2022 29th Asia-Pacific Software Engineering Conference (APSEC) | 978-1-6654-5537-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/APSEC57359.2022.00013

Exploring Activity and Contributors on GitHub: Who, What, When, and Where

Xiaoya Xia
Data Science and Engineering
East China Normal University
Shanghai, China
51195100018@stu.ecnu.edu.cn

Zhenjie Weng
Data Science and Engineering
East China Normal University
Shanghai, China
51205903087@stu.ecnu.edu.cn

Wei Wang
Data Science and Engineering
East China Normal University
Shanghai, China
wwang@dase.ecnu.edu.cn

Shengyu Zhao
School of Electronic and Information Engineering
Tongji University
Shanghai, China
frank_zsy@tongji.edu.cn

***Abstract*—Apart from being a code hosting platform, GitHub is the place where large-scale open collaborations and contributions happen. Every minute, thousands of developers are submitting code, having discussions of issues or pull requests, with all user behaviors recorded in the GitHub Event Stream (GES). Exploration of the activities in the GES could help understand who is active, the way they work, the time when they are active and even their location. To this end, a large-scale analysis was initially performed based on the 0.86 billion event records generated in 2020. We extracted 902K active contributors out of 14 million GitHub accounts by observing their activity distribution, then explored their behavior distribution, active time in the day and week, and estimated time zone distributions on the basis of their circadian activity rhythm. To go deeper, a case study of 79 projects in CNCF and contrast analyses of different project maturity levels were conducted. Our results showed that from a macro perspective, bots are increasingly more active and can serve numerous projects. Contributors work on weekdays, and are globally more inclined toward the daytime working hours in the Americas and Europe. The time zone distribution also reveals that UTC+2 and UTC-4 have the most active contributors. A critical discovery was the validation and quantification of a high bus factor risk exists in the OSS ecosystem. Whether from a large group point of view or within specific projects, a rather small group of OSS contributors (less than 20%) undertook the majority of the work. The GES can provide a wealth of information about open source software (OSS). Our findings provide insights into global GitHub collaboration behaviors and may be of help for researchers and practitioners to further understand modern OSS ecosystem.**

***Index Terms*—GitHub behavior data, Contributor Activity, Open Source Collaboration**

I. Introduction

GitHub is now the indispensable platform for open source software (OSS) development. In addition to code hosting and version control, it provides a fertile environment for large-scale distributed social coding and collaborations[1]. Countless OSS clusters and communities are born on it. The massive volumes of behavioral data produced by OSS developers may serve to reflect individual contributing patterns[2], [3], group collaboration modes[4], [5], [6], [7], as well as community status indicators[8], [9]. GitHub defines multiple kinds of events that are triggered by different actions and records the behavior stream of users. Tool suites have been explored and developed to mine the event stream from the GitHub API, and provide a way to obtain a panoramic view of modern OSS behaviors that occur on GitHub. For example, GHTorrent[10] and GHArchive[11] are both OSS projects that aim at monitoring and providing an offline GitHub Event Stream (GES)[12], making it easily accessible for further analysis.

One of the difference between the collaboration mechanism in open source software development and the traditional development is the emphasize on voluntarily contribution instead of mandatory work. The boundaries in OSS thus became blurring, making it difficult to identify, track, and understand the pattern of contributors. The issue becomes particularly critical for giant open source projects like Kubernetes, TensorFlow, VSCode, or large federated organizations like the Apache Software Foundation (ASF), the Cloud Native Computing Foundation (CNCF), and the Linux Foundation, to name a few[13]. For an OSS ecosystem (whether in project or foundation level) with tremendous trace events and personnel flow, the difficulty not only lies in (i) how to identify and have a deep understanding of its contributors especially the ones that are most devoted, but also (ii) where the baseline is. While past works have formulated ways to identify patterns of different roles (e.g.,newcomers[14], causal contributors[3], [15], maintainers[16]), the perspective mostly stays on the division of authority. To our knowledge there still lacks a comprehensive framework for large project owners or federated OSS organization managers to get a great grasp of the contributors and activity status based on event traces. Many studies take *commit*[17], [18], [19] as the main factor indicating a developer's activity, with social features built into GitHub, software developers' activities should not be limited solely to code[20]. For instance, GitHub Issue is an

978-1-6654-5537-4/22/$31.00 ©2022 IEEE
DOI 10.1109/APSEC57359.2022.00013

important venue where many project interactions take place. The activities pertaining to issues are hard to ignore, and are relative to project evolution as a whole.

One key point that contributes to GitHub's success is the introduction of the pull-based collaboration model[4], in which a contributor forks a repository to make changes independently, then submits a pull request to the main repository for review[21]. A contributor usually opens an issue that links to the pull request for any discussion. Each pull request is reviewed by project members to decide whether it will be merged. As we can see in the pull-based model, contributions are related to several behaviours including discuss, open issues/pull requests, review and so forth. Contributorship, defined as only a code commit, does not cover the complete spectrum of contributions[12].

Hence, in this study, we consider multiple behaviors extracted from the GES to position contributors and their activities in distributed collaborations. Specifically, we defined a composite activity model that takes into account five behavior types representing a complete pull-based collaboration process on GitHub to describe the activity level of contributors. We proposed four basic *Wh-questions* to reveal active contributors, their behaviors and work patterns from their GES event traces, serving as an analysis framework for projects owners:

Q1: Who is active and how active are they?

Q1 aims to provide a basic overview of the activity ecosystem on GitHub, which includes bots, users, and contributors. Since the study concentrates on activities in contributions, we define active contributors (ACs) as contributors who are among the top-ranked by the activity model. Focusing on ACs, we further formulated Q2, Q3, and Q4:

Q2: What are the active contributors doing?

Q3: When are the active contributors active?

Q4: Where are the active contributors?

Q3 and Q4 investigate ACs from the perspective of the working time, to observe the circadian rhythm and weekly pattern of contributors, and further estimate their time zone distribution based on the circadian pattern observed.

