2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)

An Empirical Study of Multiple Names and Email Addresses in OSS Version Control Repositories

Jiaxin Zhu
Institute of Software, Chinese Academy of Sciences
University of Chinese Academy of Sciences, China
zhujiaxin@otcaix.iscas.ac.cn

Jun Wei
State Key Lab of Computer Science,
Institute of Software, Chinese Academy of Sciences
University of Chinese Academy of Sciences, China
wj@otcaix.iscas.ac.cn

***Abstract*—Data produced by version control systems are widely used in software research and development. Version control data users always use the name or email address field to identify the committer or author of a modification. However, developers may use multiple names and email addresses, which brings difficulties for identification of distinct developers. In this paper, we sample 450 Git repositories from GitHub to study the multiple names and email addresses of developers. We conduct a conservative estimation of its prevalence and impact on related measurements. We merge the multiple names and email addresses of a developer through a method of high precision. With the merged identities, we obtain a number of interesting findings, e.g., about 6% of the developers used multiple names or email addresses in more than 60% of the repositories, and they contributed about half of all the commits. Our impact analysis shows that the multiple names and email addresses issue cannot be ignored for the basic related measurements, e.g., the number of developers in a repository. Our results could help researchers and practitioners have a more clear understanding of multiple names and email addresses in practice to improve the accuracy of related measurements.**

***Index Terms*—version control, data, multiple names, multiple email addresses, pattern, impact**

I. Introduction

Data recorded in software development tools, e.g., version control systems and issue tracking systems, are often used in software engineering research and practice [1], [19], [27], [37], [49]. In many cases, distinct developers have to be identified in the data. For example, to measure the experience of a developer in a project [14], [18], first, the related developers for each record have to be identified. In many IT systems, it is easy to identify a person by her unique ID, e.g., passport number, driving license number in the government IT systems. However, in most of the software development tools, developers do not have to follow the real identity policy, and there are always no strict IDs [24]. It means that it is very difficult to accurately identify the developers in the data recorded by the tools, which may introduce bias of the analysis results.

Version control system is one of the most commonly used tools in software development. Version control data are amongst the most widely used software engineering data in research and practice, and many analyses of the data require identification of developers, e.g., code contribution analyses, developer/reviewer recommendation and community construction [39], [46]. Most version control systems, including the most popular Git, have data fields of name and email address in the record of a commit. Using these two fields is a common way to identify the developers in most cases when other data are not included. However, there are a number of risks for the identification accuracy. Different developers may have and use the same name, and sometimes, a developer may use multiple names. A developer may use multiple email addresses too. In the rest of the paper, we abbreviate the ***multiple names/multiple email addresses of a developer*** with ***MNE***.

We preliminarily examine 629 research papers in the proceedings of ICSE [7], [13], [40], FSE [5], [25], [51] and MSR [17], [22], [47] of the past three years through manual check of whether MNE is involved, mentioned and addressed, and find at least 8 papers where the authors faced the issue in using version control data. Among them, six papers do not elaborate whether or how they address the issue [15], [26], [33], [38], [45], [48], and the other two papers do [8], [20]. The examination result implies that researchers do not have a unified understanding of the MNE issue. Some may consider MNE as a trivial issue and have not paid enough attention to it. Since version control data are widely used, it is important to enrich our knowledge of MNE and come up with effective strategies to handle it.

In this paper, we conduct an empirical study of MNE in Git repositories. We attempt to understand its patterns and explore its impact on the related measures. We sample 450 Git repositories from GitHub with control of their age and popularity (in terms of number of forks). We extract the names and email addresses of the developers (committers and authors) from the repositories by going through the commit histories and use a high precision method to merge the names and email addresses that belong to the same developer to achieve a **conservative lower bound** of the prevalence and impact of the MNE issue. With the merged names and email addresses of a developer, we find that about 6% of the developers used MNE in more than 60% of the repositories. More importantly, they contributed about half of all the commits. The MNE derive from letter case variance, name and email address segments recombination and letters replacement. The similarity based on Levenshtein distance for many MNE pairs is not high. We study the age and popularity of the repositories and developers' stay time and number of commits within the repository. They are significantly related to MNE, but the relationships are not strong. Developers having MNE interleaved their names and

2574-3864/19/$31.00 ©2019 IEEE
DOI 10.1109/MSR.2019.00068

email address, even within the same week. Our impact analysis shows that the MNE issue cannot be ignored for the basic related measurements, in particular for the measures of large projects and core developers. The existing similarity based approaches may not work well for many MNE, and other strategies and data sources should be included to ensure the accuracy.

The contributions of our paper include:

- It reveals a conservative lower bound of the prevalence and impact of the MNE issue in version control repositories.
- It provides a number of quantitative evidences for researchers and practitioners to review the existing approaches of merging MNE.
- It offers several implications for version control data users to make decisions in handling the MNE problem.

The data and descriptions for the replication of this study are available at: https://github.com/jxshin/MNE.

The remainder of this paper is structured as follows. In Section II, we introduce the background and review the related work. In Section III, we form our research requestions. We elaborate how we collect the data in Section IV-A and how we handle MNE in Section IV-B. We answer the research questions of MNE patterns in Section V, and research questions of impact in Section VI. We discuss the implications for version control data users in Section VII. We outline the limitations and threats in Section VIII and summarize in Section IX.

II. Background and Related Work

A. Version Control System and Version Control Data

Version control systems are used to record developers' modifications to the code, and the recorded information includes the modifications, developers who made the modification and the time of the modification [28]. A commit is the basic record unit in most modern version control systems such as Git [4]. A developer in the record is often labelled by her name and email address. The name and email address are set and can be changed by developers themselves freely, and version control systems often do not have any restrictions and checks. That is why there could be multiple names/email addresses belong to a developer and duplicated names used by different developers.

