with newcomers' technical success, even though we were unable to find sufficient evidence to explicitly support the role of documentation contribution in newcomers' early careers. This suggests potential opportunities for future research examining how dynamic code and documentation contributions jointly influence newcomers' careers and success in open source projects through novel research designs. The potential correlations we observed could be starting points and anchors for these future studies.

Our work, particularly the recognition of the importance of early career contribution behaviors, suggests rich opportunities for future theory development and empirical research. In addition to those we have mentioned above, there are still

many open research questions that could be added to a future research agenda. For instance, we identified the correlations between dynamic contribution patterns in early career and technical success, but have not yet explained why there are these correlations. It is an inherent restriction of the exploratory methodology used in this paper. Researchers may employ confirmatory methodology to clarify and explain the reasons behind these correlations. These correlations may take effect jointly with external factors. Also, the results indicated that these relationships varied according to ecosystems. Thus, future work may include investigating how such differences relate to ecosystem-level characteristics such as practices, values, and so on [19], [20]. These differences could be potential factors that influence the contribution patterns in difference ecosystems, which worth investigating in future research. Besides, the technical success is not the only type of success in an OSS project; one might be organizationally successful, or politically, without a high centrality, and newcomers could choose various career pathways to achieve different types of success in OSS projects [21]. Do these types of success in OSS correlated with their behavioral patterns?

7.2 Design Implications

The study brings out opportunities for computational tools and mechanism designs to help newcomers in their early careers. We are going to give two examples of them.

According to our findings, one simple yet effective intervention may help newcomers keep track of their contribution patterns. The heatmap visualization of commit activity on a developer's profile may not fully satisfy such a purpose; it neither provides granularity at the project level nor has any analytic facilities to offer insights about dynamic patterns [86]. Our work suggests a potential visual design space of incorporating the visual elements reflecting contribution patterns such as general trends, the average amount of contribution, contribution frequency, and others. Also, visualizations could facilitate newcomers' learning from role models in their projects. Many newcomers have no idea about how to start and contribute when joining the project [3]. Visualizing and summarizing successful contributors' contribution trajectories in the same projects or the projects within the same ecosystem may help newcomers learn the possible ways to increase the chances to be a developer with higher technical impact in open source projects.

Besides, another effective intervention could be the reminder mechanism. A simple reminder mechanism could facilitate newcomers to maintain consistency in contributing to OSS projects. Our findings suggest maintaining a consistent contributing rhythm is significantly correlated with a newcomer's technical success. Once detecting some ongoing long inactive period in a newcomer's contributing trajectory, such a reminder mechanism could remind them by saying: "Hey, you haven't committed any code to [project name] since last month, would you like to check out what is happening there." It could also integrate sophisticated dynamic contribution forecasting techniques, for example, ARIMA-related techniques or SE-specialized GAP model [87], to enable some early intervention for preventing newcomers' inactivity in advance. Doing so could help newcomers maintain consistent and persistent code contributions and even bring some back on track. Moreover, if seeking to make such messages more noticeable, the visualization of daily activities, similar to what we show in Fig. 6, could replace GitHub's heatmap style visualization of activities in a user's profile page.

7.3 Recommendations to Newcomers

Our findings and corresponding discussions above can be distilled into immediate recommendations for newcomers to help them achieve technical success in their early careers. The main recommendation is that a newcomer shall be persistent to make regular contributions at some consistent levels during their early career, no matter for source code or documentation (like Newcomer A in Fig. 6). To be specific, a newcomer might need to bear the following in mind:

- *Keeping a smooth rhythm may be important (Section 5.2.1, 5.2.2);*
- *Try to contribute frequently but not abruptly (Section 5.2.3, 5.2.4);*
- *Avoid drastic decreases and long inactive periods (Section 5.2.3);*
- *Contributing a lot in a single day does not necessarily help too much (Section 5.2.4);*
- *Keep an eye on others' contribution patterns, particularly to learn from those who are in the same ecosystems (Section 5.3.2);*
- *Making intensive contributions, even when you focus on documentation contributions. (Section 6).*

Note that practicing the above recommendations is neither sufficient nor necessary to guarantee a newcomer's technical success. However, in general, adopting the above recommendations should improve the chance of achieving technical success in the long run.

