Understanding Work Rhythms in Software Development and Their Effects on Technical Performance

Zhang, Jiayun; Gong, Qingyuan; Chen, Yang; Xiao, Yu; Wang, Xin; Ding, Aaron Yi

DOI
10.1049/2024/8846233

Publication date
2024

Document Version
Final published version

Published in
IET Software

Wiley
IET Software
Volume 2024, Article ID 8846233, 19 pages
https://doi.org/10.1049/2024/8846233

IET The Institution of Engineering and Technology | WILEY

Research Article

Understanding Work Rhythms in Software Development and Their Effects on Technical Performance

Jiayun Zhang,[1] **Qingyuan Gong**,[2] **Yang Chen**,[1] **Yu Xiao**,[3] **Xin Wang**,[1] **and Aaron Yi Ding**[4]

[1]*Shanghai Key Lab of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai, China*
[2]*Research Institute of Intelligent Complex Systems, Fudan University, Shanghai, China*
[3]*Department of Information and Communications Engineering, Aalto University, Espoo, Finland*
[4]*Department of Engineering Systems and Services, Delft University of Technology, Delft, Netherlands*

Correspondence should be addressed to Qingyuan Gong; gongqingyuan@fudan.edu.cn and Yang Chen; chenyang@fudan.edu.cn

Received 14 June 2023; Revised 30 April 2024; Accepted 15 May 2024

Academic Editor: Alessandro Marchetto

The temporal patterns of code submissions, denoted as work rhythms, provide valuable insight into the work habits and productivity in software development. In this paper, we investigate the work rhythms in software development and their effects on technical performance by analyzing the profiles of developers and projects from 110 international organizations and their commit activities on GitHub. Using clustering, we identify four work rhythms among individual developers and three work rhythms among software projects. Strong correlations are found between work rhythms and work regions, seniority, and collaboration roles. We then define practical measures for technical performance and examine the effects of different work rhythms on them. Our findings suggest that moderate overtime is related to good technical performance, whereas fixed office hours are associated with receiving less attention. Furthermore, we survey 92 developers to understand their experience with working overtime and the reasons behind it. The survey reveals that developers often work longer than required. A positive attitude towards extended working hours is associated with situations that require addressing unexpected issues or when clear incentives are provided. In addition to the insights from our quantitative and qualitative studies, this work sheds light on tangible measures for both software companies and individual developers to improve the recruitment process, project planning, and productivity assessment.

1. Introduction

The time allocation for work activities is closely related to a software developer's daily routine and reflects her/his work habits. We define the work rhythms in the process of software development as the temporal patterns shown in developers' code submission activities. A typical work rhythm of a developer could be described as follows: the developer may start the work at 9 a.m. on working days and concentrate on writing and submitting code during working hours. She/he would take a short break at noon for lunch and the code submissions could stop for a while as well. After finishing the tasks at 6 p.m., the codes will not be updated until 9 a.m. on the next working day. Developers working in companies with diverse cultures follow different work rhythms. It was reported that one third of software developers do not adopt a typical working hour rhythm (e.g., from 10 a.m. to 6 p.m.) [1]. The issues of developers' work rhythms have been discussed extensively. Some Chinese tech companies have adopted an unofficial work schedule known as the "996 working hour system," which requires employees to work from 9 a.m. to 9 p.m., 6 days a week. The public quickly took notice of these extreme working hours as they were shared on social media (https://github.com/996icu/996.ICU). This abnormal work schedule has received criticism, arguing that developers cannot keep focusing on programming during such long working hours and their efficiency and productivity decrease after working for long hours (https://www.scmp.com/tech/start-ups/article/3005947/quantity-or-quality-chinas-996-work-culture-comes-under-scrutiny). However, global leading news media, such as Cable News Network (CNN; https://edition.cnn.com/2019/04/15/business/jack-ma-996-china/index.html) and British Broadcasting Corporation (BBC) News (https://www.bbc.com/news/business-47934513), reported another voice that many successful entrepreneurs

weighed on the advantages of long-hour work schedules to the companies. These heated discussions with controversial perspectives press an urge demand to understand developers' work rhythms and their effects on practical technical performance.

Studying work rhythms in software development yields many important implications. For example, the profiles and activities in online developer communities are considered as reliable indicators of technical performance during the hiring process [2]. However, having more commits during off-hours does not necessarily equate to better code quality. Instead of assessing based on the quantity of commits, it is crucial to acquire a deeper understanding of work rhythms and their effects. Such insights can help employers gain deeper knowledge about job applicants' work habits before hiring. In addition, software development teams can rely on more rational assessments of technical performance rather than judging merely by the time spent in the office. With an understanding of the effects of work rhythms on technical performance, both project teams and individual developers can better allocate and schedule their time in development.

The existing studies on the work rhythms of people in different occupations often cover their effects on work performance. Alternative work schedules, such as flexible and compressed work schedules, had positive effects on work-related criteria including productivity and job satisfaction [3, 4]. Conversely, sustained work during long working hours was associated with an increased risk of errors and decreased work performance [5, 6, 7, 8, 9]. In the field of software engineering, multiple studies have examined the relationship between code quality and the time when the work is performed. It has been found that the bugginess of commits is related to the time (i.e., the hour of the day) when those commits have been made, but there are large variations among individuals and projects [10, 11, 12].

Previous studies have primarily focused on the effects of work hours on code quality, within the contexts of limited organizations and have primarily considered code bugginess as a quality metric. In addition, they have not sufficiently addressed the circadian and weekly patterns that characterize developers' work habits. Our study leverages a large-scale real-world dataset from GitHub to explore how work rhythms correlate with multiple dimensions of technical performance. Considering that project-level working behaviors often involve collaborative efforts of multiple contributors and do not necessarily reflect the work patterns of individual developers, our study analyzes both project- (in our study, the term "project" is used synonymously with "repository") and individual-level metrics. We aim to provide a more comprehensive understanding of work patterns from two different yet interconnected perspectives. Specifically, we apply spectral biclustering [13] to identify the work rhythms from both the individual and project perspectives. The biclustering algorithm simultaneously groups both rows and columns of a data matrix, allowing us to understand the groups of similar subjects (i.e., developers/repositories) and their typical commit behaviors at the same time. We analyze the relationship between the identified work rhythms and demographics (such as region and account/repository age) and collaboration roles (i.e., whether a developer is a structural hole spanner (SHS) [14]). We use popularity metrics (such as followers, stars, forks, and issues on GitHub) and code productivity (measured by lines of code changed per week) as indicators of technical performance. Then, we perform a comprehensive analysis to investigate how these work rhythms influence technical performance. Furthermore, we conduct a survey study to complement the results of empirical data analysis.

Our major contributions are summarized as follows:

(1) We design an approach with spectral biclustering algorithm to identify the work rhythms of repositories and individual developers. This method reveals four distinct work rhythms among individuals and three among repositories.
(2) We present an empirical analysis of the correlations between work rhythms and demographics including regions, age, and collaboration roles. We define multiple practical measures for technical performance and study the effects of work rhythms on them.
(3) We conduct a survey involving 92 respondents to gain insights into developers' experiences and the reasons and attitudes towards overtime work.

We introduce the background and related works in Section 2 and research questions in Section 3, followed by our research methods (Section 4) and results (Section 5). We discuss the significance of our contributions in Section 6 and offer some concluding remarks in Section 7.

2. Background and Related Work

Developers are engaged in multiple work activities in a given week and follow some rules in the time usage in software development [15, 16, 17]. Sequential analysis of the generated contents is crucial for understanding the behavior patterns of online users [18, 19]. The widely used development tools such as version control systems and online developer communities ensure the transparency of the workflows, which provide researchers with abundant resources to investigate developers' work practices [20, 21, 22, 23]. By exploring the data from these development tools, multiple studies have examined developers' work practices and contributions.

First, the work time in software development has been studied. For example, Claes et al. [1] defined work rhythm as the circadian and weekly patterns of commits. They analyzed the commit timestamps of 86 open source software projects and reported that two-thirds of the developers follow a standard work schedule and rarely work nights and weekends. In addition, Traulle and Dalle [24] investigated the evolution of developers' work rhythms. They observe a trend where developers adopt more regular work patterns over time and start working increasingly earlier. Furthermore, this study is related to our previous work [25], which examined the commit activities of tech companies in China and the United States and compared the differences in working hours between companies in the two countries. Compared with our previous work, this study expands the scope and introduces new research questions—the correlations between

work rhythm and technical performance. In addition, we enlarge the dataset to include a wider range of regions and approach the analysis of working behaviors at more granular levels by examining both project- and individual-level behaviors.

Second, the relationships between work quality and work time have been investigated. For example, Khomh et al. [26] studied the impact of Firefox's rapid release cycle on software quality. They found that the fast release cycle did not lead to more bugs but accelerated the process of fixing bugs. In addition, several studies focused on the relationships between the bugginess of code and the hour of the day when the code is submitted. For instance, Eyolfson et al. [10, 11] studied three well-known open source projects and found that more bugs are contained in commits made during midnight and early morning, while commits made in the morning have the best quality. Prechelt and Pepper [12] investigated a closed-source industry project and proposed that 8 p.m. is the hour with the highest error rate. It is observed that results vary across different projects.

Previous research on the effects of work time often investigates projects from limited organizations and only considers the bugginess of code as the metric of code quality. In addition, these studies typically focus on the effects of specific hours of the day, rather than the circadian and weekly patterns. There is no sufficient investigation with solid evidence yet to show the relationship between work rhythms and technical performance from multiple aspects. In this paper, we perform data analysis on a real-world code submission dataset collected from GitHub, a prominent online developer platform with more than 100 million developers and hosting more than 420 million repositories (https://github.com/about, accessed on May 18, 2024).

During the software development, people often use Git, a distributed version control system, to monitor the modifications to the code. To submit code changes to Git, people make *commits* that include details such as authorship, timestamp, and the code changes made. The temporal distribution of a developer's commit logs reflects her/his rhythm of submitting code changes. These commit logs can be accessed if the projects are uploaded to GitHub and set to publicly visible. Figure 1 shows the time distribution of developers' code submissions on GitHub. The statistics are generated according to the GitHub User Dataset [27, 28]. The dataset consists of the information and activities about more than 10 million randomly selected GitHub users. We focus on the users who have more than 100 commits and have submitted codes on more than 100 different days. Among these users, we select 13,201 of the developers with 5,406,933 commits. In general, developers commit more frequently on weekdays than at weekends. There are peak hours of code submissions at 11 a.m., 4 p.m., and 10 p.m., and an off-peak period during the early morning, which conforms to the common sense of people's daily life. The aggregated commit logs in Figure 1 show that developers exhibit temporal regularities in code submissions. However, given the differences in the adoption of work practices, such general work rhythm could not represent effectively the work habit of each developer.

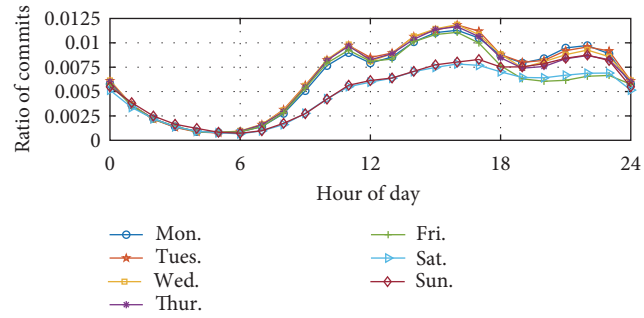

FIGURE 1: Time distribution of code submissions on GitHub. The *x*-axis shows the hour of the day, with both 0 and 24 representing 12 a.m. The *y*-axis shows the ratio of commits made within an hour to the total number of commits.