Commit logs, i.e., version control data, have being used to solve many software engineering problems. The data can be used to study how developers participate in a software project [8], [20], [29], [38], [48], to recommend developers for development tasks [39], [46], to predict bugs or come up with bug fixes [15], [21], [26], [33], [44], [45], etc. There are also substantial studies using such data to investigate the social networks of developers [11], [30], [42]. As introduced in Section I, researchers who face the MNE issue may not have a quantitative understanding of it.

B. Identity Recognition Studies

There have been several approaches identifying distinct developers. Bird et al. used Levenshtein distance to merge similar names and email addresses in version control data [3]. Canfora et al. proposed a similar approach in their study of bug fixes in FreeBSD/OpenBSD [6]. Robles et al. attempted to identify distinct developers from multiple data sources [35]. Goeminne and Mens extended these approaches and achieved a better performance [16]. Kouters et al. used Latent Semantic Analysis for the identification [23]. Oliva et al. merged email addresses associated with the same name [32]. The approach of Both Kouters et al. [23], Goeminne and Mens [16] also merges email addresses with the same prefix. Wiese et al. conducted a performance comparison of the existing approaches [41]. The results show that most of the approaches have similar recall, while Oliva et al.'s approach has the highest precision.

Most of the related studies focus on the approaches of identifying distinct developers. There are few studies quantifying the MNE problem in version control data, in particular, the prevalence and impact. In this paper, we attempt to obtain a quantitative understanding of MNE in version control data, e.g., whether the MNE problem is significant and deserves very careful treatment, and help data users more accurately analyse the version control data.

III. Research Questions

We raise the following research questions for MNE patterns (RQ 1, 2, 3 and 4) and MNE impact on related measures (RQ5).

RQ1: How prevalent is the MNE issue in the version control data? The extent of prevalence is the basic problem which determines the importance of understanding and dealing with MNE.

RQ2: How are MNE of a developer different from each other in form? In Section II, we see that many existing approaches of merging MNE are based on the form similarity of names and email addresses. It is helpful to know the form difference of MNE to evaluate and improve such methods.

RQ3: Do developers use MNE all the time? For developers who used MNE, it's interesting to know whether they regularly used only one name and one email address within a time period. There might be a chance of having a lower risk in the measurements when the studied commit history is short.

RQ4: What are the factors related to MNE? The related factors can help data users estimate the risk in different situations. We speculate that the age and popularity of a repository, developer's time of stay within a repository, number of commits committed and authored by a developer within a repository are related.

RQ5: What is the impact of MNE on the basic measures of the number of developers in a repository, developer's time of stay within a repository, number of commits committed and authored by a developer within a repository? These measures are very common in the investigations of software development [29], [34], [50]. The measurement results could be biased due to the MNE issue, and wrong conclusions might be drawn.

IV. METHODOLOGY

A. Data Collection

To make our results representative, we consider several aspects to collect the data. First, we select GitHub as the data source, which is the most popular software development platform [12]. It is built on a powerful and popular version control system, Git, and has over 106 million repositories and over 36 million users[1]. Second, we sample repositories with the control of their popularity and age with consideration of our research questions. The data collection is conducted in Oct. 2018.

1) Data Sampling: Because it's very costly to obtain the full repository list and calculate the overall distribution to conduct the sampling, we select three age intervals and popularity intervals to search repositories with different ages and popularity. We attempt to make these sub-ranges stretch cross the full range. The intervals are shown in Table I. The age interval is based on the even division of the GitHub history.For the popularity interval, we view the repositories having less than 10 forks as less popular ones, and those with over 1,000 as top popular ones. The moderate popular ones are at the middle of the log scale from 10 to 1000. Older repositories tend to have more forks, therefore, each popularity group of our analysis in Section V-D includes repositories of different ages.

The combination of the two types of intervals forms nine groups of repositories, as shown in Table I.

TABLE I
REPOSITORY AGE AND POPULARITY INTERVALS

Popularity	Age	# of sampled repos	Group label
<= 10 forks	< 1 year	50	A-1
<= 10 forks	5 year	50	A-2
<= 10 forks	10 year	50	A-3
100-200 forks	< 1 year	50	B-1
100-200 forks	5 year	50	B-2
100-200 forks	10 year	50	B-3
> 1000 forks	< 1 year	50	C-1
> 1000 forks	5 year	50	C-2
> 1000 forks	10 year	50	C-3

We search repositories for each group through GitHub search API[2]. The search query includes creation date and number of forks of the repositories. In addition, we only consider repositories which are not forks[3] and are active, i.e., whose latest push was done after 2018-06-01, and include the corresponding criteria in the query. Because of the access restriction of GitHub, each search query can obtain up to 100 items. Because the repositories are already randomly indexed by GitHub, we only make one query for each group to reduce the effort. Among the searched repositories, we further randomly sample 50 repositories within each group as the final repositories for our study. We download the sampled repositories by *git clone* and obtain the corresponding repository information through GitHub repository API[4]. As shown in Table II, the sampled repositories include diversities [31] on programming languages (which also implies application domains) and sizes.