7.4 Threats to Validity

We aimed to investigate fine-grained contribution patterns and technical success in OSS projects. Thus, it is reasonable to conduct an empirical study by analyzing digital trace data [88]. However, as in any empirical study, there are limitations associated with the particular study design.

From the perspective of *c*onstruct validity, our study involves two classes of primary constructs–contribution patterns encoded in time series features and technical success measured by a developer's centrality. All their definitions and operationalizations are straightforward and based on extant literature [17], [73], [78]. We removed hundreds of high-order time series features that could not link to behavioral patterns, thus ensuring direct interpretations between contribution patterns and time series features. However, we recognize that measuring technical success is a complex task and one that may have some alternative metrics. The construct validity would be enhanced with replications using other metrics in the future.

From the perspective of *i*nternal validity, the study shares the same set of limitations with other empirical work using repository data generated in software development. We took multiple measures to ensure that the data collection and preparation were free of most of the perils summarized in [89]. For example, we spent a fair amount of effort resolving the name entities and carefully distinguishing code contributions from other types of contributions. The

analysis process was done with mature statistical techniques and performed in an unbiased way.

From the perspective of *e*xternal validity, a significant threat in any empirical research is about the generalizability of results. Although the sampled ecosystems and corresponding projects represented a broad spectrum of OSS ecosystems and projects, it is far from representing the whole OSS universe. Thus, the results and findings may not be directly generalizable to other ecosystems and projects.

From the perspective of *c*onclusion validity, we have high-level confidence in the identified relationships between dynamic contribution patterns and newcomers' technical success in their early careers since we ensured the assumptions of multivariate analysis were satisfied, used HLM to control individual project's impacts on variances, and performed necessary model diagnosis. The model only contains one grouping level variable, i.e., which project a newcomer belonged to; thus, there would be no multicollinearity at the grouping level. The statistical models all had high statistical power. In the complementary case study, we did not perform any statistical modeling to identify potential relationships. Instead, we focused on qualitative observation and reasoning. Though documentation contribution might also potentially correlate to technical success, we acknowledged that such correlations could not be confirmed based on the current dataset.

8 Concluding Remarks

This paper reports on a study focusing on newcomers' dynamic contribution patterns during their early career, along with their technical success in OSS projects. Using fine-grained newcomers' contribution data from 54 large OSS projects from three ecosystems, we identified correlations between newcomers' contribution patterns (represented by time series features) and their centrality. The results confirm the existence of the correlations between newcomers' contribution patterns and their centrality. In general, being consistent and persistent in contribution is positively associated with newcomers' technical success measured by their centrality. While these correlations generally exist in all three ecosystems, we also observed some differences in detailed contribution patterns correlated with the centrality across ecosystems. Furthermore, we studied newcomers' documentation contribution as a complementary case study, to gain some insights about the effect of documentation contribution in newcomers' early career technical success. We discussed the implications and summarized practical recommendations to OSS newcomers.

To the best of our knowledge, this is the first research inquiry focusing on newcomer's dynamic technical contribution patterns in their early career. It brings out rich future opportunities related to the novel focus of the dynamic patterns of newcomers' contributions to OSS projects during their early career beyond onboarding processes. For instance, why and how do these dynamic contribution patterns influence newcomers' technical success? Do they also relate to other types of success, e.g., social success? How these dynamic contribution patterns impact one's career with other factors? Our present work will motivate researchers to pay more attention to develop a holistic understanding of newcomers' careers beyond their onboarding.

Acknowledgments

We would like to thank the editor and anonymous reviewers for their insightful comments and suggestions.