3. Research Questions

We aim to study the work rhythms of developers and software projects to have a comprehensive view of work rhythms in software development from both the individual and group levels. Our study is guided by the following four research questions:

RQ1. What are the work rhythms of individual developers and software projects?

RQ2. Are work rhythms related to demographics and collaboration roles?

The first two RQs intend to reveal representative work rhythms among individual developers and software projects and examine discrepancies in the demographics of the developers with different work rhythms.

RQ3. What are the correlations between different work rhythms and technical performance?

The third RQ is to seek a deeper understanding of the relationships of different work rhythms with the outcome of work by considering various metrics for technical performance.

RQ4. What are developers' attitudes towards work rhythms and productivity?

The last RQ investigates developers' actual work experience and their views on productivity.

4. Methods

In this section, we present the data collection and analysis methods in our study. A summary of the research subjects, variables, and the methods of data analysis for each research question is provided in Table 1. The overview of the methodology is presented in Figure 2.

4.1. Data Collection. The commit logs of public projects on GitHub are publicly visible and can be retrieved using the GitHub API. Our data collection adhered to "terms of service" of GitHub (https://help.github.com/articles/github-terms-of-service/). The data collection took place from May 1 to May 27, 2019. The dataset covers the commit activities of the source repositories of 110 organizations ever since the repositories were created. The location of the companies spread a wide range from the United States (such as Facebook, Amazon, and Google) to China (such as Baidu,

Table 1: Summary of research subjects, variables, and methods of analysis applied to research questions.

Research question	Subject	Variable	Analysis method
RQ1	Developer/repository	Commit frequency during the week	Spectral biclustering
RQ2	Developer	Account creation time	Mann–Whitney *U* test
		Structral hole spanner	APGreedy, Pearson's *chi*-squared test
	Repository	Regions	Pearson's *chi*-squared test
		Repository creation time	Mann–Whitney *U* test
RQ3	Developer	Number of followers	Mann–Whitney *U* test
		Average number of stars	
		h-index of stars	
	Repository	Number of stars	Mann–Whitney *U* test
		Number of forks	
		Number of open issues	
		Lines of code changed per week	
RQ4	Developer	Required and actual working hours	User study
		Time allocation for work activities	
		Attitude towards working overtime	

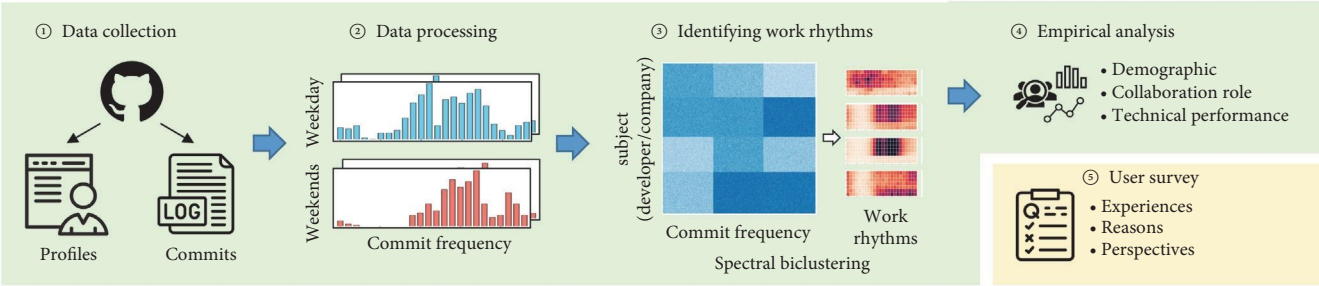

Figure 2: The workflow of our study. We collect profiles of developers and projects, along with their commit activities, from 110 organizations on GitHub. Data processing is performed on these commits, which are then used to identify work rhythms via spectral biclustering. We conduct an empirical analysis of the demographics, collaboration roles, and technical performances across these identified work rhythms using hypothesis testing. Furthermore, we administer a user survey to understand developers' attitudes on work rhythms and productivity.

Tencent, and Alibaba) and Europe (such as SAP, Nokia, and Spotify). To accurately assess work rhythms, we used the local time of each commit log to avoid the potential influence of different time zones in which the commits were made. Commit logs without time zone information (9.03% of the total) were excluded. Following the data cleaning, a total of 1,532,439 commits remained. Then, we group these commits by repositories and committers respectively and form the following two datasets for our analysis.

Company repositories. We scanned the repository lists of the 110 organization accounts and crawled descriptive information about the repositories and commit logs submitted into the repositories. We selected repositories with at least 300 commits and formed the *repository dataset* with a total of 1,131 repositories and 1,111,685 commits.

Individual developers. To study the work rhythms of individual developers, we first merged different identities of the same developer, as a developer may have multiple identities on GitHub and in the version control system. We extracted the email from the version control system's author field and GitHub account ID from the author field recorded in GitHub commit activity. We created a mapping from email addresses to GitHub accounts and grouped together identities that shared the same account ID or email address. Following this dealiasing process, 47.1% of the committer identities were merged. Then, we chose the core developers by selecting those with at least 30 commits. These developers are the top 12.5% of the committers and have made 85% out of all commits in our dataset. We further crawl the GitHub account information of the developers, including number of followers and number of stars in each of their own repositories. Finally, we formed our *developer dataset* with 7,509 individual developers and 1,296,715 commits, among which 2,754 have detailed information about their GitHub accounts.

4.2. Identifying Work Rhythms. To profile how commits are created by a developer or in a project repository, we compute the frequencies of commit activities across different time intervals and apply clustering to identify patterns.

4.2.1. Data Processing. For each developer or repository, we calculate the average percentage of commits for each hour of the day on both weekdays and weekends. Formally, we

denote the commit logs of a repository or a developer as $L = \{c_1, c_2, \ldots, c_H\}$, where c_h is the hth commit and H is the total number of commits. We segment a week into 168 hr (24 hr per day) and count the number of commits made in each hour as $N = \{n_1, n_2, \ldots, n_{168}\}$. To reduce noise in the empirical data, we follow Goyal et al.'s [29] method to take a 3-hr average and divide it by the total number of commits to obtain the commit frequency for each hour:

$$f_t = \begin{cases} \dfrac{n_{t-1} + n_t + n_{t+1}}{3 \times \sum_{i=1}^{168} n_i}, & 2 \le t \le 167, t \in N \\ \dfrac{n_{168} + n_1 + n_2}{3 \times \sum_{i=1}^{168} n_i}, & t = 1 \\ \dfrac{n_{167} + n_{168} + n_1}{3 \times \sum_{i=1}^{168} n_i}, & t = 168 \end{cases} . \quad (1)$$

Then, for each hour of the day, we compute the average commit frequency in that hour on a weekday and a weekend, as shown in Equations (2) and (3) respectively:

$$\bar{f}_h^{\text{weekday}} = \frac{1}{5}\left(f_h^1 + f_h^2 + f_h^3 + f_h^4 + f_h^5\right), \quad (2)$$

$$\bar{f}_h^{\text{weekend}} = \frac{1}{2}\left(f_h^6 + f_h^7\right), \quad (3)$$

where h denotes the hth hour of the day and f_h^d denotes the commit frequency of the hth hour of the day on the dth day of the week. Finally, the profile of a developer's commit behavior is represented as a 48-dimensional vector $\{\bar{f}_h^{d\text{type}} | h \in \{1, 2, \ldots, 24\}, d\text{type} \in \{\text{weekday}, \text{weekend}\}\}$.

4.2.2. Biclustering Model. Among various classical clustering methods, such as K-means [30], DBSCAN [31], and the state-of-the-art ones designed for specific applications such as topic models (latent Dirichlet allocation) [32, 33], we choose the spectral biclustering [13] algorithm to discover the work rhythms in our dataset. Spectral biclustering is a clustering technique, which generates biclusters—a group of samples (in row) that show similar behavior across a subset of features (in column), or vice versa. In our scenario, we group both developers/repositories and the commit behavior at a time to understand the groups of similar subjects and their typical behaviors. Specifically, developers/repositories grouped in different row clusters show different commit behaviors. In addition, the column clusters outputted by the algorithm enable us to infer how developers/repositories in different row clusters behave in each subset of hours. Developers/repositories with the same rhythm have similar commit frequencies in each subset of hours.

The model takes the 48-dimensional vectors as input and automatically discovers the clusters of work rhythms by measuring the similarities between them. To implement the clustering model, we used Scikit-learn [34], a widely used machine-learning library. To determine the optimal parameter setting, we perform an iterative search for the number of work rhythms k from 2 to 8 with empirical experiments. For each k, we visualize the rhythms and examine the number of samples in clusters to ensure that the clusters have sufficient individuals and exhibit distinct patterns beyond mere time shifting. We choose k as the largest value among those tested that yields stable and distinctive work rhythms.

4.3. Empirical Analysis on Identified Work Rhythms

4.3.1. Demographics of Developers and Repositories. We intend to explore whether developers or repositories with specific demographic information tend to follow specific work rhythms.

First, local cultures may have an impact on work rhythms. To investigate whether there is a difference among developers who work on repositories from different regions in terms of work rhythms, we examine the countries of the repositories that the developers worked on. For each developer, we group the repositories that she/he has made contributions to and check which countries the organizations of the repositories belong to. If a developer has contributions to repositories from more than one country, we set the work region of the developer as "multiple countries." We target four different regions: the United States, China, Europe, and multiple countries.

In addition, considering the fact that senior developers may take charge of more projects than junior developers, we assume that senior developers have different work rhythms from young developers. For this purpose, we investigate whether there is a correlation between the type of work rhythms and the seniority of the developers. We use the number of days after the creation time of GitHub account as a proxy for one's seniority in programming.

Furthermore, according to Vasilescu et al.'s [35] study, there are differences in terms of productivity between younger repositories and older ones. As a result, repositories with longer histories may have different work rhythms from newly created ones. We count the number of days since a project was created on GitHub as the measure of repository age.

4.3.2. Collaboration Role. Collaboration is an important feature of software engineering. The developer's participation in project collaboration is a testament to her/his technical ability.

The structural hole theory [14, 36, 37, 38] in social network analytics suggests that people who are positioned in structural holes, known as SHS, play a critical role in the collaboration and management of the teams. A structural hole is perceived as a gap between two closely connected groups. SHS fill in the gaps among different groups. They control the diffusion of valuable information across groups and come up with new ideas by combining ideas from multiple sources [14]. Bhowmik et al. [39] studied the role of structural holes in requirements identification of open-source software development and found that structural holes are positively related to the contribution of a larger amount of new requirements and play an important role in new requirement identification.