TABLE II
SUMMARY OF SAMPLED REPOSITORIES

Top popular programming languages			
Language	# of repos (ratio)	Language	# of repos (ratio)
JavaScript	104 (23.11%)	Python	46 (10.22%)
Ruby	44 (9.78%)	Go	19 (4.22%)
Swift	17 (3.78%)	HTML	16 (3.56%)
Java	16 (3.56%)	C	16 (3.56%)
C++	14 (3.11%)	other 35 langs	158 (35.11%)

Distribution of repository size in KB (quantiles)				
0%	25%	50%	75%	100%
3.0	313.5	2,231.0	12,562.5	1,432,774.0

2) Data Extraction: We extract the names, email addresses of committers and authors and the operation time from the downloaded repositories by traversing the commit logs. The name and email address of a developer (i.e. author or committer) in a commit form a name-email pair, i.e., a tuple $< name, email >$. We totally obtain 2,302,806 name-email pairs. The pairs are merged in Section IV-B to identify distinct developers.

B. Merging of MNE

The existing approaches of merging MNE in version control data cannot achieve high precision and recall at the same time [41]. To address this problem, we attempt to start from employing an approach with a precision of $\sim 100\%$, which could ensure a **conservative lower bound** for the estimation of prevalence and impact.

1) Assumptions: We introduce two assumptions: 1) An email address only belongs to one developer; 2) The possibility is very high that a name only belongs to one developer within one repository. These two assumptions are also used by Oliva et al. [32], but they did not investigate the confidence of the assumptions.

We analyse the confidence of our assumptions as following. The first assumption is based on the convention that developers always use personal email address in the version control system[5] in order to be accurately contacted for their commits. There is a possibility that someone assumes others' email addresses. However, it can be ignored in the OSS projects, because it's a common sense that participants value their reputation in the community [20]. For the second assumption, we estimate the popularity of the developers' names with the approved developers index of four OSS communities/projects: Apache Software Foundation (ASF)[6], Linux kernel[7], Ubuntu[8],

[1]The number was obtained at Jan. 22th, 2019 via GitHub search.
[2]https://developer.github.com/v3/search/
[3]https://help.github.com/articles/about-forks/
[4]https://developer.github.com/v3/repos/
[5]https://www.kernel.org/doc/html/latest/process/maintainer-pgp-guide.html
[6]https://people.apache.org/committer-index.html
[7]https://github.com/torvalds/linux/blob/master/CREDITS
[8]https://launchpad.net/∼ubuntumembers/+members

and Go[9], which are widely participated. All the developers are authenticated with their real names in these indexes. Each entry in an index indicates a distinct developer. Table III shows the popularity of developers' names within each community/project. Most of the developers' names are exclusive within each community/project, and the possibility that a name belongs to one developer is not less than 99.64%, even when there are thousands of developers. It indicates a very high confidence for the assumption that a name is exclusive within a single repository.

TABLE III
NAMES POPULARITY

Site	# of Name	# of developers	# of exclusive names (ratio)
ASF	3,086	3,097	3,075 (99.64%)
Linux kernel	550	551	549 (99.82%)
Ubuntu	423	423	423 (100%)
Go	1,819	1,819	1,819 (100%)

2) Merging Process: We merge names and email addresses that belong to the same developers through joint comparisons. According to our first assumption, we traverse all the name-email pairs and merge pairs with identical email address (case insensitive). The merged name-email pairs become name-email group pairs. A name-email group pair is a tuple consisting of two sets, the name set and email address set $< \{name1, name2...\}, \{email1, email2, ...\} >$. After the step above, there are some name-email group pairs having multiple names. According to our second assumption, we traverse name-email group pairs within repositories and merge the pairs that have intersection of identical names (case insensitive). Be cause the names in Git may be informal or not indicate a person, we also include a restriction to ensure the precision: **only names including at least two segments (split by spacing), e.g., first name and last name, are considered in the intersection**. After this step, there are some name-email group pairs having multiple email addresses. Finally, we iteratively merge the name-email group pairs that have intersection of identical email addresses (case insensitive) until there are no intersections between any two of the pairs. The volume of the data is not very large, and the merging process can be done in several hours on average PCs.

3) Analysis of the Precision: We follow Wiese et al.'s method [41] to analyse the precision for each developer and the average precision for all. The precision for a developer is the number of identified names and email addresses that really belong to her divided by the number of all the identified names and email addresses of her. The average precision is estimated by Equation 1. Assume that there are N developers in a repository, n developers sharing the same name. All of the precision for the $N - n$ developers using exclusive names is 100% according the first assumption illustrated in Section IV-B1. Assume that the average precision for the n developers is p. In the pilot developer indexes introduced in Section IV-B1, n/N is 0.7%, 0.4%, 0% and 0% for ASF, Linux kernel, Ubuntu and Go respectively. Therefore, we estimate that n/N here is 0.7% approximately. We can see that the finally estimated average precision is very close to 100%, even if p was 0.

$$\begin{aligned} precision_{avg} &= \frac{1}{N} \times \sum_{i=1}^{N} precision_i \\ &= \frac{1}{N} \times ((N-n) \times 100\% + n \times p) \\ &= 100\% - n(1-p)/N \\ &\sim 100\% - 0.7\% \times (1-p) \\ &\sim 99.3\% + 0.7\% \times p \\ &\sim 99.3\% \end{aligned} \tag{1}$$

4) Analysis of the Recall: Wiese et al.'s study shows that Oliva et al.'s approach [32] has the highest precision [41]. Our approach is very similar with it. The difference is that the merging process in our approach is conducted more strictly, i.e., we have a higher precision but a lower recall. The average recall of Oliva et al.'s approach is around 50%, i.e., that of ours is lower than 50%. Therefore, the true prevalence and impact of the MNE issue should be higher than what we report in this paper.

C. Quantitative Analyses

We quantitatively study our research questions. We report basic statistics, including mean, median, etc. in the measurements.