References

[1] G. von Krogh and E. von Hippel, "Special issue on open source software development," *Res. Policy*, vol. 32, no. 7, pp. 1149–1157, 2003.
[2] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/libre open-source software development: What we know and what we do not know," *ACM Comput. Surv.*, vol. 44, no. 2, pp. 7:1–7:35, 2012.
[3] I. Steinmacher, I. S. Wiese, T. Conte, M. A. Gerosa, and D. F. Redmiles, "The hard life of open source software project newcomers," in *Proc. 7th Int. Workshop Cooperative Hum. Aspects Softw. Eng.*, 2014, pp. 72–78.
[4] I. Steinmacher, T. Conte, M. A. Gerosa, and D. F. Redmiles, "Social barriers faced by newcomers placing their first contribution in open source software projects," in *Proc. 18th ACM Conf. Comput. Supported Cooperative Work Social Comput.*, 2015, pp. 1379–1392.
[5] I. Steinmacher, M. A. G. Silva, M. A. Gerosa, and D. F. Redmiles, "A systematic literature review on the barriers faced by newcomers to open source software projects," *Inf. Softw. Technol.*, vol. 59, pp. 67–85, 2015.
[6] I. Steinmacher, M. A. Gerosa, T. U. Conte, and D. F. Redmiles, "Overcoming social barriers when contributing to open source software projects," *Comput. Support. Cooperative Work.*, vol. 28, no. 1-2, pp. 247–290, 2019.
[7] I. Steinmacher, T. U. Conte, C. Treude, and M. A. Gerosa, "Overcoming open source project entry barriers with a portal for newcomers," in *Proc. 38th Int. Conf. Softw. Eng.*, 2016, pp. 273–284.
[8] I. Steinmacher, I. S. Wiese, A. P. Chaves, and M. A. Gerosa, "Why do newcomers abandon open source software projects," in *Proc. 6th Int. Workshop Cooperative Hum. Aspects Softw. Eng.*, 2013, pp. 25–32.
[9] A. De Vos, I. De Clippeleer, and T. Dewilde, "Proactive career behaviours and career success during the early career," *J. Occup. Organizational Psychol.*, vol. 82, no. 4, pp. 761–777, 2009.
[10] C. Kirchmeyer, "Demographic similarity to the work group: A longitudinal study of managers at the early career stage," *J. Organizational Behav.*, vol. 16, no. 1, pp. 67–83, 1995.
[11] S. A. Stumpf, "A longitudinal study of career success, embeddedness, and mobility of early career professionals," *J. Vocational Behav.*, vol. 85, no. 2, pp. 180–190, 2014.
[12] A. Hars and S. Ou, "Working for free? - Motivations of participating in open source projects," in *Proc. 34th Annu. Hawaii Int. Conf. Syst. Sci.*, 2001, pp. 25–31.
[13] I. Steinmacher, I. S. Wiese, and M. A. Gerosa, "Recommending mentors to software project newcomers," in *Proc. IEEE 3rd Int. Workshop Recommendation Syst. Softw. Eng.*, 2012, pp. 63–67.
[14] M. L. Savickas, "Career construction: A developmental theory of vocational behavior," in *LCareer Choice Develop.*, D. Brown, Ed., San Francisco, CA, USA: Jossey-Bass, 2002, pp. 149–205.
[15] S. E. Sullivan and Y. Baruch, "Advances in career theory and research: A critical review and agenda for future exploration," *J. Manage.*, vol. 35, no. 6, pp. 1542–1571, 2009.
[16] H.-G. Wolff and K. Moser, "Effects of networking on career success: A longitudinal study," *J. Appl. Psychol.*, vol. 94, no. 1, pp. 196–206, 2009.
[17] C. Jergensen, A. Sarma, and P. Wagstrom, "The onion patch: Migration in open source ecosystems," in *Proc. 19th ACM SIGSOFT Symp. Found. Softw. Eng., 13th Eur. Softw. Eng. Conf.*, 2011, pp. 70–80.
[18] D. G. Messerschmitt and C. Szyperski, *Software Ecosystem: Understanding an Indispensable Technology and Industry. Ser*, vol. 1. Cambridge, MA, USA: MIT Press, 2005.
[19] C. Bogart, C. Kästner, J. D. Herbsleb, and F. Thung, "How to break an API: Cost negotiation and community values in three software ecosystems," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2016, pp. 109–120.
[20] C. Bogart, C. Kästner, J. D. Herbsleb, and F. Thung, "When and how to make breaking changes: Policies and practices in 18 open source software ecosystems," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 4, pp. 42:1–42:56, 2021.