We intend to see whether there is a difference in terms of work rhythms between SHS developers and ordinary developers. We build a collaboration graph using our dataset, in which the node represents a developer and an edge between

two nodes represents the two developers have committed to the same repository. We apply an advanced SHS identification algorithm called APGreedy [40] (there are several SHS identification algorithms [37, 41, 42] and APGreedy is a representative one) to find the SHS in the collaboration graph and choose the top 500 developers as the SHS developers. After filtering out developers with less than 30 commits, we obtain 246 SHS developers in total. Accordingly, we select 246 non-SHS developers from the rest using random sampling to represent the ordinary developers.

4.3.3. Developer-Level Measures on Technical Performance. We define the following measures for evaluating the technical performance of a developer:

Average number of stars. GitHub provides starring function for users to mark their interest in projects. We count the average number of stars received by the repositories owned by the developer. Receiving more stars indicates a higher popularity of a project [43].

Number of followers. We use the number of followers a GitHub user has at the time of data collection as a signal of standing [44] within the community. Users with lots of followers are influential in the developer community as many people are paying attention to their activities.

H-index of Stars. The h-index [45] was originally introduced as a metric to evaluate both the productivity and citation impact of a scholar's research publications. It has been used to measure the influence of users' generated contents in social networks [46]. We define h-index of a developer as the maximum value of c such that the given developer has published c repositories that have each been starred at least c times. We use this metric to measure both the productivity and influence of a developer on GitHub.

4.3.4. Repository-Level Measures of Technical Performance. To examine the technical performance of repositories, we define the following measures:

Number of stars. We use the number of stars a repository has received to evaluate the popularity of a repository. A repository with many stars implies that many people show their interests in it [35, 47].

Number of forks. The "forking" function on GitHub enables developers to create a copy of a repository as their personal repository and then they can make changes to the code freely. Similar to the number of stars discussed above, the number of forks a repository has received is another important indicator that a repository is popular [35, 44, 48].

Number of open issues. Issues can be used to track bugs, enhancements, or other requests. In cases where the project's problem was suspect, submitters and core members often engaged in extended discussions about the appropriateness of the code [49, 50]. Repositories with more open issues receive more attention than those with less.

Lines of code changed per week ($LOC_{changed}$). This measure is defined as the average number of lines of code changed (the sum of additions and deletions) in all commits in a repository per week. It is a measure of outputs produced per unit time, which serves as a proxy for productivity [35, 51, 52, 53].

4.3.5. Hypothesis Testing. To accurately identify behavioral differences among different populations, we conduct statistical hypothesis testing on different groups.

First, we conduct Pearson's chi-squared test [54] to examine if there are significant differences in the work rhythms among different groups (i.e., regions and collaboration roles) of projects or developers. The Pearson's chi-squared test is commonly used for evaluating the significance of the association between two categories in sets of categorical data.

Second, we statistically validate if there are significant differences in the demographics and technical performance among different groups of software projects and developers. We compute the measures of each subject within the group and the measures of the population outside the group. Then, we apply the Mann–Whitney U test [55], which is commonly used to determine whether two independent samples are from populations with the same distribution.

The results of Pearson's chi-squared test and Mann–Whitney U test are measured by p-value, where a smaller p-value indicates higher significance level in rejecting the null hypothesis H_0. A p-value below 0.05 indicates a significant difference among the two populations in terms of the selected measure. Cramer's V and Cliff's delta effect size are used to supplement the results of Pearson's chi-squared test and Mann–Whitney U test, respectively.

4.4. User Survey. To investigate how developers experience and think of their work rhythms and productivity, we designed an online survey and sent it to developers in selected tech companies. The selected companies included a mix of large corporations and startups.

Our survey was reviewed and approved by the Institute of Science and Technology, Fudan University. Prior to the launch of the survey, we invited seven developers from different tech companies and did a pilot test. These participants completed the questionnaire and provided feedback, which we used to refine the survey. Next, we performed an undeclared pilot test involving 10 participants from selected companies in our dataset. We reviewed and discussed their responses to ensure that the questionnaire was free of major issues. After finalizing the survey, we distributed it online and asked the pilot participants to share the link to the survey with others. The survey had 1,516 views and received 92 responses from eligible respondents who identified their current job as software development. The survey questions are given in the appendix.

First, to validate our result on work rhythms, we asked survey participants about their required working hours and actual working hours on a typical work day. The participants are required to provide both their required and actual start time and end time of work or to implicate there is no required working hours.

Next, we asked participants about the time they spent on different work activities and programming themes. According to Meyer et al. [56], developers primarily identified coding-related tasks as productive, whereas activities such as attending meetings, reading, and writing emails were often considered as

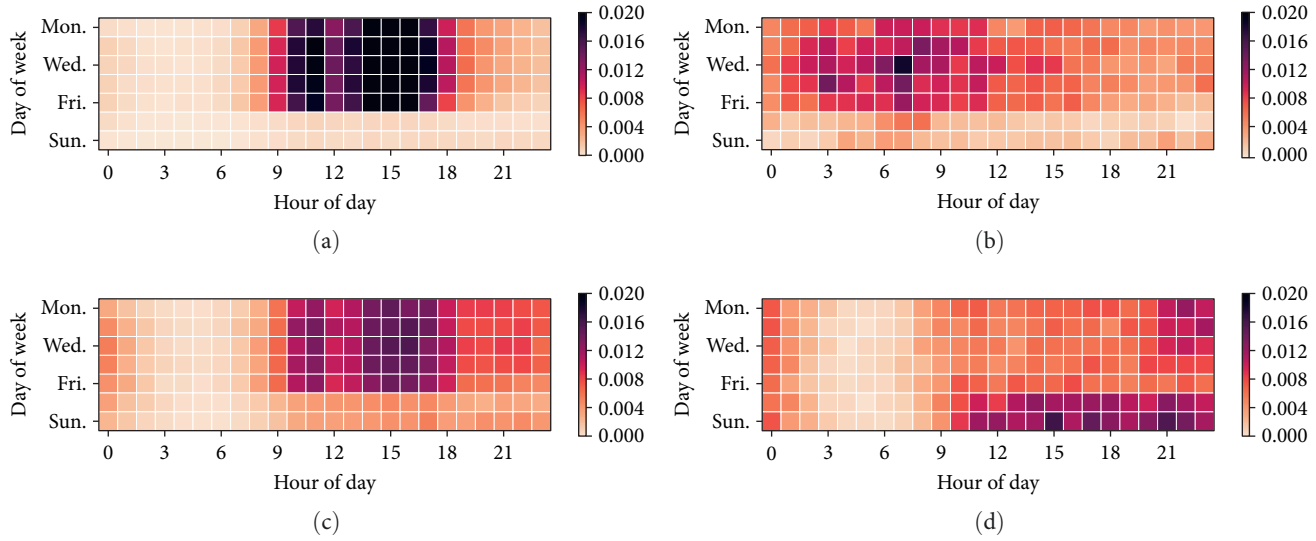

FIGURE 3: (a–d) Identified work rhythms among developers in GitHub dataset. Deeper color indicates higher commit frequency during the time slot, with the color bars on the right denoting the corresponding values of commit frequency.

unproductive. To gain insight into productivity both during and outside office hours, we asked participants to indicate the percentage of time they spent on various work activities during these periods, including coding, studying, project planning, writing documents, contacting colleagues, meeting, social activities, and others. Participants could choose one among the following five options to indicate the percentage of time they spent on each work activity or programming theme: "less than 5%," "between 5% and 20%," "between 20% and 35%," "between 35% and 50%," and "more than 50%." In addition, according to Meyer et al.'s [56] work, different types of programming tasks impact productivity differently. For instance, activities such as development and bug fixing were perceived as productive, whereas testing was considered as unproductive. We also asked participants about the percentage of time they spent on different programming themes in off-hours, using the same options as in the previous question. We asked participants to specify the detailed information if they had been involved in activities or programming theme other than those we listed.

Moreover, to understand whether developers believe extra working hours can contribute to productivity, we included a question asking whether extra working hours increase productivity. Participants were given the option to select either "agree," "neutral," or "disagree." Then, we cross checked their ideas with their motivations for working overtime. Beckers et al. [57] proposed that the outcome of extra working hours was affected by motivation. Highly motivated workers might have more active attitude towards extra working hours. To see how participants' perspectives on extra working hours differ with motivations, we included a multiple-choice question, listing nine common reasons for working overtime. These options were derived from initial interviews with several developers, who explained why they worked overtime. Their reasons were used as initial options in pilot tests. During the pilot tests, participants were asked to provide additional reasons if theirs were not listed. We then reviewed their answers and adjusted the options to ensure that the given reasons covered all cases. Finally, we concluded nine reasons from their responses, such as (1) handling emergencies (such as application crashes), (2) meeting deadlines, (3) making up for the time wasted on programming-independent work activities during office hours, (4) taxi reimbursement (some companies covered the taxi expenses within specific hours), (5) good environment of company (such as free snacks and air conditioners), (6) peer pressure (participants mentioned they stayed in the office after work because most of their colleagues did not leave), (7) company requirements, (8) enjoying coding in spare time, and (9) working for bonus. One or more options could be selected. Participants could also specify their reasons if they are not given as options.

5. Results

5.1. RQ1. What Are the Work Rhythms of Projects and Developers?

5.1.1. Work Rhythms of Developers. We apply clustering analysis on the commit behavior of developers in our dataset. Four work rhythms are detected among the developers in our dataset. We visualize the four detected work rhythms in the form of heatmap, as shown in Figures 3(a), 3(b), 3(c), and 3(d), with the x-axis representing the hours and the y-axis representing the days in a week. The color intensity of each time slot shows the aggregated commit frequency among developers, where darker color indicates higher commit frequencies. The detected work rhythms exhibit unique characteristics. The 48 hr in weekdays and weekends are divided into four subsets, as shown in Table 2. We observe the commit behavior in the subsets of hours and summarize the following characteristics:

#1: Nine-to-five worker. As shown in Figure 3(a), developers with work rhythm #1 concentrate on programming during regular office hours (9 a.m. to 5 p.m.) on

Table 2: Column clusters output by biclustering on developers dataset.

Subset	Weekday	Weekend
1	9 a.m. to 5 p.m.	—
2	7 p.m. to 12 a.m. (mid night)	3 p.m. to 11 p.m.
3	—	9 a.m. to 2 p.m. and 12 a.m.
4	1 a.m. to 8 a.m. and 6 p.m.	1 a.m. to 8 a.m.

The 48 hr in weekdays and weekends are divided into four time subsets. Developers with the same rhythm have the same degree of commit frequency in each time subset. For example, as shown in Figure 3(a), developers with rhythm #1 made commits at a high frequency during 9 a.m. to 5 p.m. on weekdays (i.e., time subset #1), whereas they have much fewer commits during the other time subsets.