We conduct (Pearson's) chi-squared test [43] to inspect the significance of association between two categorical variables. We report effect size with the phi coefficient [9]. We employ Pearson correlation coefficient [2] to measure the association between two continuous variables. We fit the logistic regression models [10] to investigate the association between a boolean variable and a continuous variable, and report effect size of the independent variable with odds ratio.

We conduct Wilcoxon signed rank test [36] to check the significance of difference between two paired groups. An observation in one group is based on the merged identity, and in the other group, it is based on the name/email address without merging. Because we are interested in the difference of the value, we report the effect size with the median and mean difference. We use R[10] to conduct these analyses.

V. PATTERNS OF MNE

We obtain 45,853 merged identity (merged name-email group pairs) in total through the method described in Section IV-B, i.e, 45,853 distinct developers are identified.

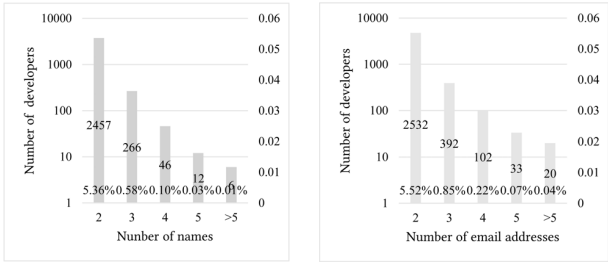
A. Prevalence

We count and calculate the number and ratio of developers using MNE, the number and ratio of commits associated with MNE. The 45,853 identified distinct developers used 46,361 (case sensitive) distinct names and 49,738 (case sensitive)

[9]https://github.com/golang/go/blob/master/CONTRIBUTORS

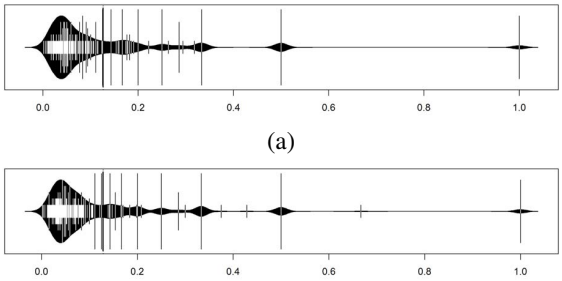
[10]https://www.r-project.org/

distinct email addresses. There are 2,787 (6.08%) developers using multiple names who committed or authored 508,247 (44.09%) commits and 3,079 (6.71%) developers using multiple email addresses who committed or authored 687,741 (59.67%) commits. The prevalence of multiple names and multiple email addresses are similar, while it is a little bit higher for multiple email addresses. Figure 1 (a) and (b) show the distribution of number of names and email addresses used by the developers respectively. We can see that most of the developers who used MNE have two names or two email addresses, but there are some developer using more than five names or email addresses. More importantly, developers who used MNE contributed about half of the commits.



(a) (b)

Fig. 1. Distribution of the number of names and email addresses used by the developers.



(a) (b)

Fig. 2. Beanplots of the ratio of developers using multiple names (a) and multiple email addresses (b) within one repository.

We also count the number of repositories that have developers using MNE and calculate the ratio of these developers within each repository. There are 293 (65.11%) repositories having developers using multiple names and 298 (66.22%) repositories having developers using multiple email addresses. Figure 2 (a) and (b) show the distribution of ratio of developers using MNE within a repository. The ratio for both multiple names and multiple email addresses ranges from 0.18% to 100%. There are five repositories where all the developers used multiple names and five repositories where all the developers used multiple email addresses. For example, in the repository *asavinov/bistro*, there are only one developer, and he used two email addresses. For the majority of the repositories, the ratio is similar with the ratio across the repositories. The median ratio of multiple names is 6.36%, and it is 6.80% for multiple email addresses. It implies that the MNE issue should be considered both within and across the repositories.

Answer to RQ1: About 6% of the developers used MNE in more than 60% of the repositories, and they contributed about half of the commits.

B. Durative

We set five time windows, one week, one month, one quarter, half a year, one year to observe how developers MNE within different time periods. We split the commit history of each repository (in commit time sequence) by the time windows, and count the number of names and email addresses used by each developer in the period. We refer **M** developers to those who used multiple names/email addresses within a repository. We report the number and ratio of two types of developers in Table IV: **type MBS** (multiple but single), the M developers who only used one name/email address within each period of the repository's commit history, **type MAM** (multiple and multiple), the M developers who used multiple names/email addresses in more than one third periods of the repository's commit history. The results show that there are always MBS and MAM developers in a certain period of a repository's commit history. About 18% of the M developers kept using only one name and 25% of the M developers kept using only one email address over one year. About 24% of the M developers kept using multiple names and 22% of the M developers kept using multiple email addresses within one week. The number and ratio of MBS developer decrease with the increase of time window, and it is on the contrary for the number of ratio of MAM developers.

Answer to RQ2: The majority of the developers who used MNE interleave their names and email address, even within the same week.

TABLE IV
DURATIVE OF MULTIPLE NAMES AND EMAIL ADDRESSES USAGE

Name			
time window	# of MBS (ratio)	# of MAM (ratio)	# of M
one week	1,204 (48.30%)	609 (24.43%)	
one month	975 (39.11%)	935 (37.51%)	
one quarter	758 (30.41%)	1,289 (51.70%)	2,493
half a year	617 (24.75%)	1,562 (62.66%)	
one year	450 (18.05%)	1,858 (74.53%)	
Email address			
time window	# of MBS (ratio)	# of MAM (ratio)	# of M
one week	1,638 (50.93%)	696 (21.64%)	
one month	1,406 (43.72%)	1,027 (31.93%)	
one quarter	1,198 (37.25%)	1,402 (43.59%)	3,216
half a year	1,024 (31.84%)	1,733 (53.89%)	
one year	803 (24.97%)	2,158 (67.10%)	

C. Forms

We try to find out the form difference among the MNE used by a developer to obtain implications for approaches of MNE merging. We collect 3,817 name pairs and 5,200 email address pairs from developers' MNE. A name pair or email address pair is a name or email address tuple where the names or email addresses belong to one developer. By manually inspecting

the pairs, we summarise three orthometric types of name and email address variation.