For each question, we try to answer and understand contributors from holistic and ad-hoc views. We performed a large-scale analysis based on 0.86 billion complete GES records archived for the whole year in 2020. To gain a deeper insight and make our findings more easily validated, we also performed a case study on CNCF projects. The article revolves around the above four basic questions, which are discussed in detail in Section 5. Key contributions are highlighted below:

- We proposed a multifaceted activity model to measure contribution activity in GitHub collaborations. The model can effectively help projects to identify their most active GitHub contributors who are not limited to code contributions;
- Based on Pareto's law (also known as the 80/20 rule), we found a great imbalance in the activity distribution of GitHub contributors. 20% of the contributors accounted for almost 90% of the total activity. In some communities, the distribution can deviate as high as 90/10;
- We conducted a holistic observation of the most active contributors from both a panoramic and an ad-hoc point of view. The Wh-questions are proposed and formulated as a framework for OSS communities to get a better assessment of their status;
- We proposed an estimation method to present the time zone distribution based on contributors' active time frame period, which supplement the rough location distribution when there is a lot of missing contributor geolocation information of a community.

II. Related Work

GitHub As the locus of most of the OSS development activities, efforts have been taken in early research to mine the large-scale GitHub timeline. Gousios *et al.*[22] built a set of tools to mine the GitHub data. GHTorrent[10] is maintained by Gousios to provide an offline mirror of GitHub data. Mombach *et al.*[23] compares GHTorrent with GitHub Rest API[24] and GHArchive[11]; the latter was created by Gregorik to store public events. The study reported that GHTorrent and GHArchive are more appropriate to retrieve historical data, while GitHub REST API is more appropriate for the most recent and up-to-date data. Gousios *et al.* also describes the distributed software development process on GitHub as the pull-based development model[21].

Measuring Activity GitHub integrates some metrics to show the popularity trend, with the most explicit proxy indicated by the number of stars of a repository or a developer's followers[25]. However, popularity only appears as a static status. Previous work tended to consider *commits* as the single activity representing a developer's working pattern or describing a project's developing status. Kolassa *et al.* [17] focused on the commit frequency, considering a commit as the fundamental unit of work as code contribution. The study also derived an *activity* indicator based on the commit frequency. Other researchers such as Young *et al.*[12] pointed out that code change is not the only measurement of contributions; they committed to defining new contribution taxonomies. Studies in software activity measurements have various perspectives such as developers[26], projects[27], [9], patterns and impacts[28], and variable controls[4].

Contributor's dates and time Maëlick *et al.*[18] studied programmers' circadian working patterns by investigating the commit timestamps of 86 large open source software projects and found that two thirds of engineers empirically followed the typical hours from 10 am to 6 pm, and did not usually work during nights and weekends. Sall *et al.*[29] studied weekend work activity patterns in the San Francisco Bay Area. The results show that variables like gender, race, type of work and income affect the likelihood of working during the weekend.

Contributor's geographic location Most of the studies related to geographical information extracted their data from GitHub profiles[30], [31]. A restriction of using GitHub profiles data is that the input can be irregular or false, like "earth" or "internet"[32]; also, the data is static, while developers can move all over the world. In light of this, Gonzalez-Barahona *et* *al.*[33] used time zone information attached in git-log commit information to determine the geographical distribution. As mentioned previously, the limitation of this method is that a commit is not the only proxy of contribution in a project.

III. Dataset

The GitHub Event API defines multiple types of events that cover a diverse variety of activities initiated by users on the platform, ranging from starring and forking a project to collaborating on issues or pull requests. Such an event stream likely reflects certain work patterns exhibited by developers. Although the GitHub Event API is well designed and documented, one significant challenge of analyzing its data is the API rate limit of 5,000 requests per hour, which makes it difficult to run large-scale explorations. Therefore, we turned to GHArchive[11], which is also an open source project developed to monitor the public GitHub timeline and make it available in hourly archives (downloadable in the JSON format). We used an HTTP client to download and collect all the archives from 2020 (from 00:00:00 1/1/2020 to 23:59:59 12/31/2020).

The 2020 data volume is quite sizable. In order to meet the requirement for high-speed analysis of such big data, we parsed the raw data into a well-defined structure and imported it into the column-oriented database server, ClickHouse cluster cloud services, which allows real-time online analysis using SQL queries. We extracted 142 rows from the original JSON document. The table structure and record samples are presented in our supplementary material[1]. All the analyses and scripts are based on sending SQL queries to the ClickHouse database server and are available for replication.

In this paper, explorations are based on *860 million* records generated in a time-series manner in 2020. The number of GitHub accounts we found in 2020 was as high as 14.54 million.

Specifically, we performed a case study of CNCF. CNCF is a vendor-neutral community to build sustainable ecosystems for cloud native software, the most notable rockstar project among which is Kubernetes (79.7K stars). We referred to different maturity levels of sandbox, incubating, and graduated projects labeled by CNCF landscape. Almost all the CNCF projects are hosted on GitHub. 16 sandbox projects and 2 incubating projects were excluded because we could not retrieve their GES records during 2020 (changed repository names, created after 2020, have no activity at all, etc). There are over 1.1 million records and 103,787 accounts found in 79 CNCF projects in our sample. Detailed information is presented in Table 1.

To better address the *Where* question, we used multiple data sources aside from (1) the timestamp information in the GES, and also (2) the timezone information from each commit in git-logs[33].

[1]https://doi.org/10.5281/zenodo.6811484

TABLE I
Dataset Overview

	records	repos	accounts	contributors	bots
all	863,415,606	54,205,696	14,540,459	4,519,712	16,036
cncf	1,131,976	79	103,787	24,978	121
grad.	596,537	16	61,676	15,189	59
incub.	208,790	20	26,520	5,365	36
sandb.	326,649	43	38,901	7,146	56

IV. Research Method

A. Measuring the level of Activity

Activity of contributors A_c on GitHub is determined by multiple behavioral data (issue tracker data, pull request data, etc). We proposed an activity evaluation model that was composed of a set of behaviors. For a certain dataset D during the time period from t_{start} to t_{end}, a set of behaviors B_k can influence the activity level of GitHub contributor C. The $AGGREGATE$ function groups and aggregates behaviors by C in dataset D, with weight values W_{B_k} assigned to each behavior. The activity of C, A_c, is an average value during a certain period. It is calculated by the summation of weighted behaviors divided by days (366 days in our case):

$$A_c = \sum_1^k \frac{W_{B_k} AGGREGATE(B_k, D)}{t_{end} - t_{start}} \quad (1)$$