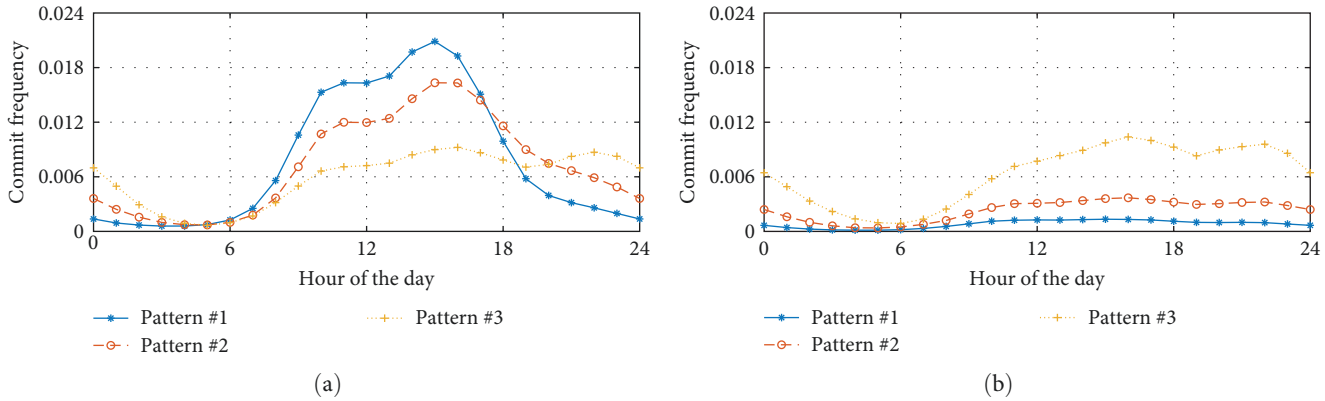

Figure 4: (a and b) Identified work rhythms among company repositories in GitHub dataset.

Table 3: Column clusters outputted by biclustering on repository dataset.

Subset	Weekday	Weekend
1	9 a.m. to 5 p.m.	—
2	7 p.m. to 12 a.m. (midnight)	9 a.m. to 12 a.m. (midnight)
3	1 a.m. to 8 a.m. and 6 p.m.	1 a.m. to 8 a.m.

The 48 hr in weekdays and weekends are divided into three subsets.

weekdays. They submit code changes less frequently after work hours or on weekends.

#2: Flex timers. As shown in Figure 3(b), the code submissions of developers with rhythm #2 are uniformly distributed on almost every hour on weekdays. Developers with this rhythm are likely to submit code changes at any time of the day and do not display fixed work and rest time.

#3: Overnight developers. As shown in Figure 3(c), developers with rhythm #3 submit their codes from 9 a.m. to 12 a.m. They also make code submissions on weekends following a similar daily working schedule as weekday, whereas the commit frequency on weekends is lower than that on weekdays.

#4: Off-hour developers. As shown in Figure 3(d), the peak time of the code submissions of developers with rhythm #4 is weekday nights and weekends, instead of regular working hours on weekdays.

5.1.2. Work Rhythms of Projects. We also apply clustering analysis on the commit behavior of repositories. Three work rhythms are detected among the repositories in our dataset. Figures 4(a) and 4(b) present the temporal distributions of commit frequency for identified rhythms. The 48 hr in weekdays and weekends are divided into three subsets, as shown in Table 3. We summarize the features of the three identified rhythms as follows:

#1: Typical office hours. Repositories with work rhythm #1 adopt typical work time, usually from 9 a.m. to 5 p.m on weekdays. Code changes are rarely submitted into those repositories on weekends.

#2: Slightly extended working hours. Repositories with rhythm #2 extend the typical work time to 6 p.m. on weekdays. Compared with developers in rhythm #1, repositories with rhythm #2 usually have more code submissions on weekends.

#3: Working over night and weekend. Repositories with rhythm #3 endure longer working hours than the other two rhythms. Developers of these repositories work equally on weekdays and weekends, starting from nine in the morning to the midnight.

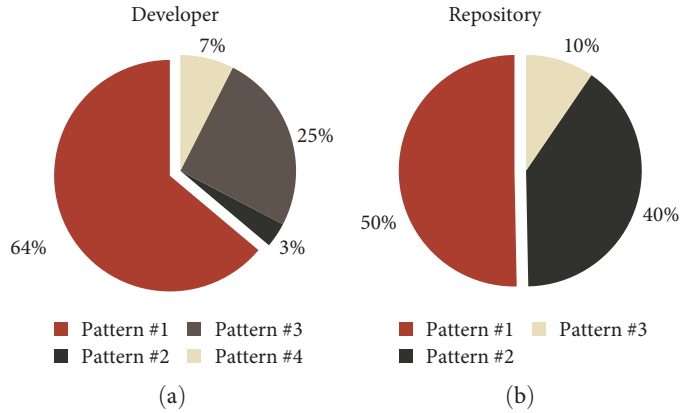

FIGURE 5: (a and b) Percentage of developers and repositories in each work rhythm.

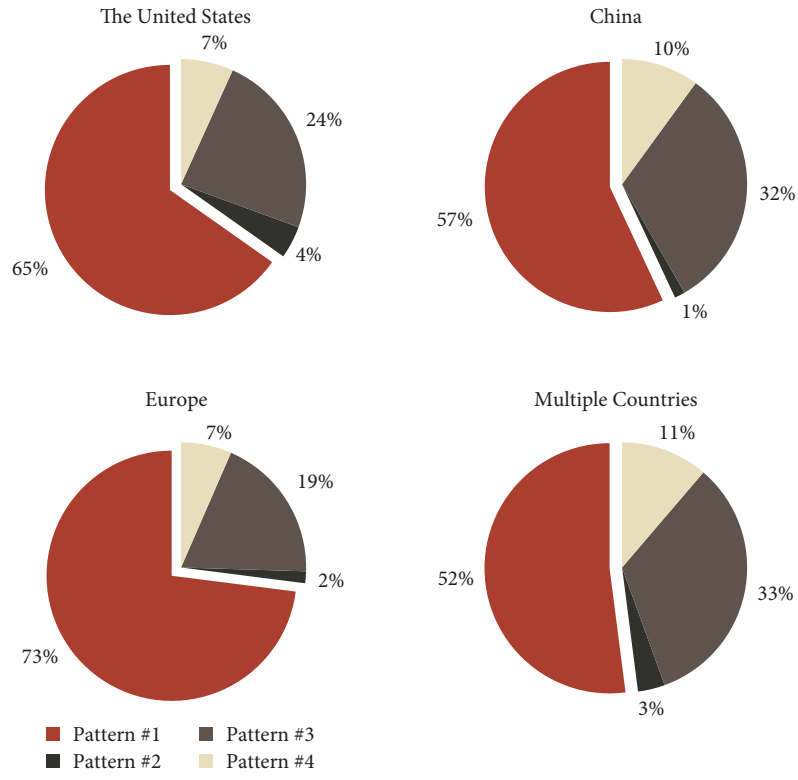

FIGURE 6: Percentage of developers with each type of work rhythm in different regions.

The percentage of developers and repositories in each detected work rhythm is shown in Figures 5(a) and 5(b), respectively. Among the four work rhythms detected in the developer dataset, we observe that about two-thirds of the developers follow rhythm #1 (typical working hours), which conforms to Claes et al.'s [1] finding. Among the three work rhythms detected in the repository dataset, rhythm #1 covers half of the repositories and rhythm #2 takes up 40% repositories, and the rest 10% repositories follow rhythm #3.

5.2. RQ2. Are Work Rhythms Related to Demographics and Collaboration Role? Do work rhythms vary across different regions? We examine the work regions of the developers. The percentages of developers per rhythm in each region are shown in Figure 6. Developers working for organizations in the United States and Europe mainly follow rhythm #1, whereas rhythms #3 and #4 are more prevalent among developers working for organizations in China or "multiple countries". We divide developers into two groups according to their work regions: the United States and Europe as a group and China and "multiple countries" as another group. We apply chi-square test to check the frequency of the two groups in each of the four rhythms. We find a significant difference between the two groups of developers in terms of the four work rhythms (p-value < 0.001, Cramer's $V = 0.325$).

Is there a correlation between work rhythm and developer seniority? We investigate the account age of developers in each rhythm and perform Mann–Whitney U test.

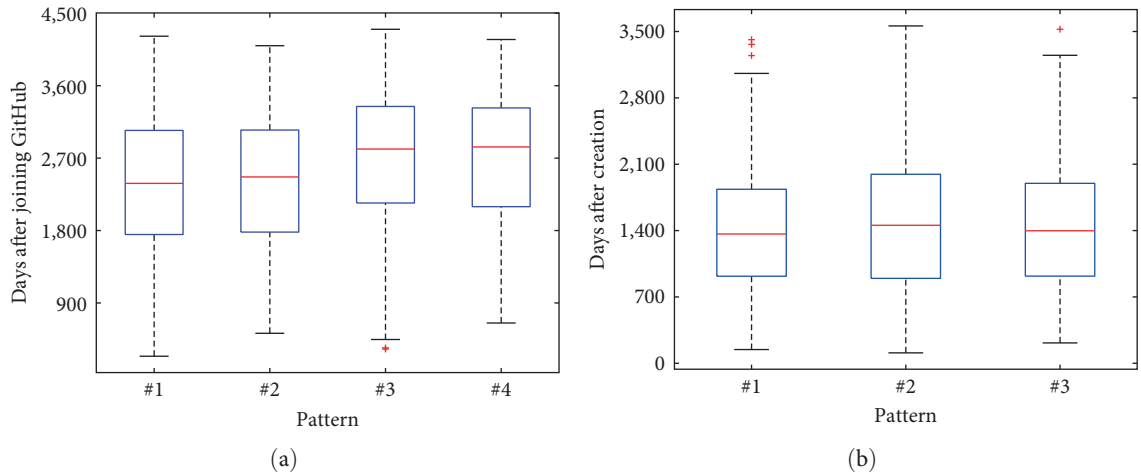

FIGURE 7: (a and b) Seniority of individual developers and maturity of repositories. The five horizontal lines of each box represent, from bottom to top, the minimum, first quartile, median, third quartile, and maximum values (the minimum/maximum are the lowest/highest values excluding outliers).

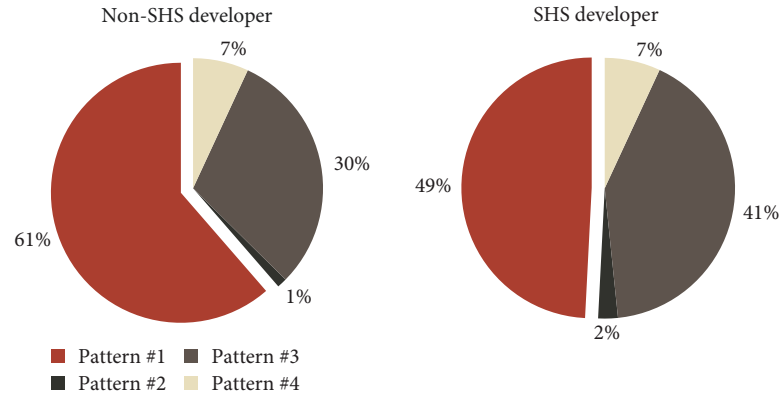

FIGURE 8: Percentage of developers with each type of work rhythm within SHS developers and Non-SHS developers.

Figure 7(a) shows the account ages of the developers for each work rhythm in box plots. Developers with rhythms #3 (p-value < 0.001, Cliff's delta $d = 0.20$) and #4 (p-value $= 0.004$, $d = 0.13$) tend to create their GitHub accounts earlier than those with other rhythms, which indicates that developers with rhythms #3 and #4 start to be engaged in software development earlier than those with the other two rhythms. Developers with rhythm #1 created their GitHub accounts later than others (p-value < 0.001, $d = -0.20$).

Is there a correlation between work rhythm and project maturity? We investigate the repository age in each rhythm and perform Mann–Whitney U test. As shown in Figure 7(b), repositories with the three rhythms do not show significant difference in terms of repository ages (p-values > 0.05).