1) Type 1: Only the case of the letters changes. We find that 173 (4.53%) name pairs and 84 (1.62%) email address pairs belong to this type. It shows that there are developers typing their names or email addresses with different cases, but it's not very common.

2) Type 2: Only some segments of the names or email addresses change[11]. Name always have several segments, e.g., first name, middle name and last name. Developers may use one or some or all of the segments with different orders. We find that 1,251 (32.77%) name pairs belong to this type. It indicates that developers often used different segments of their names.

An email address has two segments: the prefix indicating the individual person, and the postfix indicating the email server. Within the same email server, the prefix of a developer must be different from that of others. We find that 1,368 (26.31%) email address pairs have the same prefix with different postfixes. It means that developers often registered several email services with the same prefix and used them in the version control systems. We find that 297 (5.71%) email address pairs have the same postfix with different prefixes. It indicates that developers sometimes registered an email services with several prefixes and used them in the version control systems, but it's not common.

3) Type 3: Some letters are replaced[12]. MNE pairs which are neither Type 1 nor Type 2 belong to this Type. We find 2,402 (62.93%) name pairs and 3,415 (65.67%) email address pairs of Type 3. It indicates that the majority of the MNE pairs do not have clear difference patterns in form. We calculate the Levenshtein distance, which is a commonly used similarity measure of strings, to investigate the similarity of MNE of Type 3. The similarity is calculated as:

$$Similarity = 1 - \frac{LevenshteinDistance(A, B)}{max(length(A), length(B))} \quad (2)$$

Figure 3 (a) and (b) show the distribution of the similarity. We can see that most of the pairs' similarity is not very high. The median similarity of the name pairs of Type 3 is 0.3077, and it is 0.3333 for the email address pairs.

Answer to RQ3: Many MNE do not have clear difference patterns in form and are not very similar with each other in form.

D. Related Factors

We investigate related factors of MNE from the perspective of repositories and developers.

1) Perspective of Repositories: We study whether the age and popularity of the repositories are associated with the occurrence of MNE.

Age of Repositories: We have sampled repositories of different ages in Section IV-A, which form three groups, 1, 2 and 3 (see Table I). Table V shows the number and ratio of repositories having MNE within each group. We can see that with the increase of age, the ratio increases for both multiple names and multiple email address. We conduct chi-square test to check whether the association is significant. Each observation in the test is a repository. The two categorical variables of the test are: 1) the age group of the repository, and 2) whether the repository has developers used MNE. The p-value and effect size of the test for multiple names are 2.143×10^{-5} and 0.219. The p-value and effect size of the test for multiple email addresses are 6.937×10^{-9} and 0.289. The association is statistically significant but not strong.

[11]We ignore the case of the letters in Type 2.

[12]We ignore the case of the letters in Type 3.

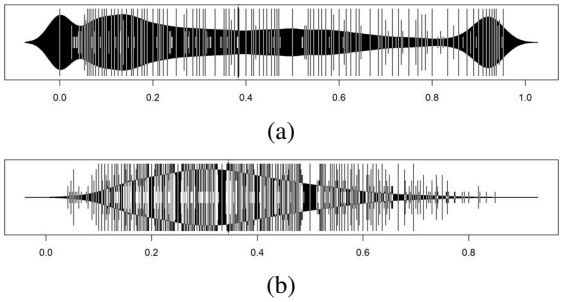


Fig. 3. Beanplots of the similarity of names (a) and email addresses (b) of Type 3 used by the same developer.

TABLE V
NUMBER AND RATIO OF REPOSITORIES HAVING MNE AMONG AGE GROUPS

Grp.	Age	Multiple names		Multiple email addresses	
		# of repos (all)	ratio	# of repo (all)	ratio
1	$<$ 1 year	80 (150)	53.33%	73 (150)	48.67%
2	5 year	95 (150)	63.33%	102 (150)	68.00%
3	10 year	118 (150)	78.67%	123 (150)	82.00%

Table VI shows the number and ratio of developers using MNE within each group. We can see that with the increase of age, the number and ratio increase for multiple email address. We also conduct chi-square test to see whether the association is significant. Each observation in the test is a developer. The two categorical variables of the test are: 1) the age group of the repository where the developer is, and 2) whether the developer used MNE. The p-value and effect size of the test for multiple names are 6.515×10^{-8} and 0.025. The p-value and effect size of the test for multiple email addresses are $< 2.2 \times 10^{-16}$ and 0.066. The association is statistically significant but not strong either.

TABLE VI
NUMBER AND RATIO OF DEVELOPERS USING MNE AMONG AGE GROUPS

Grp.	Age	Multiple names		Multiple email addresses	
		# of devs (all)	ratio	# of devs (all)	ratio
1	$<$ 1 year	443 (10,496)	4.22%	337 (10,496)	3.21%
2	5 year	707 (12,370)	5.72%	733 (12,370)	5.93%
3	10 year	1,343 (29,214)	4.60%	2,146 (29,214)	7.35%

We calculate the Pearson correlation coefficient of the repositories' age and ratio of developers using multiple names and multiple email addresses. Each observation in the calculation is a repository. The results are 0.050 and 0.143 for multiple names and multiple email addresses respectively, indicating a weak relationship.