The behavior B and weight W are designed to be abstractive. To make it simple in this study, we heuristically selected 5 collaborative behaviors $B_k = \{b_1, ..., b_5\}$ that apply to all projects to comprehensively calculate a contributor's activity under the pull-based contributing workflow, as shown in Table 2. The five behaviors account for most of the collaboration scenarios when making contributions on GitHub. We do not select *fork* and *star* because from the perspective of a project, these two kinds of behaviors tend to be one-off (a contributor usually only forks or stars a project once) and involve no interaction; others like push, release, wiki have similar problems: (i) no collaboration; (ii) required special permission; (iii) not applied by many GitHub repos, i.e. many projects don't enable wiki function. Specifically, each behavior is related to a GitHub event type (*type* field in the GES) and an actor interaction action (*action* field in the GES). For example, the behavior *open issue* corresponds to IssueEvent when *action* = 'opened.' Communications can happen both in issues and pull requests in the form of a comment. It is worth mentioning that in this activity model, we only consider commits from the pull-based collaboration workflow and omit the commits that directly push code to a repository, which would generally be considered as a deprecated collaboration behavior and only part of the roles can perform.

Behavior weight values are actually subjective to different communities, based on different situation. For example, projects like JOSS might value activity in Issues more than in PRs. We considered factor analysis and EWM. The difficulties lie in (i) being hard to perform calculations on such sizable datasets; (ii) still lacking interpretability. To apply the large-scale scenario in this study, we assign equal weight values to each behavior W_{B_k}.

TABLE II
THE ACTIVITY MODEL

behavior	event type	action	description
comment	IssueCommentEvent	created	comment in issues or PRs
open issue	IssuesEvent	opened	open an issue
open pr	PullRequestEvent	opened	submit a pull request
review pr	PullRequestReviewCommentEvent	created	initiate review comments for PRs
pr merged	PullRequestEvent	closed&pull_merged=1	the PR is merged into the repository

B. Identifying bots

GitHub allows bots to automate tasks and improve collaboration workflow. Bot behaviors are not negligible but not useful to the understanding of human traits. On the other hand, bot behaviors can also provide insights into the modern OSS development. It is helpful to identify bots and observe their behaviors; it is also important to exclude bots when analyzing real developers' work patterns. In our study, we define bots as GitHub accounts with usernames that end with either '[bot]' or 'bot.' 16,036 bot accounts were identified in this way. For projects in CNCF, the first author also performed a manual check to exclude false positives and identify the missing bots.

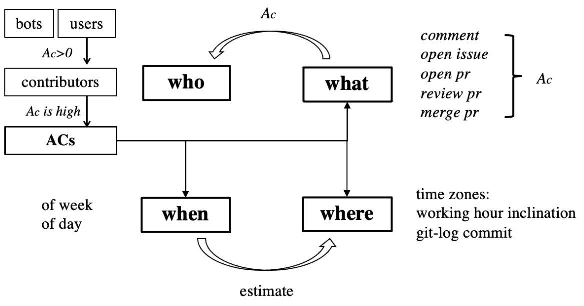


Fig. 1. Basic Structure

C. Defining different roles

Among all the contributors that show up in a project, only a small group can be defined as 'active.' A common situation is where developers only express interest in a project, leave some comments or even submit one or two pull requests and then disappear. A few people undertaking most of the work is a familiar story in open source. For example, three developers submitted 73% of the code in Bootstrap in 2017[34]. Code is not the only responsibility of contributors in the OSS community, as was discussed above. In another example, two maintainers from Godot needed to provide responses to over 120 questions in the course of a year[34].

The activity model can reflect contributors' activity levels on GitHub from multiple perspectives. However, what is the threshold to distinguish the most active ones? The demarcation should be established according to the workload distribution. This study aims to seek out core contributors who are doing the majority of the work[35] and analyze their characteristics. To this end, we defined different roles based on numerous GitHub accounts in our dataset: bots, users, contributors, active contributors (ACs), as illustrated in Figure 1. First, GitHub accounts can be divided into bots and users. Contributors are defined as users who have initiated at least once one of the five contributing behaviors defined in the activity model, meaning $A_c > 0$. We used this parameter to filter out approximately two-thirds of non-contributors. The number of contributors in each case is listed in Table 1. ACs are top x contributors by the A_c ranking, where x is the proportion of contributors which is set according to the activity distribution. It is helpful to filter out sporadic contributors and only focus on the most active ones, as we also rely on sufficient behavioral records to estimate contributors' working hour patterns and time zones.

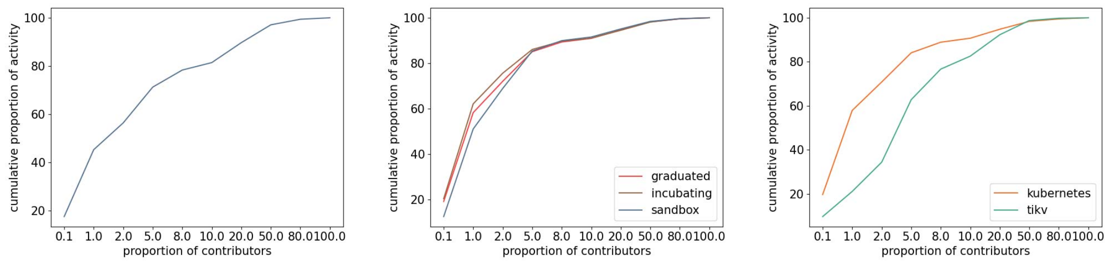
Note that A_c is independent of a specific project, representing a contributor's collaboration activity across all GitHub projects. However, in the present case study, active contributors should represent developers who are most devoted to that specific project or community. Their behaviors in other projects should not be taken into account. Thus, when calculating A_c in the case study, we only factor in the behaviors within dataset D_{case}.

D. Extracting Time Zones

To determine the geographical distribution of GitHub contributors, we combined two methods:

1) *Time zone estimation*: the GES records provide a *created_at* field, which stores the standard UTC timestamp. Here we present an estimating method based on the hypothesis that OSS developers' contributing patterns generally fall within the hours from 10 am to 6 pm, which is based on the research conclusion made by Maëlick Claes in 2018[18]. We aggregated all the timestamps per contributor and extracted the consecutive 8 hours of the day when a contributor is most active, presuming that the start of this period is the contributor's 10 am local time and the end of this period is the contributor's 6 pm local time. According to the variance from the UTC time, we can roughly determine a contributor's time zone.