Do SHS developers have specific work rhythms? The percentages of developers in each rhythm among SHS developers and ordinary developers are shown in Figure 8. There are more developers with rhythm #1 and fewer developers with rhythm #3 among ordinary developers than among SHS developers. We apply chi-square test and find a significant difference between SHS and non-SHS developers in terms of rhythms #1 and #3 (p-value $= 0.006$, Cramer's $V = 0.128$). Compared with ordinary developers, SHS developers tend to be overnight developers rather than work in fixed office hours.

5.3. RQ3. What Are the Correlations between Different Work Rhythms and Technical Performance? Next, we examine the effects of work rhythms on various measures of technical performance. Figures 9(a), 9(b), and 9(c) present the performance on the three measures for developers. We perform Mann–Whitney U test and the results are shown in Table 4. The value in each entry of the table is the ratio between the median value of the measures within the group and outside the group. A less than 1 value indicates that the developers with the selected rhythm have smaller value in the chosen measure and a higher than 1 value means otherwise. In addition, * marks the difference is significant with p-value ≤ 0.05, ** marks p-value ≤ 0.01 and *** marks p-value ≤ 0.001. As shown in Table 4, developers with rhythms #3 and #4 had more followers (Cliff's delta $d = 0.30$ and 0.16 respectively), received more stars from their own repositories ($d = 0.228$ and 0.158 respectively) and had higher

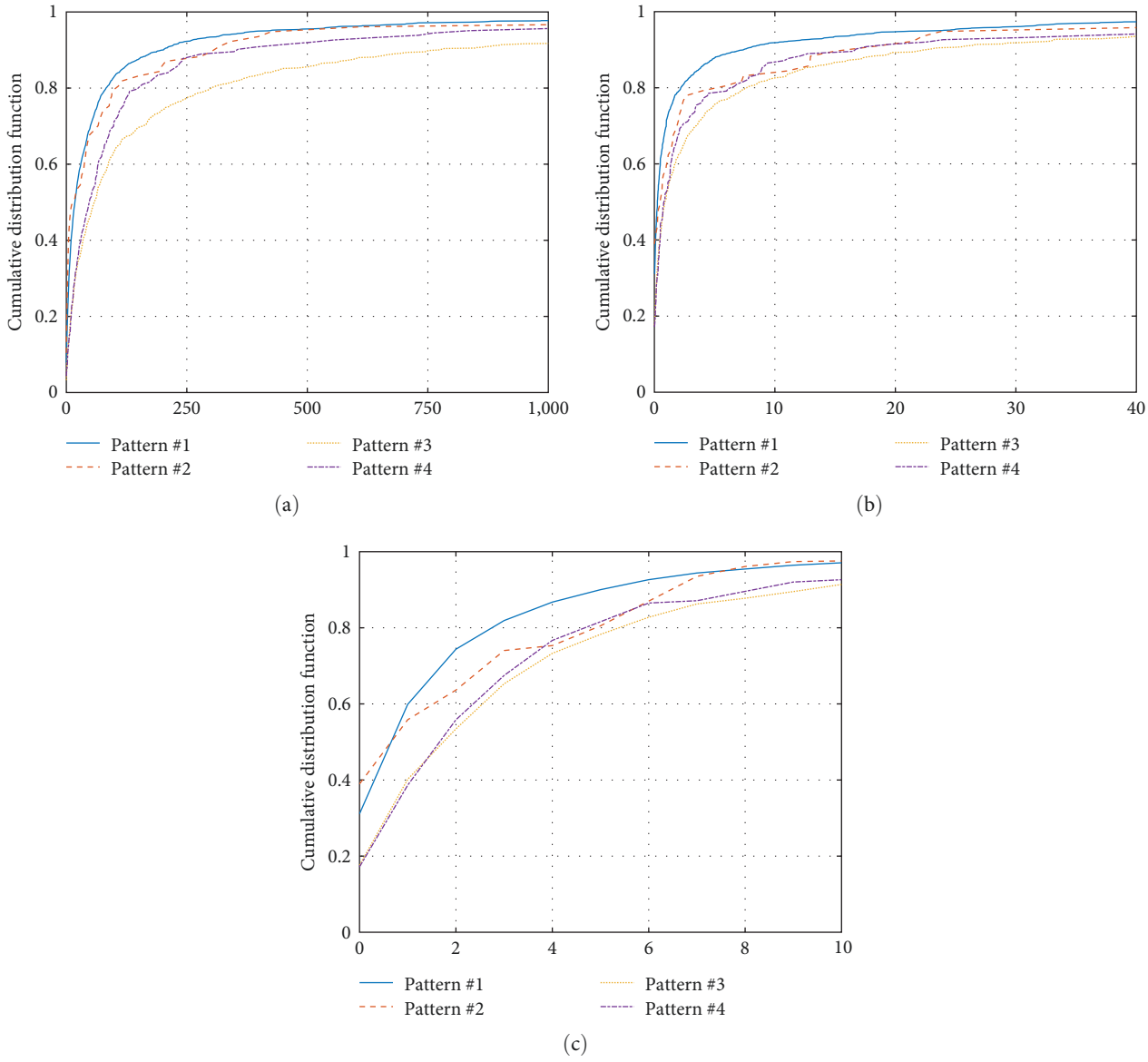

FIGURE 9: (a–c)Technical performance of developers with four types of work rhythms. The cumulative distribution functions of three metrics: number of followers, average number of stars, and h-index of the developers within each group.

TABLE 4: Correlation between developers' work rhythms and technical performance.

Rhythm	Average number of stars	Number of followers	h-Index
#1	0.30***	0.34***	0.50***
#2	1.25	0.63	1.00
#3	3.12***	2.95***	2.00***
#4	2.17***	1.85***	2.00***

*** marks difference is significant with p-value ≤ 0.001.

h-indexes ($d = 0.239$ and 0.169 respectively). In contrast, developers with rhythm #1 perform the worst in all three measures: average number of stars ($d = -0.235$), number of followers ($d = -0.282$), and h-index ($d = -0.243$).

We also examine the effect of repositories' work rhythms on technical performance and apply Mann–Whitney U test. The results are shown in Figures 10(a), 10(b), 10(c), and 10(d) and Table 5. Repositories with rhythm #2 receive more stars ($d = 0.085$) and have more forks ($d = 0.090$) than those with the other two rhythms. Repositories with rhythm #3 receive more stars than others ($d = 0.151$). As for the number of open issues, there is no significant difference among the three work rhythms.

It is interesting to find that although repositories with rhythm #1 have larger $\text{LOC}_{\text{changed}}$ than those with the other two rhythms, their values of the other measures of technical

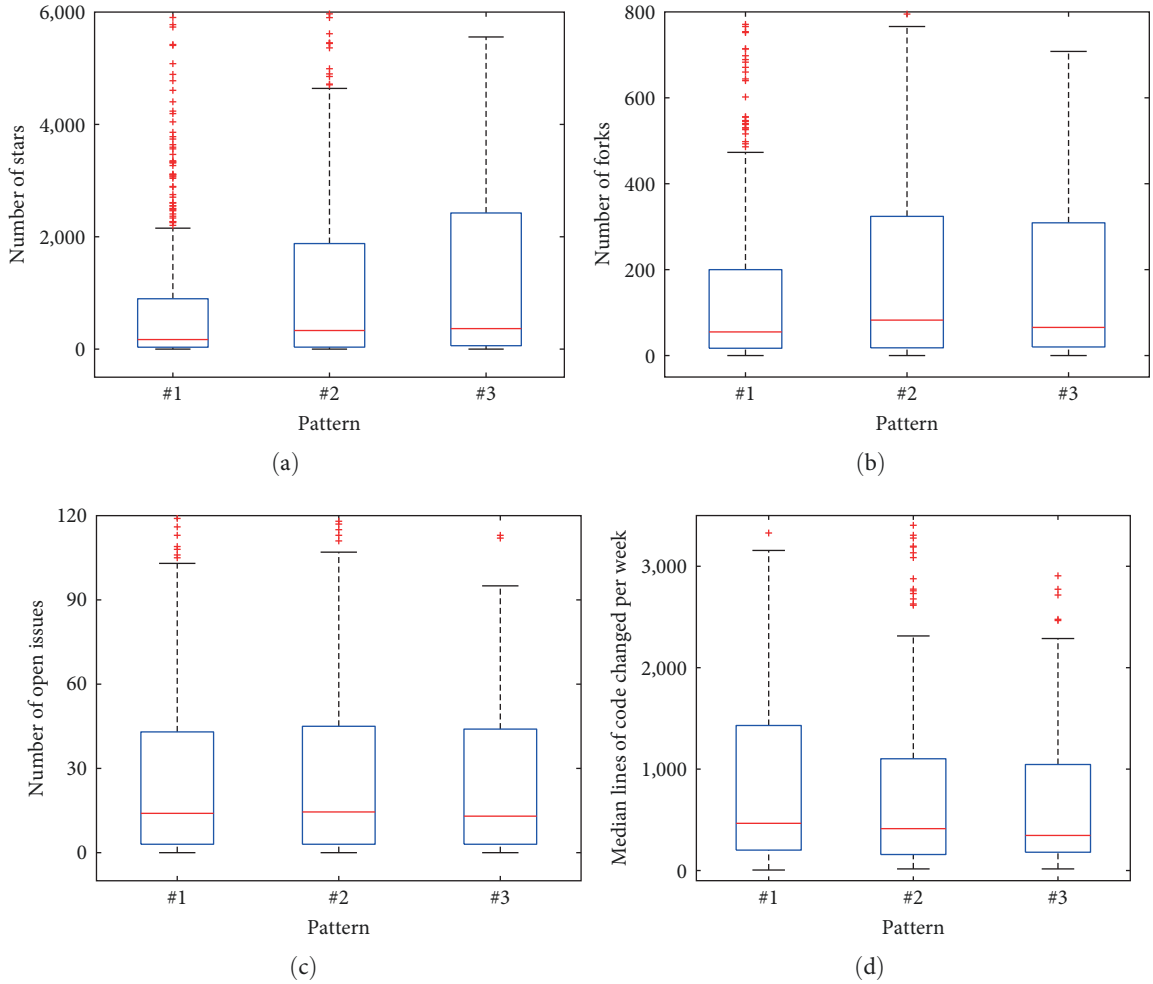

FIGURE 10: (a–d) Technical performance of repositories with three types of work rhythms. The five horizontal lines of each box represent the minimum, first quartile, median, third quartile, and maximum values from bottom to top (the minimum/maximum are the lowest/highest values excluding outliers).

TABLE 5: Repositories' work rhythms and performance.