[21] B. Trinkenreich, M. Guizani, I. Wiese, A. Sarma, and I. Steinmacher, "Hidden figures: Roles and pathways of successful OSS contributors," *Proc. ACM Hum. Comput. Interact.*, vol. 4, no. CSCW2, pp. 180:1–180:22, 2020.
[22] D. Sholler, I. Steinmacher, D. Ford, M. Averick, M. Hoye, and G. Wilson, "Ten simple rules for helping newcomers become contributors to open projects," *PLoS Comput. Biol.*, vol. 15, no. 9, 2019, Art. no. e1007296.
[23] C. Casalnuovo, B. Vasilescu, P. T. Devanbu, and V. Filkov, "Developer onboarding in github: The role of prior social links and language experience," in *Proc. 10th Joint Meeting Found. Softw. Eng.*, 2015, pp. 817–828.
[24] J. Coelho, M. T. Valente, L. L. Silva, and A. C. Hora, "Why we engage in FLOSS: Answers from core developers," in *Proc. 11th Int. Workshop Cooperative Hum. Aspects Softw. Eng.*, 2018, pp. 114–121.
[25] F. Fronchetti, I. Wiese, G. Pinto, and I. Steinmacher, "What attracts newcomers to onboard on OSS projects? TL, DR: Popularity," in *ser. IFIP Adv. Inf. Commun. Technol.*, 2019, vol. 556, pp. 91–103.
[26] G. H. L. Pinto, F. F. Filho, I. Steinmacher, and M. A. Gerosa, "Training software engineers using open-source software: The professors' perspective," in *Proc. 30th IEEE Conf. Softw. Eng. Educ. Training*, 2017, pp. 117–121.
[27] H. S. Qiu, Y. L. Li, H. S. Padala, A. Sarma, and B. Vasilescu, "The signals that potential contributors look for when choosing open-source projects," *Proc. ACM Hum. Comput. Interact.*, vol. 3, no. CSCW, pp. 122:1–122:29, 2019.
[28] I. Steinmacher, A. P. Chaves, T. U. Conte, and M. A. Gerosa, "Preliminary empirical identification of barriers faced by newcomers to open source software projects," in *Proc. IEEE Braz. Symp. Softw. Eng.*, 2014, pp. 51–60.
[29] M. Zhou and A. Mockus, "Developer fluency: Achieving true mastery in software projects," in *Proc. 18th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, G. Roman and A. van der Hoek, Eds., 2010, pp. 137–146.
[30] S. Balali, I. Steinmacher, U. Annamalai, A. Sarma, and M. A. Gerosa, "Newcomers' barriers... is that all? An analysis of mentors' and newcomers' barriers in OSS projects," *Comput. Support. Cooperative Work.*, vol. 27, no. 3-6, pp. 679–714, 2018.
[31] G. Matturro, K. Barrella, and P. Benitez, "Difficulties of newcomers joining software projects already in execution," in *Proc. Int. Conf. Comput. Sci. Comput. Intell.*, 2017, pp. 993–998.
[32] M. Zhou and A. Mockus, "What make long term contributors: Willingness and opportunity in OSS community," in *Proc. 34th Int. Conf. Softw. Eng.*, 2012, pp. 518–528.
[33] G. Avelino, E. Constantinou, M. T. Valente, and A. Serebrenik, "On the abandonment and survival of open source projects: An empirical investigation," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas.*, 2019, pp. 1–12.
[34] E. Skriptsova *et al.* , "Analysis of newcomers activity in communicative posts on github," in *Digital Transformation Global Soc.*, D. A. Alexandrov, A. V. Boukhanovsky, A. V. Chugunov, Y. Kabanov, O. Koltsova, and I. Musabirov, Eds. Berlin, Germany: Springer, 2019, pp. 452–460.
[35] C. J. Mendez *et al.*, "Open source barriers to entry, revisited: A sociotechnical perspective," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 1004–1015.
[36] A. Lee and J. C. Carver, "FLOSS participants' perceptions about gender and inclusiveness: A survey," in *Proc. IEEE 41st Int. Conf. Softw. Eng.*, 2019, pp. 677–687.
[37] Z. Wang, Y. Wang, and D. F. Redmiles, "Competence-confidence gap: A threat to female developers' contribution on github," in *Proc. 40th Int. Conf. Softw. Eng.: Softw. Engineering Soc.*, 2018, pp. 81–90.
[38] S. H. Padala *et al.*, "How gender-biased tools shape newcomer experiences in oss projects," *IEEE Trans. Softw. Eng.*, vol. 48, no. 1, pp. 241–259, Jan. 2022.
[39] S. da Silva Amorim, J. D. McGregor, E. S. de Almeida, and C. von FlachG. Chavez, "Educating to achieve healthy open source ecosystems," in *Proc. 12th Eur. Conf. Softw. Archit.: Companion Proc.*, 2018, pp. 28:1–28:7.
[40] R. Britto, D. S. Cruzes, D. Smite, and A. Sablis, "Onboarding software developers and teams in three globally distributed legacy projects: A multi-case study," *J. Softw. Evol. Process.*, vol. 30, no. 4, 2018, Art. no. e1921.
[41] M. Zhou, *Onboarding and Retaining of Contributors in FLOSS Ecosystem*. New York, NY, USA: Springer, 2019.