It should be noted that using this method, it is difficult to accurately estimate the time zone of a single contributor, since the behavior of one contributor can be sporadic and does not conform to a typical circadian work pattern. But it is effective in estimating the proportion of contributors in each time zone so as to further deduce the global coverage of contributors and the degree of collaboration globalization within a project or a community consisting of a group of projects.



(a) GitHub (b) CNCF (c) Kubernetes and Tikv

Fig. 2. Activity distribution

2) Time zone in commit: In a Git repository, each commit provides a timestamp and a time zone. This is obtained from the configuration of the computer where the commit took place[33], [36]. The time zone is specified in integers as a time difference from GMT. For example, a commit that is submitted by a developer who lives in Beijing will be postfixed with a +0800 following the message date. To simplify, we rounded off the non-integer time zones, such as +0530 that is used in India, to the floor integer and merged the time zone as GMT+5[33]. Note that the commit information covers only part of the contributors, since not all of the contributors have to necessarily commit to a repository, and contributors who have committed might not be active since. We use time zone in commit to perform mutual verification of a project's location dispersion .

V. Results and Empirical Analysis

In this section, we present our results from both a global perspective and with a focus on specific cases. For the case study, we performed contrast analyses according to the different maturity levels of projects: sandbox, incubating, and graduated. We also selected two star projects for comparative analyses: Kubernetes and Tikv. The selection was based on regional and scale consideration, the prior knowledge is Kubernetes is donated by Google and actively maintained by a large community(mainly in the western world), while Tikv is donated by PingCap and actively maintained by a relatively smaller community (mainly in the eastern world).

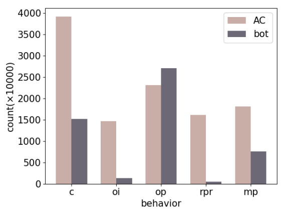
A. Q1: Who is active and how active are they?

Bots. We sorted the A_c for all accounts and found that the most active accounts were actually bots carrying out a large number of continuous operations. Among the 14.54 million GitHub accounts that showed behaviors in 2020, a total of 16,036 bots were identified. GitHub Apps (ending in '[bot]') are one kind of bots that are commonly deployed on the platform. They run on the server end, thus serving many projects concurrently and demonstrate an extremely high level of activity. The activity proportion of bots was 33% among all accounts, meaning that although the number of bots is not large compared to all accounts, they are extremely active overall.

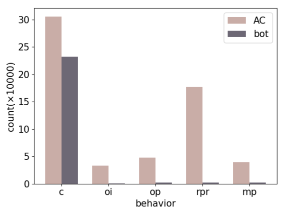
Active Contributors (ACs). First, we excluded bots from real user accounts, since their high presence greatly affects the result. We then extracted contributors by filtering out the accounts with $A_c = 0$. We identified 4,519,712 contributors across GitHub, with 24,978 of them being CNCF contributors, then observed the activity distribution of contributors to find ACs (the group of contributors with the highest A_c). Figure 2 shows the activity distribution of different cases. There is an often mentioned indicator, 'bus factor,' to describe the number of people who have to be hit by a bus (or just leave) before the project is in serious trouble[35], [37]. Activity distribution can also reflect this kind of risk. We sorted all contributors according to A_c in descending order, manually selected several sampling points of the proportion of top contributors: 0.1, 1, 2, 5, 8, 10, 20, 50, 80, and calculated the corresponding cumulative proportion of the total activity. Figure 2 (a) illustrates the overall activity distribution. The key points are: 5% contributors correspond to 71% activity; 10% contributors correspond to 81% activity; 20% contributors correspond to 89% activity.

We found large-scale OSS collaborations conform to a highly imbalanced distribution. We took **20%** as the baseline to distinguish ACs (meaning, x=20), as it covers most of the contributions for all our cases, as will be seen in Figure 2 (b) and (c). The total number of active contributors identified was 902,560.

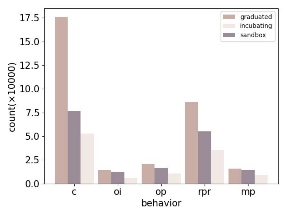
In CNCF, we found the distribution was characterized by a higher imbalance, whether it was in graduated, incubating or sandbox projects. Their curves start to overlap at around 5% of contributors, reaching over 80% of the activity, whereby 10% contributors correspond to around 90% activity, which lines up with the rockstar project Kubernetes, where 10% contributors accounted for 90.7% of total activity. As we can see in large-scale projects, output and input are highly imbalanced, reaching as high as 90/10. Only a small group of people (10%) drive the progress (90%), and most of the rest of developers remain peripheral. 20% contributors (AC threshold) accounted for the majority of the activity (95% for all three, 3037, 1073, 1429 ACs were identified in graduated, incubating, sandbox projects, respectively). If the bus factor threshold[35] of Kubernetes is 50% (meaning the smallest number of people that make 50% of contributions), this gigantic open source project has a high risk of facing project breakdown if less than 1% of the contributors quit



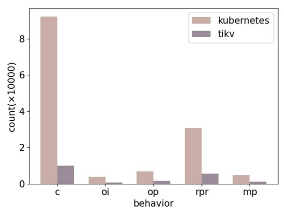
(a) ACs *vs* bot on GitHub



(b) ACs *vs* bot in CNCF



(c) ACs by maturity



(d) ACs by project

Fig. 3. Behavior distribution
Note: c - *comment*; oi - *open issue*; op - *open pr*; rpr - *review pr*; mp - *pr merged*

(around 50 developers). However, the risk of Tikv is explicitly lower, being more evenly distributed, as 5% contributors only accounted for 62.8% activity.

Who: Bots are the most active accounts on GitHub. Besides bots, only a small group of contributors are active. The activity distribution is noted for an imbalance among GitHub contributors. 20% of the GitHub contributors accounted for almost 90% of the total activity.

B. *Q2: What are the active contributors doing?*

When addressing the what, when, and where, we only consider ACs (which are distinguished by the activity threshold of the top 20% contributors). But we will discuss at greater length the behavior of bots. The introduction of the dependabot[bot] (Dependabot) is as follows:

"Creates pull requests to keep your dependencies secure and up-to-date. 7,734,086 pull requests merged, and counting!"