Rhythm	Number of stars	Number of forks	Number of open issues	$LOC_{changed}$
#1	0.51^{***}	0.71^{**}	1.00	1.17^{*}
#2	1.66^{*}	1.50^{*}	1.03	0.91^{*}
#3	1.55^{**}	0.99	0.93	0.78

* marks the difference is significant with p-value ≤ 0.05, ** marks p-value ≤ 0.01, and *** marks p-value ≤ 0.001.

performance including stars ($d = -0.133$) and forks ($d = -0.10$) turn out to be lower. To discover the reason for this phenomenon, we further check the number of lines of code added and deleted per commit in each hour of a day. As shown in Figures 11(a) and 11(b), during the typical office hours, both the lines of code added and deleted per commit submitted into repositories with rhythm #1 are larger than those with the other two rhythms. During 4.–5 p.m. the sizes of the commits are the largest among commits in all hours of the day. The commit sizes peak between 4 and 5 p.m., suggesting a hypothesis that developers working on repositories with rhythm #1 may submit larger commits just before leaving the office to finish their workday on time. However, this practice might lead to lower code quality, necessitating deletions and rewrites the next day. As a result, these repositories have more frequent code changes, but their stars and forks are fewer.

5.4. RQ4. What Are Developers' Attitudes on Work Rhythm and Productivity?

5.4.1. Required Working Hour vs. Actual Working Hour. We ask participants about their companies' required working hour and their actual working hour on a typical work day. As shown in Figure 12, most participants reply that their companies require an 8-hr work day schedule. However, they usually work longer hours than required.

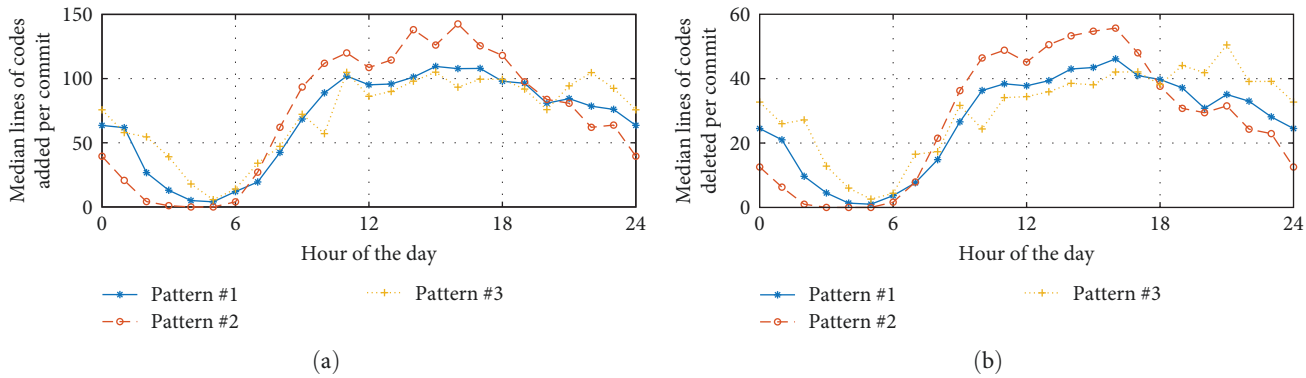

FIGURE 11: (a and b) Median number of lines of codes changed per commit.

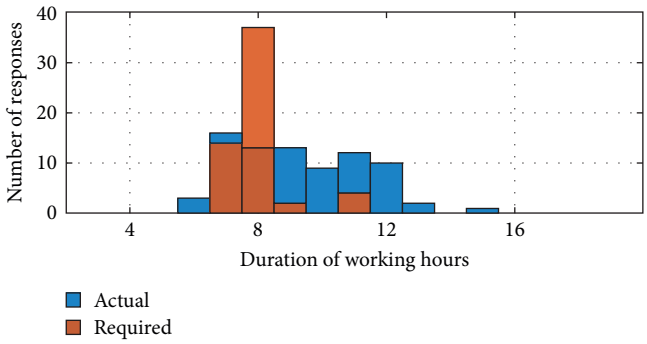

FIGURE 12: Required working hours vs. actual working hours.

5.4.2. Content Switch between Office Hours and Off-Hours. Figure 13 presents the distribution of activities during office hours and off-hours. Coding occupies the majority of time in both periods. The rankings for time spent on different tasks are mostly consistent, except for meetings and studying. During the office hours, meetings rank the third and the sixth respectively, whereas, during the off-hours, studying moves up to the second and meetings drop to the sixth. As shown in Figure 14, the most common programming activity during off-hours is developing, followed by testing, bug fixing, and creating backups.

5.4.3. Perspectives on Productivity in Extra Working Hours. Except for 25 participants (27.17%) who claim they do not work extra hours, 38 participants (41.30%) believe that additional working hours enhance productivity, 26 participants (28.26%) believe that additional work time does not boost productivity, and three participants (3.26%) are neutral.

We ask participants why they work overtime. Among all the options, "deadline" receives the most votes (33.3%). "Emergency" is the second most popular reason with 32.3% responses. In addition, 24.7% mention that they work overtime to make up for the time wasted on programming-independent work activities during office hours, 19.4% say that their companies require extra working hours, 16.1% agree that they work overtime because of peer pressure, 15.1% claim that they work overtime because they enjoy coding in their spare time, 7.5% say that they stay in the office after work because their companies provide good environment, and 6.5% mention that they work overtime because their companies provide taxi reimbursement. Only 1.1% say because of the bonus that their companies offer for overtime work.

We cross-check their motivations and views on the productivity of additional working hours. The results are shown in Figure 15, in which the height of a rectangle represents the proportion of participants who agree on the option and the flow represents the proportion of participants who agree on both the two options on each side. According to the results, more respondents agree extra working hours could increase productivity if they work overtime for emergencies (19 agree and 8 disagree), deadlines (18 agree and 10 disagree), making up for the time wasted on programming-independent work activities (13 agree and 10 disagree), taxi reimbursement (4 agree and 2 disagree), or good environment of their companies (3 agree and 2 disagree). In contrast, fewer respondents agree with the idea if they work overtime because of the company's requirements (8 agree and 9 disagree), peer pressures (7 agree and 8 disagree), or bonus (0 agrees and 1 disagrees). Among the respondents who work overtime because they enjoy coding in their spare time, the numbers of participants who hold both views are the same (4 agree and 4 disagree).

6. Discussion

6.1. Implications for Software Practice. The purpose of this paper is to investigate the work rhythms in software development and their effects on technical performance. We identify four typical work rhythms in the developer dataset. The typical working hours (from 9 a.m. to 5 p.m. on weekdays) cover 64% of developers in the dataset. The rest three rhythms represent an aperiodic work rhythm, an overnight work rhythm, and an off-hour work rhythm, respectively. In addition, three work rhythms are detected among repositories in the dataset. There are one typical work rhythm covering half of the repositories and two different types of overtime work rhythm.

Work rhythms are correlated with demographics and collaboration roles. Work rhythms with moderately extended working hours are more popular among senior developers. The maturity of a repository does not decrease the chance of requiring its developers to work extra hours. Developers who

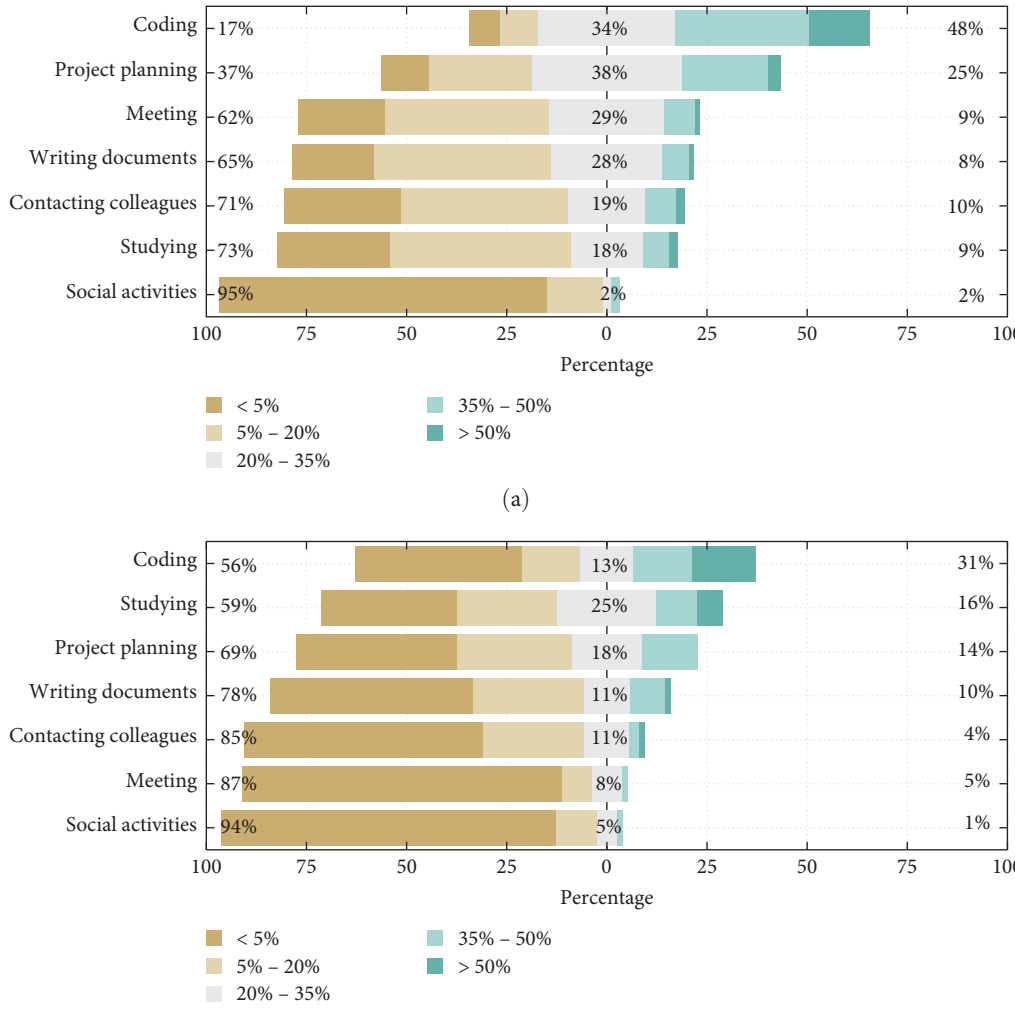

FIGURE 13: (a and b) Content switch between office hours and off-hours. The percentages on the right represent respondents who spend more than 35% of their time on the activities, whereas the percentages on the left indicate those who spend less than 20% of their time on the activities. The percentage in the middle shows respondents who select 20%–35%.

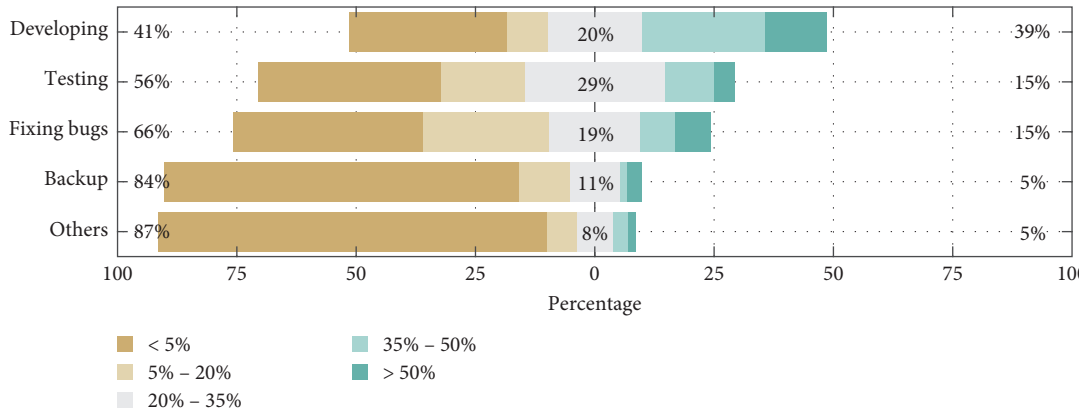

FIGURE 14: Content of coding in extra working hours.