Popularity of Repositories: We have sampled repositories of different popularity in Section IV-A, which forms three groups, A, B and C (see Table I). Table VII shows the number and ratio of repositories having MNE within each group. We can see that with the increase of popularity, the ratio of repositories having MNE increases for both multiple names and multiple email addresses. We conduct chi-square test to see whether the association is significant. Each observation in the test is a repository. The two categorical variables of the test are: 1) the popularity group of the repository, and 2) whether the repository has developers used MNE. The p-value and effect size of the test for multiple names are 5.028×10^{-16} and 0.396. The p-value and effect size of the test for multiple email addresses are 1.297×10^{-8} and 0.284. The association is significant for both multiple names and multiple email addresses, however, the effect is moderate for multiple names and small for multiple email addresses.

TABLE VII
NUMBER AND RATIO OF REPOSITORIES HAVING MNE AMONG POPULARITY GROUPS

Grp.	Forks	Multiple names		Multiple email addresses	
		# of repos (all)	ratio	# of repo (all)	ratio
A	$<= 10$	65 (150)	43.33%	72 (150)	48.00%
B	$100-200$	94 (150)	62.67%	106 (150)	70.67%
C	> 1000	134 (150)	89.33%	120 (150)	80.00%

Table VIII shows the number and ratio of developers using MNE within each group. We can see that with the increase of popularity, the ratio of developers using MNE within repository decreases first and increases then for both multiple names and multiple email addresses. We also conduct chi-square test to check whether the association is significant. Each observation in the test is a developer. The two categorical variables of the test are: 1) the popularity group of the repository where the developer is, and 2) whether the developer used MNE. The p-value and effect size of the test for multiple names are 6.369×10^{-10} and 0.029. The p-value and effect size of the test for multiple email address are 2.048×10^{-13} and 0.033. The association is statistically significant but not strong.

TABLE VIII
NUMBER AND RATIO OF DEVELOPERS USING MNE AMONG POPULARITY GROUPS

Grp.	Forks	Multiple names		Multiple email addresses	
		# of devs (all)	ratio	# of devs (all)	ratio
A	$<= 10$	78 (827)	9.43%	98 (827)	11.85%
B	$100-200$	305 (7,039)	4.33%	364 (7,039)	5.17%
C	> 1000	2,110 (44,214)	4.77%	2,754 (44,214)	6.23%

We calculate the Pearson correlation coefficient of the repositories' number of forks and ratio of developers using multiple names and multiple email addresses. Each observation in the calculation is a repository. The results are -0.097 and -0.121. The relationship is weak too.

2) Perspective of Developers: We study whether the time a developer stayed in the repository and the number of her commits in the repositories are associated with whether she used MNE.

Time of Stay in the Repository: We calculate the time a developer stayed in a repository by finding the time of the first commit she committed or authored and the time of the last commit she committed or authored. The time span is the time of stay.

We fit the logistic regression models (shown in Equation 3 and 4, where the time unit is one month) to inspect whether the chance a developer used multiple names or multiple email addresses is associated with the time she stayed in the repository. We report the effect size with odds ratio.

$$use\_multi\_names(true\_or\_false) \sim time\_of\_stay \quad (3)$$

$$use\_multi\_email\_addresses(true\_or\_false) \sim time\_of\_stay \quad (4)$$

Table IX shows the fitting results, where the p-value is much smaller than 0.01, and the odds ratio is 1.04297 and 1.064076 for Model 3 and 4 respectively, which indicates that the longer a developer stayed in a repository, the higher chance that she used MNE. The odds would multiply by 1.04297 and 1.064076 if she stayed one month longer.

TABLE IX
FITTING RESULTS OF MODEL 3 AND 4

Model 3				
	Estimate	Std. Error	z value	$Pr(<\mid z \mid)$
(Intercept)	-3.368e+00	2.461e-02	-136.84	<2e-16
time of stay	0.0420724	0.0009111	46.18	<2e-16
Model 4				
	Estimate	Std. Error	z value	$Pr(<\mid z \mid)$
(Intercept)	-3.312e+00	2.398e-02	-138.08	<2e-16
time of stay	0.062107	0.001035	60.03	<2e-16

We calculate the Pearson correlation coefficient between developers' stay time and the number of names used, the number of email addresses used. Table X shows the results, where the association is weak and moderate, and the longer a developer stayed in a repository, the more names and email addresses she would use.

TABLE X
PEARSON CORRELATION COEFFICIENT BETWEEN DEVELOPERS' TIME OF STAY AND NUMBER NAMES USED, NUMBER OF EMAIL ADDRESSES USED

	# of names	# of email addresses
time of stay	0.28	0.46

Number of Commits We fit the logistic regression models (shown in Equation 5, 6, 7 and 8) to inspect whether the

chance a developer used multiple names and email addresses is associated with the number of commits she committed and authored in the repository.

$$use\_multi\_name(true\_or\_false) \sim \#of\_comnits\_committed \quad (5)$$

$$use\_multi\_email\_addresses(true\_or\_false) \sim \#of\_comnits\_committed \quad (6)$$

$$use\_multi\_name(true\_or\_false) \sim \#of\_comnits\_authored \quad (7)$$

$$use\_multi\_email\_addresses(true\_or\_false) \sim \#of\_comnits\_authored \quad (8)$$

Table XI shows the fitting results, where the p-value is much smaller than 0.01, and the odds ratio is 1.000855, 1.001706, 1.001376 and 1.002545 Model 5, 6, 7 and 8 respectively. The results indicate that the more commits a developer committed and authored in a repository, the higher the chance that she used MNE is. The odds would multiply by 1.000855 and 1.001706 if she committed one more commit. And the odds would multiply by 1.001376 and 1.002545 if she authored one more commit.