In 2020, Dependabot opened over 17 million pull requests, among which around 1.3 million were merged, according to our dataset.

We compared the behaviors of ACs and bots. Figure 3 (a) shows the count of the five behaviors of all ACs and all bots; Figure 3 (b) shows the count of the five behaviors of ACs and bots in CNCF. Overall, bots were integrated to open pull requests and comment on issues/pull requests, with some bots even merging pull requests, but only a few bots were used to open issues and review pull requests. We also noticed that in 2020, 10K bots opened more pull requests than 900K ACs, from which we empirically deduce that making change requests for new updates is the most preferred automation area. The usage of bots varied across communities. We found CNCF almost exclusively used bots to do the commenting work.

We also compared ACs' behaviors by maturity levels and by single projects, as shown in Figure 3 (c) and (d). Despite the number of graduated projects (16) being less than incubating (20) and sandbox projects (43), the number of behaviors was still far more than the other two. As we can see in CNCF, the most frequent behaviors were commenting and reviewing, with the less frequent behaviors being opening issues and pull requests. It stands to reason that *comment* was the most frequently occurring behavior in all developer scenarios, since it cost less (in time, energy, etc.) compared to other behaviors most of the time. However, the large number of *review pr* behaviors, coupled with comment behaviors, empirically indicates that ACs are mostly maintainers who are responsible for community response and change review.

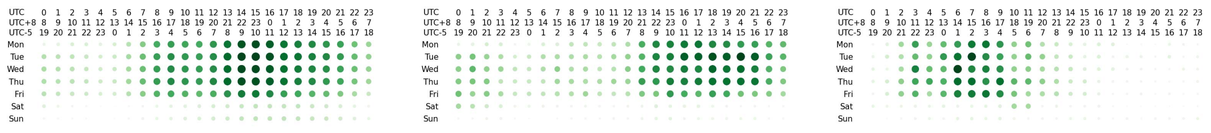
What: Commenting is the most popular behavior, as it is a lighter action compared to contributing code. In a unit of the community, ACs seem to spend more energy on reviewing besides commenting, reinforcing their role as maintainers.

C. *Q3: When are the active contributors active?*

We aggregated the activity of ACs by the hour and day of the week. Figure 4 demonstrates the working hour distribution on GitHub, Kubernetes and Tikv, respectively. We used the standard UTC timestamp information in the GES records. For all active contributors in Figure 4 (a), the dot grows bigger from 8 am to 9 pm UTC time. It is quite a long time frame indicating contributors were collaborating globally. As we superimpose the other two timelines to compare, we can see the period also corresponds to 3 am to 4 pm of the U.S.Central Standard Time (UTC-5), 4 pm to 5 am of the China Standard Time (UTC+8). It can be reasonably seen that it is the period when people from the main software development productivity areas (Europe, North America, Asia) have the most hours staying awake. From an overall point of view, the working time pattern is geared more toward the daytime in Europe. In Kubernetes, the most active period was from 1 pm to 11 pm UTC time, which was 8 am to 6 pm UTC-5, and 10 pm to 9 am UTC+8. Tikv had a shorter active time frame (from 3 am to 9 am UTC), which can be empirically considered as a not so globally collaborative project. We observed a clear disparity between the distribution of Kubernetes being more inclined to the North American daytime, and the distribution of Tikv was more inclined to the China daytime hours.

We found that over the course of a week contributors were evidently less active on weekends than weekdays, which bore out the findings of the GitHub Octoverse report[38] that more developers use GitHub to work instead of developing merely based on interests. Activity on Friday was slightly decreased compared to other weekdays.

We also compared the activity percentage on weekdays and weekends for projects at different maturity levels. CNCF had a similar pattern where contributors are much more active on



(a) GitHub (b) Kubernetes (c) Tikv

Fig. 4. Working hour distribution.

Aggregated according to the standard UTC time. The horizontal axis represents the 24 hours in a day. Apart from the 24 hours in UTC time, we also put the UTC+8 and UTC-5 timeline, representing the circadian working hours in Asia and Europe. The vertical axis represents 7 days in a week. The size of the dot represents the total activity in the hour of the week and is re-scaled by min-max normalization.

weekdays, meaning they were spending the office hours to contribute. Incubating projects have a lower activity percentage on weekends (5.8%, compared to 7.4% and 7.5%), indicating that active contributors in incubating projects were even more concentrated on weekdays.

We then extracted each AC's activity pattern of the day, estimated the contributor's local time zone using the method illustrated in Section 4: we considered the consecutive 8 hours of the day during which a contributor is most active as the contributor's local daytime from 10 am to 6 pm, then translated the hours to the corresponding estimated time zone axis, and merged the activity of all contributors on that axis. Figure 5 shows the circadian working pattern of all ACs after merging their estimated local time zone axes. Contributors typically start working at 9 am, with a rapid rise in work activity, then have a short lunch break around 1 pm, reach a productivity peak at 5 pm, then the activity starts to drop off. Still, compared to other occupations[39], [40], active contributors to OSS maintain consistent productivity levels until late in the evening. Figure 5 also elucidates that Fridays are less productive, especially during the afternoons.

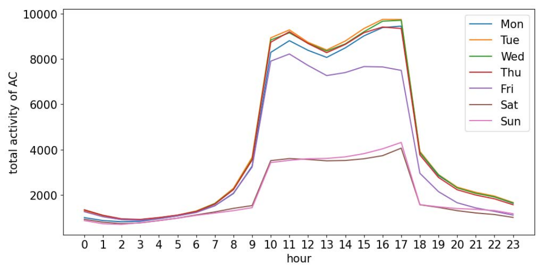


Fig. 5. The circadian rhyme of ACs

The abscissa represents 24 hours in a day of an AC's estimated local time. The ordinate represents the summation of all the ACs' activity in that hour.

When: The working time of active contributors correlate with traditional office hours, being significantly more active on weekdays than weekends, and are relatively less active on Friday afternoons. During the course of one day, there are two activity peaks at 10 am and 5 pm, with ACs maintaining a relatively high productivity level until late.