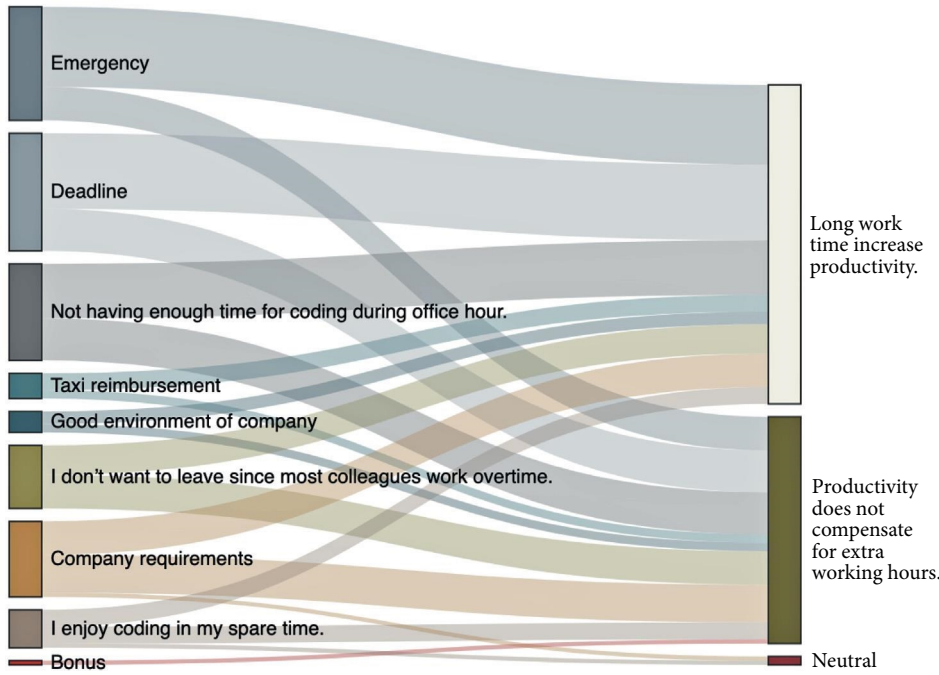

FIGURE 15: Relationship between the motivation of working overtime and perspective on extra working hours. The height of a rectangle represents the proportion of participants who agree on the option and the flow represents the proportion of participants who agree on both the two options on each side.

bridge collaboration groups consist of a higher proportion of "overnight developers" than others.

Work rhythms with a moderate amount of extended working hours appear to be associated with good technical performance. According to our results, projects and developers following the work rhythms with moderate hours of overtime work (rhythms #3 and #4 in developers' rhythms and rhythms #2 and #3 in repositories' rhythms) turn out to have better work performance than those following other rhythms. Projects and developers following fixed-hour work rhythms (rhythm #1 in developers' rhythms and rhythm #1 in repositories' rhythms) show poorer technique performance. Developers who follow aperiodic work rhythm (rhythm #2 in developers' rhythms) do not present better performance than others.

Developers' perspectives on productivity in extended working hours are influenced by their motivations of working overtime. They would feel extended working hours increase their productivity when the time for coding is insufficient due to some unexpected arrangments (such as approaching deadline) or the companies give clear incentives (such as reimbursing taxi fares), while fewer believe that extended working hours could increase productivity if they are under the requirement of companies, or work for bonus, or just follow the other colleagues to work overtime. Tech companies and teams could benefit from practices, for example, not forcing the members to work extra hours, and providing employees with better work environment and clear incentives.

6.2. Limitations and Threats to Validity. Being a first study to reveal work rhythms in software development and their effects on technical performance, there are a few limitations in our work. First, the data analysis in our study is limited to public open-source projects hosted on GitHub. Therefore, our conclusions are specific to the open-source projects and their contributors. Although our findings demonstrate notable distinctions between work rhythms, we cannot guarantee their broader applicability to the entire industry, as comprehensive data on a wider range of companies and closed-source projects would be necessary. We notice that there are alternative platforms such as GitLab where organizations release their work projects in a timely way. In addition, while we aim to capture an authentic snapshot of developer activity in open-source projects by forming an actual distribution of repositories in the companies, the variation in the number of repositories across these companies could potentially introduce bias into the results. In future work, we plan to explore other data sources to validate and expand our findings.

Second, our quantitative analysis on the work rhythms primarily focuses on the commit activities. Analysis on more comprehensive dataset could better reveal of rules of one field of research [58]. Other activities, such as meetings and document writing, also occupy developers' working hours; therefore, the time spent on programming might not fully represent their work schedule. However, because programming is a major task of developers' work, the temporal pattern of commits is a strong indicator of work time and our findings could provide insights into developers' working status. We also acknowledge that there might be a delay between the time of making commits and the actual time of completing coding tasks. However, because our

analysis is based on aggregated commits rather than individual ones, the impact of such delays should be negligible.

Third, the metrics that we use to measure the technical performance are indirect. For developers, we use average number of stars, number of followers, and h-index of stars as indicators of their reputations. For repositories, we consider number of stars, number of forks, and number of issues as proxies for user attention. More user attention and discussions mean that the repositories and developers are recognized by more people, which indicates their good technical performance. In addition, we use the lines of code changed per week to measure code productivity. Although these measures are intuitively reasonable, they could only show technical performance in some way. More metrics such as code quality should be addressed to obtain a comprehensive understanding of the technical performance.

7. Conclusions and Future Work

In this paper, we aim to discover work rhythms in software development and investigate their effects on technical performance. We found four work rhythms among individuals and three work rhythms among repositories in our dataset. The findings indicate that developers working for organizations in China or multiple countries tend to follow long-hour work rhythms, whereas those working for organizations in the United States and Europe tend to follow the typical work rhythm. Regarding the effects of work rhythms on technical performance, we found that a moderate amount of overtime work is related to good technical performance, whereas fixed office hours appear to be associated with projects and developers who receive less attention. In addition, our survey study indicates that developers usually tend to work longer than their companies' required working hours. A positive attitude towards overtime work is often linked to situations that require addressing unexpected issues, such as approaching deadlines, or when clear incentives are provided.

For future work, we aim to delve deeper into the underlying mechanisms behind developers' work. We wish to understand the underlying causes for different working rhythms by considering the interplay between work rhythms and other factors, such as technical roles and collaboration patterns. Furthermore, we plan to investigate the causal relationship between work rhythms and technical performance by conducting experimentation and incremental studies.

Appendix

The User Survey

We conducted a user survey to gain deeper insights into the working time of software developers. The survey was comprised of 12 questions and took approximately 5–10 min to complete. All responses were kept confidential and anonymous. The data collected from the survey were used for research purposes only and for overall analysis. The survey questions are listed below:

(1) What is the country of your company?
(2) How long have you been employed at your current company?
(3) What is the type of your current job? (e.g., development, testing, product management, etc.)
(4) What is your company's designated working hour for workdays? (Please fill in the start and end time in 24-hr format.)
(5) What are your actual working hour for workdays? (Please fill in the start and end time in 24-hr format.)
(6) How often do you work overtime on weekends? (Please choose one from the options.)
    (i) I work on both Saturday and Sunday every weekend
    (ii) I work on either Saturday or Sunday every weekend
    (iii) I sometimes work on weekends (less than once a week, please specify how many days per month on average)
    (iv) I never work on weekends
    (v) Other (please specify)
(7) Please rate the following statements according to how well they match your actual situation. (1: very inconsistent, 2: somewhat inconsistent, 3: average, 4: somewhat consistent, 5: very consistent.)
    (i) Most of my colleagues work overtime.
    (ii) My company provides benefits for overtime worker.
    (iii) I enjoy working overtime.
    (iv) I work during holidays.
    (v) I work more before/after holidays.
(8) During your designated work hours, what percentage of your time is spent on each of the following activities? (1: below 5%, 2: 5%–20%, 3: 20%–35%, 4: 35%–50%, 5: above 50%.)
    (i) Coding
    (ii) Project planning
    (iii) Meetings
    (iv) Reading/writing documents and preparing reports
    (v) Handling other work tasks, e.g., reading/writing emails, etc.
    (vi) Learning software, tools, skills, etc.
    (vii) Business entertainment, e.g., hosting colleagues, etc.
    (viii) Leisure activities
    (ix) Other

(9) During your off-work hours, what percentage of your time is spent on each of the following work-related activities? (1: below 5%, 2: 5%–20%, 3: 20%–35%, 4: 35%–50%, 5: above 50%. Skip this question if you do not work overtime.)
   (i) Coding
   (ii) Project planning
   (iii) Meetings
   (iv) Reading/writing documents and preparing reports
   (v) Handling other work tasks, e.g., reading/writing emails, etc.
   (vi) Learning software, tools, skills, etc.
   (vii) Business entertainment, e.g., hosting colleagues, etc.
   (viii) Leisure activities
   (ix) Other

(10) During your off-work hours, what percentage of your time is spent on each of the following programming tasks? (1: below 5%, 2: 5%–20%, 3: 20%–35%, 4: 35%–50%, 5: above 50%. Skip this question if you do not work overtime.)
   (i) Development
   (ii) Testing
   (iii) Backups
   (iv) Bug fixes
   (v) Other

(11) What is the main reason you engage in work-related programming activities after work hours? (Multiple-choice question.)
   (i) I do not work overtime.
   (ii) Deadlines.
   (iii) Handling emergencies (such as application crashes).
   (iv) Making up for the time wasted on programming-independent work activities during office hours.
   (v) Company requirements.
   (vi) Peer pressure (most of my colleagues have not left).
   (vii) Enjoying coding in spare time.
   (viii) The company provides good environment, e.g., free smacks and air conditioners.
   (ix) The company provides taxi reimbursements within specific hours.
   (x) Working for bonus.
   (xi) Other. (Please specify the reason.)

(12) Do you think extra working hours increase productivity?
   (i) I do not work overtime.
   (ii) Agree—overall, working overtime increases my work output.
   (iii) Disagree—overtime work does not compensate for my extra working hours.
   (iv) Neutral.

Data Availability

As the data used in this work are publicly visible and accessible on GitHub, researchers interested in accessing the data can retrieve it directly from the GitHub platform with its official API. To ensure transparency and facilitate further research, the list of organizations and repositories in our dataset is publicly available on GitHub: https://github.com/jiayunz/Work-Rhythms-in-Software-Development. Researchers can refer to this repository to gain access to the specific projects and repositories included in the dataset. For any inquiries or requests related to the dataset, researchers can contact the corresponding author through email.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been sponsored by National Natural Science Foundation of China (nos. 62072115 and 62102094), Shanghai Science and Technology Innovation Action Plan Project (no. 22510713600), European Union's Horizon 2020 Research and Innovation Programme under the grant agreement no. 101021808, and Marie Skłodowska Curie grant agreement no. 956090.