TABLE XI
FITTING RESULTS OF MODEL 5, 6, 7 AND 8

Model 5				
	Estimate	Std. Error	z value	$Pr(<\mid z \mid)$
(Intercept)	-3.026e+00	2.087e-02	-144.96	<2e-16
# of co committed	8.548e-04	6.123e-05	13.96	<2e-16
Model 6				
	Estimate	Std. Error	z value	$Pr(<\mid z \mid)$
(Intercept)	-2.788e+00	1.879e-02	-148.41	<2e-16
# of co committed	1.705e-03	8.435e-05	20.21	<2e-16
Model 7				
	Estimate	Std. Error	z value	$Pr(<\mid z \mid)$
(Intercept)	-3.049e+00	2.110e-02	-144.51	<2e-16
# of co authored	1.375e-03	7.758e-05	17.73	<2e-16
Model 8				
	Estimate	Std. Error	z value	$Pr(<\mid z \mid)$
(Intercept)	-2.8191387	0.0190508	-148.0	<2e-16
# of co authored	0.0025413	0.0001063	23.9	<2e-16

We calculate the Pearson correlation coefficient between developers' number of commits committed/authored and number of names/email addresses used. Table XII shows the results, where the associations are weak.

Answer to RQ4: All the studied factors are moderately or weakly associated with MNE, and the majority of the associations are statistically significant.

TABLE XII
PEARSON CORRELATION COEFFICIENT BETWEEN DEVELOPERS' NUMBER OF COMMITS COMMITTED/AUTHORED AND NUMBER NAMES/EMAIL ADDRESSES USED

	# of names	# of email addresses
# of commits committed	0.14	0.24
# of commits authored	0.18	0.27

VI. IMPACT OF MNE

We investigate the impact of the MNE issue on related measurements. We study three basic and commonly used metrics, the number of developers within a repository, the time a developer stayed in a repository and the number of commits a developer committed and authored in a repository. To estimate the impact, we compare the measurement results obtained through merged identity and name/email address based identity without merging.

A. Number of Developers

We count the number of developers within a repository through merged identity and name/email address based identity without merging. We separate the counting into three groups, (a), (b) and (c), and Figure 4 (a), (b) and (c) show the distribution of number of developers within a repository for the corresponding groups. Group (a) is based on the name without merging, (b) is based on the email address without merging and (c) is based on the merged identity. The median and mean are 20.50, 120.71 in (a), 20.00, 124.65 in (b) and 18.00, 114.50 in (c). We conduct Wilcoxon signed rank test to inspect whether there are significant differences. We report the effect size with the median and mean increase of the number of the developers caused by the MNE issue. The p-values of the test between (a) and (c), (b) and (c) are $< 2.2 \times 10^{-16}$ and $< 2.2 \times 10^{-16}$. The median increase and mean increase are 8.33% and 20.04% from (c) to (a). The median increase and mean increase are 10.84% and 26.98% from (c) to (b).

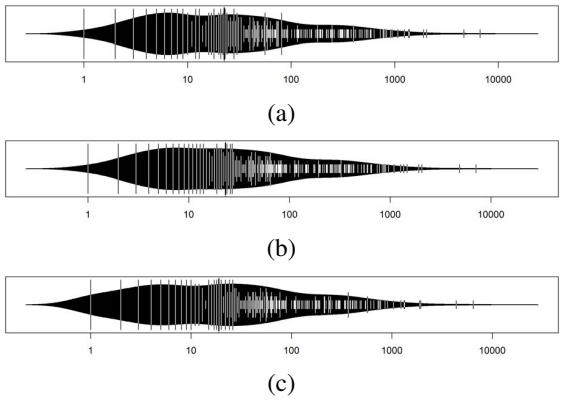


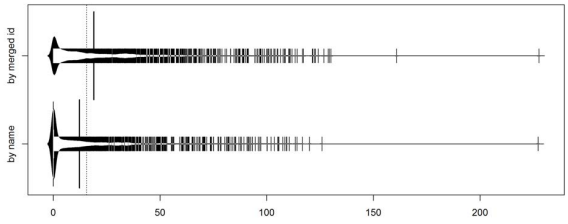
Fig. 4. Beanplots of the number of developers.

B. Time of Stay

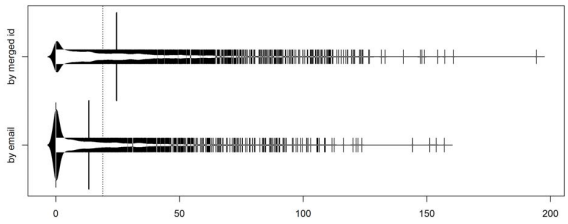
The proportion of developers using MNE is less than 10%, and the overall impact would not be obvious. Therefore, in this part we only focus on developers who used MNE. We calculate the time of stay of these developers within a repository through merged identity and name/email address based identity without merging. The calculation method is the same with that in Section V-D2. Note that, the comparison is paired. For developers identified by name/email address without merging, we choose the name/email address associated with the longest time. We calculate the median and mean difference of the stay time of the developers, and calculate

the median and mean decrease of the stay time caused by the MNE issue. Because only developers using MNE are included, it is unnecessary to test the significance.

Figure 5 shows the stay time of the developers within a repository. In part (a), the developers are identified by the name without merging vs. merged identity. The median difference and mean difference are 1.71 months and 6.79 months. The median decrease and mean decrease are 48.28% and 52.28%. In part (b), the developers are identified by the email address without merging vs. merged id. The median difference and mean difference are 4.37 months and 11.18 months. The median decrease and mean decrease are 54.51% and 55.18%.