D. Q4: Where are the active contributors?

Figure 6 is the time zone distribution of all ACs on GitHub, calculated by the time zone estimation method. The Americas (Canada, the United States, and South America) had the largest total distribution of ACs, from UTC-8 to UTC-3. Although the proportion in a single time zone was not the highest, the overall proportion of ACs in this region was as high as 32%. Europe had the highest proportion of ACs in a single time zone, UTC+2, the proportion reached 10%. In general, the number of ACs in Asia was still relatively small, but a small peak was noted in UTC+8, indicating that contributors in China represent a greater number than other Asian countries.

To validate the statistical correctness and refine our results, we combined the time zone information per commit in the git-log of the case study projects. We calculated the regional proportion of ACs and commit contributors (CCs), as illustrated in Table 3, and observed a consistent regional distribution trend. (1) UTC-8 to UTC-3, (2) UTC to UTC+3, and (3) UTC+5 to UTC+9 are the three main regions where contributors are located, roughly corresponding to the Americas, Europe and Asia. The common trends of the proportion of contributor are (1)>(2) > (3). Tikv was an exception that was absolutely dominated by contributors in (3), with few contributors in (1) and (2). Regardless of whether observing the distribution of AC or CC, the order remained consistent. The similarity in the time zone distribution pattern validates the time zone estimation. While the estimation supplement the unattainable geographical location for many contributors within a community on the whole.

The centralized distribution of Tikv corroborates the fact that it was donated by PingCap, a fast growing Beijing-based database vendor. The time zone estimation approach is based on a developer's most active period, demonstrate developers' working schedule inclination; thus, we call it the 'working inclination time zone.'

Where: The Americas and Europe still had the most ACs, from the global perspective. The results calculated by the commit time zones are in line with the results calculated by activity periods estimation in the case study, proving the validity of this finding.

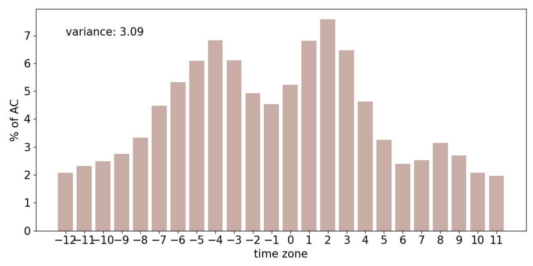


Fig. 6. Timezone distribution of all ACs on GitHub

The abscissa lists 24 time zones. The ordinate represents the proportion of ACs in that time zone.

TABLE III
Region Proportion by AC and CC

Case	(1) Americas	(2) Europe	(3) Asia
CNCF	35/43	26/33	19/22
Graduated	35/45	23/28	21/25
Incubating	40/45	26/33	16/19
Sandbox	30/37	31/40	18/22
Kubernetes	36/43	22/30	21/25
Tikv	3/13	9/4	72/81

(1) is from UTC-8 to UTC-3; (2) is from UTC to UTC+3; (3) is from UTC+5 to UTC+9. The left of '/' represents the proportion of AC (%) and the right of '/' represents the proportion of CC (%). The proportion values are rounded off decimals.

VI. Discussion

In this section, we provide additional discussion of the results presented in the previous sections.

Automation on GitHub is widely applied. We found bots were serving numerous projects and were extremely active (33% across all accounts). They were widely adopted to automate the work related to repository updates and community responsiveness. Our results show the most frequent behaviors were comment and open pull requests. When analyzing GitHub behaviors, the bot factor will seriously affect the result if bots are not excluded from the sample. The finding also instructs tool builders that automation–related tools are becoming increasingly important to improving the collaboration workflow. Platforms should also take notice to provide standard open APIs in keeping with the trend.

A small group of people are doing most of the work. Our results are in line with Nadia's observation in 2020[34], in which she pointed out that open source is transforming from a large-scale collaboration to individual efforts. We found less than 20% of contributors made more than 80% of the contribution. There is even a 90/10 activity distribution in large and famous projects. In particular, maintainers are facing the dilemma of processing the frequent entry requests, as we found active contributors are taking most of their energy to respond to issues and review pull requests. Project leaders should take care of their active contributors, building a better community structure to mitigate the workload of maintainers.

A high bus factor risk exists even considering developers' willingness to contribute. The bus factor, also known as the truck factor (TF) calculates "*the number of people on your team that would have to be hit by a truck (or quit) before the project is in serious trouble*"[41]. The study of Steinmacher *et al.* on quasi-contributors[2] reported that the small TF calculated for some projects is related to not considering external contributors whose pull requests are refused. Our findings suggest that after considering other contributions not limited to merged pull requests, many projects still have a high risk of trouble with a small proportion of contributors leaving. Online production is shifting to solitary work[34], with the bus factor risk being a critical problem that many mature OSS projects are facing.

OSS contributers are contributing on office hours **.** We found contributors were much more active on weekdays. This may be attributable to the new landscape of OSS[42], [43], in which companies are key players. Compared to graduated and sandbox projects, contributors in incubating projects were more concentrated on weekdays, which may be caused by companies behind some of the projects putting in more effort to hatch out a project to a real, mature community.

Friday afternoon is less productive, and lunch break is between 12 am to 2 pm. We found a typical work pattern of active contributors especially on weekdays, as presented in Figure 5. The activity is relatively decreased on Friday afternoons. The work pattern also reveals that lunch breaks are during 12 am to 2 pm. Our results align with the findings by Claes *et al.*[18], who reported lunch hour mostly happens at 12 o'clock. Given the working customs implied in our result, a well-governed community or company leaders should avoid setting meetings or deadlines, etc., during these time periods.

America and Europe still take the lead. Although GitHub Octoverse reported 30.7% users were located in Asia[38], our results showed that the proportion of active contributors in Asia was not as high considering the working inclination time zone. This may be attributed to scheduling variance, but the trend that the Americas and Europe have more ACs is in line with most of our cases.

VII. Implication

The RQs are raised to reveal core contributors, their behaviors and work patterns. From a macro perspective, the findings reveal the general open collaboration trend of modern digital goods, especially the OSS ecology on GitHub, which also provides baselines for specific cases. The big picture is that 20% of contributors account for 90% activity, where Kubernetes is even more imbalanced compared to this. For specific cases, it's crucial for project owners, community managers, foundation PMCs to seek out the most active/devoted contributors (not limited to code contributions), knowing their workload, preferred work rhythms, and possibly time zone distribution. We provided a wh-question analysis framework for communities hosted on GitHub to better command control of their activity status and roles.