[42] B. Strahm, C. M. Gray, and M. Vorvoreanu, "Generating mobile application onboarding insights through minimalist instruction," in *Proc. Designing Interact. Syst. Conf.*, 2018, pp. 361–372.
[43] C. Liu, D. Yang, X. Zhang, B. Ray, and M. M. Rahman, "Recommending github projects for developer onboarding," *IEEE Access*, vol. 6, pp. 52082–52094, 2018.
[44] G. Canfora, M. D. Penta, R. Oliveto, and S. Panichella, "Who is going to mentor newcomers in open source projects," in *Proc. 20th ACM SIGSOFT Symp. Found. Softw. Eng.* , 2012, pp. 1–11.
[45] I. Steinmacher, C. Treude, and M. A. Gerosa, "Let me in: Guidelines for the successful onboarding of newcomers to open source projects," *IEEE Softw.*, vol. 36, no. 4, pp. 41–49, Jul./Aug. 2019.
[46] G. C. Diniz, M. A. G. Silva, M. A. Gerosa, and I. Steinmacher, "Using gamification to orient and motivate students to contribute to OSS projects," in *Proc. 10th Int. Workshop Cooperative Hum. Aspects Softw. Eng.*, 2017, pp. 36–42.
[47] C. Toscani, D. Gery, I. Steinmacher, and S. Marczak, "A gamification proposal to support the onboarding of newcomers in the flosscoach portal," in *Proc. 17th Braz. Symp. Hum. Factors Comput. Syst.*, 2018, pp. 1:1–1:10.
[48] A. Sarma, M. A. Gerosa, I. Steinmacher, and R. Leano, "Training the future workforce through task curation in an OSS ecosystem," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2016, pp. 932–935.
[49] J. Wang and A. Sarma, "Which bug should i fix: Helping new developers onboard a new project," in *Proc. 4th Int. Workshop Cooperative Hum. Aspects Softw. Eng.*, 2011, pp. 76–79.
[50] A. Lee, J. C. Carver, and A. Bosu, "Understanding the impressions, motivations, and barriers of one time code contributors to FLOSS projects: A survey," in *Proc. IEEE 39th Int. Conf. Softw. Eng.*, 2017, pp. 187–197.
[51] G. von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: A case study," *Res. Policy*, vol. 32, no. 7, pp. 1217–1241, 2003.
[52] C. Hannebauer and V. Gruhn, "On the relationship between newcomer motivations and contribution barriers in open source projects," in *Proc. 13th Int. Symp. Open Collaboration*, 2017, pp. 2:1–2:10.
[53] L. A. Dabbish, H. C. Stuart, J. Tsay, and J. D. Herbsleb, "Social coding in github: Transparency and collaboration in an open software repository," in *Proc. ACM Conf. Comput. Supported Cooperative Work*, 2012, pp. 1277–1286.
[54] I. Steinmacher, M. A. Gerosa, and D. F. Redmiles, "Attracting, onboarding, and retaining newcomer developers in open source software projects," in *Proc. Workshop Glob. Softw. Develop. Perspective*, 2014, pp. 1–4.
[55] G. Gousios, M. D. Storey, and A. Bacchelli, "Work practices and challenges in pull-based development: The contributor's perspective," in *Proc. 38th Int. Conf. Softw. Eng.*, 2016, pp. 285–296.
[56] F. Sarker, B. Vasilescu, K. Blincoe, and V. Filkov, "Socio-technical work-rate increase associates with changes in work patterns in online projects," in *Proc. IEEE 41st Int. Conf. Softw. Eng.*, 2019, pp. 936–947.
[57] F. Calefato, G. Iaffaldano, F. Lanubile, and B. Vasilescu, "On developers' personality in large-scale distributed projects: The case of the apache ecosystem," in *Proc. 13th Conf. Glob. Softw. Eng.*, 2018, pp. 92–101.
[58] C. Jensen and W. Scacchi, "Role migration and advancement processes in OSSD projects: A comparative case study," in *Proc. 29th Int. Conf. Softw. Eng.*, 2007, pp. 364–374.
[59] C. Miller, D. G. Widder, C. Kästner, and B. Vasilescu, "Why do people give up flossing? a study of contributor disengagement in open source," in *Proc. Open Source Syst. - 15th IFIP WG 2.13 Int. Conf.*, 2019, vol. 556, pp. 116–129.
[60] P. van Wesel, B. Lin, G. Robles, and A. Serebrenik, "Reviewing career paths of the openstack developers," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol.*, 2017, pp. 544–548.
[61] T. N. Bauer and S. G. Green, "Effect of newcomer involvement in work-related activities: A longitudinal study of socialization." *J. Appl. Psychol.*, vol. 79, no. 2, pp. 211–223, 1994.
[62] T. N. Bauer, S. Perrot, R. C. Liden, and B. Erdogan, "Understanding the consequences of newcomer proactive behaviors: The moderating contextual role of servant leadership," *J. Vocational Behav.*, vol. 112, pp. 356–368, 2019.
[63] A. S. Gabriel, J. M. Diefendorff, A. A. Bennett, and M. D. Sloan, "It's about time: The promise of continuous rating assessments for the organizational sciences," *Organizational Res. Methods*, vol. 20, no. 1, pp. 32–60, 2017.