References

[1] M. Claes, M. Mäntylä, M. Kuutila, and B. Adams, "Do programmers work at night or during the weekend?" in *Proceedings of the 40th International Conference on Software Engineering*, pp. 705–715, IEEE, 2018.

[2] J. Marlow and L. Dabbish, "Activity traces and signals in software developer recruitment and hiring," in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, pp. 145–156, ACM, 2013.

[3] B. B. Baltes, T. E. Briggs, J. W. Huff, J. A. Wright, and G. A. Neuman, "Flexible and compressed workweek schedules: a meta-analysis of their effects on work-related criteria," *Journal of Applied Psychology*, vol. 84, no. 4, pp. 496–513, 1999.

[4] L. Smith, S. Folkard, P. Tucker, and I. Macdonald, "Work shift duration: a review comparing eight hour and 12 hour shift systems," *Occupational and Environmental Medicine*, vol. 55, no. 4, pp. 217–229, 1998.

[5] G. P. Krueger, "Sustained work, fatigue, sleep loss and performance: a review of the issues," *Work & Stress*, vol. 3, no. 2, pp. 129–141, 1989.

[6] E. J. Josten, J. E. Ng-A-Tham, and H. Thierry, "The effects of extended workdays on fatigue, health, performance and satisfaction in nursing," *Journal of Advanced Nursing*, vol. 44, no. 6, pp. 643–652, 2003.

[7] S. W. Lockley, L. K. Barger, N. T. Ayas, J. M. Rothschild, C. A. Czeisler, and C. P. Landrigan, "Effects of health care

provider work hours and sleep deprivation on safety and performance," *The Joint Commission Journal on Quality and Patient Safety*, vol. 33, no. 11, pp. 7–18, 2007.
[8] A. Richardson, C. Turnock, L. Harris, A. Finley, and S. Carson, "A study examining the impact of 12-hour shifts on critical care staff," *Journal of Nursing Management*, vol. 15, no. 8, pp. 838–846, 2007.
[9] S. M. Keller, P. Berryman, and E. Lukes, "Effects of extended work shifts and shift work on patient safety, productivity, and employee health," *AAOHN Journal*, vol. 57, no. 12, pp. 497–504, 2009.
[10] J. Eyolfson, L. Tan, and P. Lam, "Do time of day and developer experience affect commit bugginess?" in *Proceedings of the 8th Working Conference on Mining Software Repositories*, pp. 153–162, ACM, 2011.
[11] J. Eyolfson, L. Tan, and P. Lam, "Correlations between bugginess and time-based commit characteristics," *Empirical Software Engineering*, vol. 19, no. 4, pp. 1009–1039, 2014.
[12] L. Prechelt and A. Pepper, "Why software repositories are not used for defect-insertion circumstance analysis more often: a case study," *Information and Software Technology*, vol. 56, no. 10, pp. 1377–1389, 2014.
[13] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein, "Spectral biclustering of microarray data: coclustering genes and conditions," *Genome Research*, vol. 13, no. 4, pp. 703–716, 2003.
[14] R. S. Burt, *Structural Holes: The Social Structure of Competition*, Harvard University Press, 2009.
[15] D. E. Perry, N. A. Staudenmayer, and L. G. Votta, "Understanding and improving time usage in software development," *Software Process*, vol. 5, pp. 111–135, 1995.
[16] T. D. LaToza, G. Venolia, and R. DeLine, "Maintaining mental models: a study of developer work habits," in *Proceedings of the 28th International Conference on Software Engineering*, pp. 492–501, ACM, 2006.
[17] E. Fu, Y. Zhuang, J. Zhang, J. Zhang, and Y. Chen, "Understanding the user interactions on GitHub: a social network perspective," in *Proceedings of CSCWD*, pp. 1148–1153, IEEE, 2021.
[18] Q. Gong, Y. Chen, X. He et al., "DeepScan: exploiting deep learning for malicious account detection in location-based social networks," *IEEE Communications Magazine*, vol. 56, no. 11, pp. 21–27, 2018.
[19] X. He, Q. Gong, Y. Chen, Y. Zhang, X. Wang, and X. Fu, "DatingSec: detecting malicious accounts in dating apps using a content-based attention network," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2193–2208, 2021.
[20] M. Saini and K. Kaur, "Fuzzy analysis and prediction of commit activity in open source software projects," *IET Software*, vol. 10, no. 5, pp. 136–146, 2016.
[21] F. Javeed, A. Siddique, A. Munir, B. Shehzad, and M. I. U. Lali, "Discovering software developer's coding expertise through deep learning," *IET Software*, vol. 14, no. 3, pp. 213–220, 2020.
[22] M. A. Aljemabi, Z. Wang, and M. A. Saleh, "Mining social collaboration patterns in developer social networks," *IET Software*, vol. 14, no. 7, pp. 839–849, 2020.
[23] A. Sajedi-Badashian and E. Stroulia, "Investigating the information value of different sources of evidence of developers' expertise for bug assignment in open-source projects," *IET Software*, vol. 14, no. 7, pp. 748–758, 2020.
[24] B. Traullé and J.-M. Dalle, "The evolution of developer work rhythms," in *International Conference on Social Informatics*, pp. 420–438, Springer, 2018.
[25] J. Zhang, Y. Chen, Q. Gong et al., "Understanding the working time of developers in IT companies in China and the United States," *IEEE Software*, vol. 38, no. 2, pp. 96–106, 2021.
[26] F. Khomh, B. Adams, T. Dhaliwal, and Y. Zou, "Understanding the impact of rapid releases on software quality," *Empirical Software Engineering*, vol. 20, no. 2, pp. 336–373, 2015.
[27] Q. Gong, J. Zhang, Y. Chen et al., "Detecting malicious accounts in online developer communities using deep learning," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1251–1260, 2019.
[28] Q. Gong, Y. Liu, J. Zhang et al., "Detecting malicious accounts in online developer communities using deep learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 10, pp. 10633–10649, 2023.
[29] R. Goyal, G. Ferreira, C. Kästner, and J. Herbsleb, "Identifying unusual commits on GitHub," *Journal of Software: Evolution and Process*, vol. 30, no. 1, Article ID e1893, 2018.
[30] J. MacQueen, "Classification and analysis of multivariate observations," in *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297, University of California, Los Angeles, LA, USA, 1967.
[31] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *2nd International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, ACM, 1996.
[32] Z. Cheng, M. Trépanier, and L. Sun, "Probabilistic model for destination inference and travel pattern mining from smart card data," *Transportation*, vol. 48, no. 4, pp. 2035–2053, 2021.
[33] Z. Li, H. Yan, C. Zhang, and F. Tsung, "Individualized passenger travel pattern multi-clustering based on graph regularized tensor latent dirichlet allocation," *Data Mining and Knowledge Discovery*, vol. 36, no. 4, pp. 1247–1278, 2022.
[34] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
[35] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov, "Quality and productivity outcomes relating to continuous integration in GitHub," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pp. 805–816, ACM, 2015.
[36] R. S. Burt, M. Kilduff, and S. Tasselli, "Social network analysis: foundations and frontiers on advantage," *Annual Review of Psychology*, vol. 64, no. 1, pp. 527–547, 2013.
[37] Z. Lin, Y. Zhang, Q. Gong, Y. Chen, A. Oksanen, and A. Y. Ding, "Structural hole theory in social network analysis: a review," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 3, pp. 724–739, 2022.
[38] W. Li, Z. Xu, Y. Sun et al., "DeepPick: a deep learning approach to unveil outstanding users with public attainable features," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 291–306, 2023.
[39] T. Bhowmik, N. Niu, P. Singhania, and W. Wang, "On the role of structural holes in requirements identification: an exploratory study on open-source software development," *ACM Transactions on Management Information Systems*, vol. 6, no. 3, pp. 1–30, 2015.
[40] W. Xu, M. Rezvani, W. Liang, J. X. Yu, and C. Liu, "Efficient algorithms for the identification of top- structural hole spanners in large social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 1017–1030, 2017.
[41] Q. Gong, J. Zhang, X. Wang, and Y. Chen, "Identifying structural hole spanners in online social networks using machine learning," in *Proceedings of the ACM SIGCOMM. 2019 Conference Posters and Demos*, pp. 93–95, 2019.

[42] M. Gao, Z. Li, R. Li et al., "EasyGraph: a multifunctional, cross-platform, and effective library for interdisciplinary network analysis," *Patterns*, vol. 4, no. 10, Article ID 100839, 2023.
[43] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in GitHub," in *Proceedings of the 36th International Conference on Software Engineering*, pp. 356–366, ACM, 2014.
[44] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub: transparency and collaboration in an open software repository," in *Proceedings of the ACM. 2012 conference on Computer Supported Cooperative Work*, pp. 1277–1286, ACM, 2012.
[45] J. E. Hirsch, "An index to quantify an individual's scientific research output," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 46, pp. 16569–16572, 2005.
[46] Q. Gong, Y. Chen, X. He et al., "Cross-site prediction on social influence for cold-start users in online social networks," *ACM Transactions on the Web*, vol. 15, no. 2, pp. 1–23, 2021.
[47] H. Borges, A. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of GitHub repositories," in *2016 IEEE International Conference on Software Maintenance and Evolution*, pp. 334–344, IEEE, 2016.
[48] J. Jiang, D. Lo, J. He, X. Xia, P. S. Kochhar, and L. Zhang, "Why and how developers fork what from whom in GitHub," *Empirical Software Engineering*, vol. 22, no. 1, pp. 547–578, 2017.
[49] J. Tsay, L. Dabbish, and J. Herbsleb, "Let's talk about it: evaluating contributions through discussion in GitHub," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ACM, 2014.
[50] G. Vale, A. Schmid, A. R. Santos, E. S. De Almeida, and S. Apel, "On the relation between GitHub communication activity and merge conflicts," *Empirical Software Engineering*, vol. 25, no. 1, pp. 402–433, 2020.
[51] B. Vasilescu, K. Blincoe, Q. Xuan et al., "The sky is not the limit: multitasking across GitHub projects," in *Proceedings of the 38th International Conference on Software Engineering*, pp. 994–1005, IEEE, 2016.
[52] O. Dieste, A. M. Aranda, F. Uyaguari et al., "Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2457–2542, 2017.
[53] E. Oliveira, E. Fernandes, I. Steinmacher, M. Cristo, T. Conte, and A. Garcia, "Code and commit metrics of developer productivity: a study on team leaders perceptions," *Empirical Software Engineering*, vol. 25, no. 4, pp. 2519–2549, 2020.
[54] K. Pearson, "X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 50, no. 302, pp. 157–175, 2009.
[55] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947.
[56] A. N. Meyer, T. Fritz, G. C. Murphy, and T. Zimmermann, "Software developers' perceptions of productivity," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 19–29, ACM, 2014.
[57] D. G. J. Beckers, D. van der Linden, P. G. W. Smulders, M. A. J. Kompier, M. J. P. M. van Veldhoven, and N. W. van Yperen, "Working overtime hours: relations with fatigue, work motivation, and the quality of work," *Journal of Occupational and Environmental Medicine*, vol. 46, no. 12, pp. 1282–1289, 2004.
[58] J. Wu, B. Ye, Q. Gong et al., "Characterizing and understanding development of social computing through DBLP: a data-driven analysis," *Journal of Social Computing*, vol. 3, no. 4, pp. 287–302, 2022.