Implications for practitioners. As high bus factor risk and contributor/maintainer burnout in OSS has been brought up in recent years, the 90/10 imbalance found in this study validated and quantified this phenomena, sounding the alarm for project leaders to identify and take care of the small group of contributors who may be of importance and may be overworked; Knowing the follow up questions of *what, when and where* of these contributors implies project leaders to make better decisions, i.e., the work rhythms and timezone distribution imply a community manager to set appropriate meeting times, event venues, even sprint deadlines. Finally, the emergence of bots also implies tool builders that RPA (Robotic Process Automation) is becoming the trend on GitHub collaboration. Automation is especially popular as the form of comments and PRs openings, which may bring implications to bots developers when optimizing collaboration processes.

Implications for researchers. The global view of this study present the GitHub collaboration trend by quantitative analysis of the GES. The findings serve as a call for validations and further investigations of reasons behind those conclusions from qualitative perspective. Additionally, the activity model proposed in this study serves as an initial step to quantify contributions beyond just looking at the code, where behaviors from multiple platforms and the activity time decay factor could be considered in future research.

VIII. Threats To Validity

There are always limitations in an empirical study. First, we acknowledge that the value of A_c can be affected by the contribution behavior types and the weight assigned. To ensure the soundness of our assumptions, we relied on expert reviews and continuous discussions until we reached a consensus. Contributions in this study are restricted to GitHub behaviors, whereas people can spend their time contributing by answering emails or writing documentation outside GitHub[44]. However, our activity model considers the several imperative behaviors as representing pull-based contributions. As a mainstream collaboration platform, contributions on GitHub can, to some extent, map into the contributions in the overall community.

Another threat is that the maturity status of CNCF projects is dynamic. The project status we referred to was from the CNCF landscape as of September 2021, while they could have a different status in 2020. For example, *etcd* graduated from incubating in October 2020. There is a lag in a few projects between how we classify them and their true maturity. The explanation is when comparing the ***W's*** of different maturity statuses, we were looking at the result of exactly why it became the status, not what it was.

The time zone distribution method is not for depicting the time zone of a single contributor whose behavior can be sporadic and specific. We selected active contributors to ensure sufficient number of timestamps, thus improve the accuracy from the global view. We empirically determine that the most typical active period is 10 am to 6 pm local time, which may not apply to individuals due to different work-life habits. However, the method is meaningful in statistical dimensions for project leaders to know how geographically diverse their projects are. We accept that the statistical distribution is also affected by the predefined active period. In light of this, we refer to previous work[18] to settle it as 10 am to 6 pm. In summary, our method is helpful in deciding on community meeting times. Although a community member's time zone may be misjudged geographically due to the deviations from traditional circadian working time, the person in charge of the meeting schedule can at least be sure of the member's working time inclination.

IX. Conclusion

In this paper, we performed a large-scale study to investigate the ***four W's*** – Who, What, When, and Where – based on 14.54 million GitHub accounts, along with a case study based on 104K GitHub accounts in CNCF. We were motivated to find out who is active to contribute, observe their behavior patterns and study their active period, then infer their time zone location. We proposed an activity model composed of multiple behaviors, upon which we defined qualitative roles of contributors and active contributors on GitHub. Active contributors are a vital group of a community. The panoramic view of ACs sheds some light on the modern large-scale distributed collaborations of GitHub. The findings in the case study also provide some guidance to tool builders, venture capitals and especially community managers, and can be further used as a framework to gain some insights relative to the activity status of projects or organizations hosted on GitHub.

In future work, we will refine the activity evaluation model with more contribution behaviors taken into account out of the GitHub platform. We need to combine data from other important community channels like mailing-list, documentation hosting platform, and blog post sites. We also intend to study the behaviors of bots, form a high accuracy method to exclude them from human developers, since we found bot behaviors were extremely abnormal and can largely interfere with human-centered analyses. Finally, we also want to combine surveys and interviews to better understand the role of ACs and their behaviors.

X. Acknowledgment

We would like to thank all the team members and experts for their support and suggestions. The data analysis is supported by an open source project named OpenDigger[2]. A special word of thanks to Aleksey Zaitsev for his invaluable help with sentence polishing.

References

[1] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim 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*, pages 1277–1286, 2012.

[2]https://github.com/X-lab2017/open-digger

[2] Igor Steinmacher, Gustavo Pinto, Igor Scaliante Wiese, and Marco Aurélio Gerosa. Almost there: A study on quasi-contributors in open-source software projects. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 256–266. IEEE, 2018.

[3] Aftab Iqbal. Understanding contributor to developer turnover patterns in oss projects: A case study of apache projects. *International Scholarly Research Notices*, 2014, 2014.

[4] Marcus Vinicius Bertoncello, Gustavo Pinto, Igor Scaliante Wiese, and Igor Steinmacher. Pull requests or commits? which method should we use to study contributors' behavior? In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 592–601. IEEE, 2020.

[5] Didi Surian, David Lo, and Ee-Peng Lim. Mining collaboration patterns from a large developer network. In *2010 17th Working Conference on Reverse Engineering*, pages 269–273. IEEE, 2010.

[6] Yue Yu, Gang Yin, Huaimin Wang, and Tao Wang. Exploring the patterns of social behavior in github. In *Proceedings of the 1st international workshop on crowd-based software development methods and technologies*, pages 31–36, 2014.

[7] Omar Elazhary, Margaret-Anne Storey, Neil Ernst, and Andy Zaidman. Do as i do, not as i say: Do contribution guidelines match the github contribution process? In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 286–290. IEEE, 2019.

[8] Danaja Maldeniya, Ceren Budak, Lionel P Robert Jr, and Daniel M Romero. Herding a deluge of good samaritans: How github projects respond to increased attention. In *Proceedings of The Web Conference 2020*, pages 2055–2065, 2020.

[9] Mohammad Azeez Alshomali, John R Hamilton, Jason Holdsworth, and S Tee. Github: factors influencing project activity levels. In *Proceedings 17th international conference on electronic business. Dubai, United Arab Emirates*, pages 295–303, 2017.

[10] Georgios Gousios and Diomidis Spinellis. Ghtorrent: Github's data from a firehose. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 12–21. IEEE, 2012.