[64] K. J. Jansen and A. J. Shipp, "Fitting as a temporal sensemaking process: Shifting trajectories and stable themes," *Hum. Relations*, vol. 72, no. 7, pp. 1154–1186, 2019.
[65] T. W. Lee, P. W. Hom, M. B. Eberly, J. J. Li, and T. R. Mitchell, "On the next decade of research in voluntary employee turnover," *Acad. Manage. Perspectives*, vol. 31, no. 3, pp. 201–221, 2017.
[66] C. R. Wanberg, T. M. Glomb, Z. Song, and S. Sorenson, "Job-search persistence during unemployment: A 10-wave longitudinal study." *J. Appl. Psychol.*, vol. 90, no. 3, pp. 411–430, 2005.
[67] A. J. Shipp and M. S. Cole, "Time in individual-level organizational studies: What is it, how is it used, and why isn't it exploited more often?," *Annu. Rev. Organ. Psychol. Organ. Behav.*, vol. 2, no. 1, pp. 237–260, 2015.
[68] S. Sonnentag, "Time in organizational research: Catching up on a long neglected topic in order to improve theory," *Organizational Psychol. Rev.*, vol. 2, no. 4, pp. 361–368, 2012.
[69] D. Spadini, M. F. Aniche, and A. Bacchelli, "Pydriller: Python framework for mining software repositories," in *Proc. ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Foundations Softw. Eng.*, 2018, pp. 908–911.
[70] M. Golzadeh, A. Decan, D. Legay, and T. Mens, "A ground-truth dataset and classification model for detecting bots in github issue and PR comments," *J. Syst. Softw.*, vol. 175, 2021, Art. no. 110911.
[71] S. Chacon and B. Straub, *Pro Git*. New York, NY, USA: Springer, 2014.
[72] G. Pinto, I. Steinmacher, and M. A. Gerosa, "More common than you think: An in-depth study of casual contributors," in *Proc. IEEE 23rd Int. Conf. Softw. Anal., Evol., Reeng.*, 2016, pp. 112–123.
[73] M. Christ, A. W. Kempa-Liehr, and M. Feindt, "Distributed and parallel time series feature extraction for industrial Big Data applications," *Comput. Res. Repository*, 2016, *arXiv:abs/1610.07717*.
[74] P. Bonacich, "Some unique properties of eigenvector centrality," *Social Netw.*, vol. 29, no. 4, pp. 555–564, 2007.
[75] E. v. Hippel and G. v. Krogh, "Open source software and the 'private-collective' innovation model: Issues for organization science," *Org. Sci.*, vol. 14, no. 2, pp. 209–223, 2003.
[76] P. V. Singh, "The small-world effect: The influence of macro-level properties of developer collaboration networks on open-source project success," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 2, pp. 1–27, 2010.
[77] P. V. Singh, Y. Tan, and V. Mookerjee, "Network effects: The influence of structural capital on open source project success," *MIS Quart.*, pp. 813–829, 2011.
[78] M. Joblin, S. Apel, C. Hunsen, and W. Mauerer, "Classifying developers into core and peripheral: An empirical study on count and network metrics," in *Proc. 39th Int. Conf. Softw. Eng.*, 2017, pp. 164–174.
[79] H. C. Gall, K. Hajek, and M. Jazayeri, "Detection of logical coupling based on product release history," in *Proc. Int. Conf. Softw. Maintenance*, 1998, pp. 190–197.
[80] P. Bonacich, "Power and centrality: A family of measures," *Amer. J. Sociol.*, vol. 92, no. 5, pp. 1170–1182, 1987.
[81] B. Lin and A. Serebrenik, "Recognizing gender of stack overflow users," in *Proc. 13th Int. Conf. Mining Softw. Repositories*, M. Kim, R. Robbes, and C. Bird, Eds., 2016, pp. 425–429.
[82] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.
[83] S. W. Raudenbush and A. S. Bryk, *Hierarchical Linear Models: Applications and Data Analysis Methods*. Thousand Oaks, CA, USA: Sage, 2002.
[84] J. Terrell *et al.*, "Gender differences and bias in open source: Pull request acceptance of women versus men," *PeerJ Comput. Sci.*, vol. 3, 2017, Art. no. e111.
[85] R. Xu, "Measuring explained variation in linear mixed effects models," *Statist. Med.*, vol. 22, no. 22, pp. 3527–3541, 2003.
[86] Y. Park and C. Jensen, "Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers," in *Proc. 5th IEEE Int. Workshop Visual. Softw. Understanding Anal.*, 2009, pp. 3–10.
[87] A. Decan, E. Constantinou, T. Mens, and H. Rocha, "GAP: Forecasting commit activity in git projects," *J. Syst. Softw.*, vol. 165, 2020, Art. no. 110573.
[88] C. S. Østerlund, K. Crowston, and C. B. Jackson, "Building an apparatus: Refractive, reflective, and diffractive readings of trace data," *J. Assoc. Inf. Syst.*, vol. 21, no. 1, pp. 1–22, 2020.
[89] J. Howison and K. Crowston, "The perils and pitfalls of mining sourceforge," in *Proc. 1st Int. Workshop Mining Softw. Repositories*, 2004, pp. 7–11.