[11] [Online]. Gh archive. http://www.gharchive.org/.

[12] Jean-Gabriel Young, Amanda Casari, Katie McLaughlin, Milo Z Trujillo, Laurent Hébert-Dufresne, and James P Bagrow. Which contributions count? analysis of attribution in open source. *arXiv preprint arXiv:2103.11007*, 2021.

[13] Mariam Guizani, Amreeta Chatterjee, Bianca Trinkenreich, Mary Evelyn May, Geraldine J Noa-Guevara, Liam James Russell, Griselda G Cuevas Zambrano, Daniel Izquierdo-Cortazar, Igor Steinmacher, Marco A Gerosa, et al. The long road ahead: Ongoing challenges in contributing to large oss organizations and what to do. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2):1–30, 2021.

[14] Igor Steinmacher, Marco Aurelio Graciotto Silva, Marco Aurelio Gerosa, and David F Redmiles. A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology*, 59:67–85, 2015.

[15] Amanda Lee, Jeffrey C Carver, and Amiangshu Bosu. Understanding the impressions, motivations, and barriers of one time code contributors to floss projects: a survey. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 187–197. IEEE, 2017.

[16] Edson Dias, Paulo Meirelles, Fernando Castor, Igor Steinmacher, Igor Wiese, and Gustavo Pinto. What makes a great maintainer of open source projects? In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 982–994. IEEE, 2021.

[17] Carsten Kolassa, Dirk Riehle, and Michel A Salim. The empirical commit frequency distribution of open source projects. In *Proceedings of the 9th International Symposium on Open Collaboration*, pages 1–8, 2013.

[18] Maëlick Claes, Mika V Mäntylä, Miikka Kuutila, and Bram Adams. Do programmers work at night or during the weekend? In *Proceedings of the 40th International Conference on Software Engineering*, pages 705–715, 2018.

[19] Qi Xuan and Vladimir Filkov. Building it together: Synchronous development in oss. In *Proceedings of the 36th International Conference on Software Engineering*, pages 222–233, 2014.

[20] Frederike Ramin, Christoph Matthies, and Ralf Teusner. More than code: Contributions in scrum software engineering teams. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 137–140, 2020.

[21] Georgios Gousios, Martin Pinzger, and Arie van Deursen. An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering*, pages 345–355, 2014.

[22] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining github. In *Proceedings of the 11th working conference on mining software repositories*, pages 92–101, 2014.

[23] Thaıs Mombach and Marco Tulio Valente. Github rest api vs ghtorrent vs github archive: A comparative study, 2018.

[24] [Online]. Github rest api. https://docs.github.com/en/rest.

[25] Hudson Borges, Andre Hora, and Marco Tulio Valente. Understanding the factors that impact the popularity of github repositories. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 334–344. IEEE, 2016.

[26] Saya Onoue, Hideaki Hata, and Ken-ichi Matsumoto. A study of the characteristics of developers' activities in github. In *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, volume 2, pages 7–12. IEEE, 2013.

[27] Jailton Coelho, Marco Tulio Valente, Luciano Milen, and Luciana L Silva. Is this github project maintained? measuring the level of maintenance activity of open-source projects. *Information and Software Technology*, 122:106274, 2020.

[28] Antonio Cerone. Learning and activity patterns in oss communities and their impact on software quality. *Electronic Communications of the EASST*, 48, 2013.

[29] Elizabeth A Sall and Chandra R Bhat. An analysis of weekend work activity patterns in the san francisco bay area. *Transportation*, 34(2):161–175, 2007.

[30] Johannes Wachs, Mariusz Nitecki, William Schueller, and Axel Polleres. The geography of open source software: Evidence from github. *arXiv preprint arXiv:2107.03200*, 2021.

[31] Bence Kollanyi. Automation, algorithms, and politics— where do bots come from? an analysis of bot codes shared on github. *International Journal of Communication*, 10:20, 2016.

[32] [Online]. Chaoss metrics - contributor location. https://chaoss.community/metric-contributor-location/.

[33] Jesus M Gonzalez-Barahona, Gregorio Robles, and Daniel Izquierdo-Cortazar. Determining the geographical distribution of a community by means of a time-zone analysis. In *Proceedings of the 12th International Symposium on Open Collaboration*, pages 1–4, 2016.

[34] Nadia, Eghbal. *Working in Public: The Making and Maintenance of Open Source Software*. Stripe Press, 2020.

[35] [Online]. Chaoss metrics - bus factor. https://chaoss.community/metric-bus-factor/.

[36] Premkumar Devanbu, Pallavi Kudigrama, Cindy Rubio-González, and Bogdan Vasilescu. Timezone and time-of-day variance in github teams: an empirical method and study. In *Proceedings of the 3rd ACM SIGSOFT International Workshop on Software Analytics*, pages 19–22, 2017.

[37] Mishkin Berteig. *Agile Advice - Creating High Performance Teams In Business Organizations*. Berteig Consulting Inc., 2015.

[38] [GitHub]. The 2020 state of the octoverse. https://octoverse.github.com/.

[39] Richard Rogerson. Understanding differences in hours worked. *Review of Economic dynamics*, 9(3):365–409, 2006.

[40] Linda A Bell and Richard B Freeman. The incentive for working hard: explaining hours worked differences in the us and germany. *Labour Economics*, 8(2):181–202, 2001.

[41] Laurie Williams and Robert R. Kessler. *Pair Programming Illuminated*. Addison-Wesley Professional, 2003.

[42] Marco Gerosa, Igor Wiese, Bianca Trinkenreich, Georg Link, Gregorio Robles, Christoph Treude, Igor Steinmacher, and Anita Sarma. The shifting sands of motivation: Revisiting what drives contributors in open source. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1046–1058. IEEE, 2021.

[43] Gregorio Robles, Igor Steinmacher, Paul Adams, and Christoph Treude. Twenty years of open source software: From skepticism to mainstream. *IEEE Software*, 36(6):12–15, 2019.

[44] André N Meyer, Laura E Barton, Gail C Murphy, Thomas Zimmermann, and Thomas Fritz. The work life of developers: Activities, switches and perceived productivity. *IEEE Transactions on Software Engineering*, 43(12):1178–1193, 2017.