Yang Yue received the BEng degree in software engineering from East China Normal University in 2015, and the MEng degree in software engineering from Peking University in 2018. He is currently working toward the PhD degree in software engineering with the University of California, Irvine. His research interests include human and social factors in open source software development, particularly, OSS ideology, and career development in open source projects.

Yi Wang received the PhD degree from the University of California, Irvine, in 2015. He is currently a professor with the School of Computer Science (National Pilot Software Engineering School) and the Key Laboratory of Trustworthy Distributed Computing and Service (MoE), Beijing University of Posts and Telecommunications, China. His research interests include human and social factors in software engineering, computer-supported cooperative work (CSCW), and social computing.

David Redmiles (Member, IEEE) received the PhD degree in computer science from the University of Colorado, Boulder, in 1992. He is currently a professor with the Department of Informatics, Donald Bren School of Information and Computer Sciences, University of California, Irvine. His research interests include software engineering, human-computer interaction, and computer-supported cooperative work. Over the years, his research group has investigated themes of cognitive support for software developers, issues of trust and emotion affecting software teams, behaviors of participants in social software development platforms, global software engineering, and end-user software development. He is a member of ACM and IEEE Computer Society. He was designated an ACM distinguished scientist in 2011 and a fellow of Automated Software Engineering in 2009.