(a)



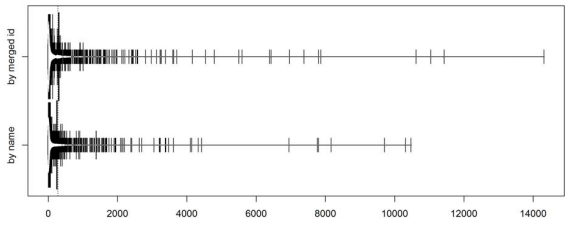
(b)

Fig. 5. Beanplots of the stay time of developer using multiple names or emails.

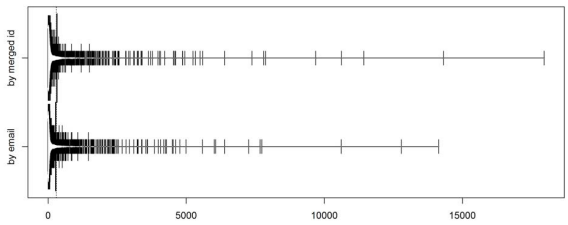
C. Number of Commits

Similar with Section VI-B, in this part, we only focus on developers using MNE too. We count the number of commits of developers within a repository through merged identity and name/email address based identity without merging. Note that, the comparison is paired. For developers identified by name/email address without merging, we choose the name/email address associated with the most commits. We calculate the median and mean difference of developers' number of commits and calculate the median and mean decrease of the number of commits caused by the MNE issue. Because only developers using MNE are included, it is unnecessary to test the significance either.

Figure 6 shows the distribution of the number of commits developers **committed** within a repository. In part (a), the developers are identified by the name without merging vs. merged identity. The median difference and mean difference are 3 commits and 56.27 commits. The median decrease and mean decrease are 25% and 24.94%. In part (b), the developers are identified by the email address without merging vs. merged identity. The median difference and mean difference are 3 commits and 48.19 commits. The median decrease and mean decrease are 25% and 25.58%.



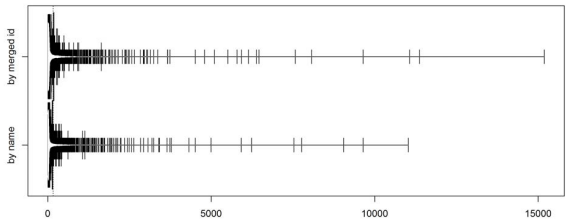
(a)



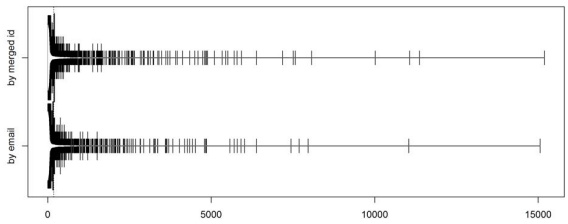
(b)

Fig. 6. Beanplots of the number of commits committed by developer using multiple names or emails.

Figure 7 shows the distribution of the number of commits developers **authored** within a repository. In part (a), the developers are identified by the name without merging vs. merged identity. The median difference and mean difference are 2 commits and 33.62 commits. The median decrease and mean decrease are 29.73% and 27.96%. In part (b), the developers are identified by the email address without merging vs. merged identity. The median difference and mean difference are 2 commits and 31.58 commits. The median decrease and mean decrease are 30.04% and 28.65%.



(a)



(b)

Fig. 7. Beanplots of the number of commits authored by developer using multiple names or emails.

Answer to RQ5: The impact of MNE on the measures of the number of developers in a repository, developers' stay time and number of commits should not be ignored.

VII. Implications for Version Control Data Users

We discuss the implications of our results for version control data users to deal with the MNE issue.

A. Patterns

In Section V-A, we only identify a small proportion of developers using MNE, however, they occur in more than half of the repositories and also contributed about half of the commits. We suggest version control data users pay more attention to the MNE issue. We also find that the usage of multiple email addresses is more common than multiple names. When a study is conducted within a single repository, using the names to identify developers could be a better choice than using the email addresses.

Our results in Section V-B imply that even the studied time period is not very long, e.g., one week, the risk of having bias introduced by the MNE issue still exists and should be addressed.

In Section V-C, we identify three types of MNE forms. For Type 1, data users should unify the case of the names and email addresses in the data pre-process. For Type 2, data users may use the heuristics of comparing the segments of the names and email addresses. For Type 3, the Levenshtein distance based similarity is often low, therefore, it is necessary to include other kinds of heuristics or data sources, e.g., issue tracking data, to achieve a high accuracy.

In Section V-D, all the studied factors are significantly associated with MNE, but the majority of the associations are not strong. The moderate or weak relationship implies that the measurement risk is difficult to control from the perspective of these factors, and merging of MNE is the only effective way to reduce the risk at present.

B. Impact

Our results in Section VI tell that the MNE issue can make a big difference in the studied basic measures. Therefore, it's necessary to merge MNE to reduce bias in the measures, in particular for the large projects and core developers. We also suggest reviews of the previous work potentially influenced by the MNE issue. New findings may emerge.

In summary, we highlight the importance of the careful handling of the MNE issue from a number of aspects. Version control data users should leverage the existing MNE merging approaches in their related studies and evaluate or discuss the threats of MNE. We also provide several quantitative evidences for the selection and improvement of the merging approaches and the estimation of the threats to the results of the studies.

VIII. Limitations

The primary limitations and threats lie in the way we merge MNE, privacy of developers, implications, and generalization of the results.

Merging of MNE. There is no ground truth for the GitHub repositories. We have to merge MNE through unsupervised method. In this study, we do NOT aim at merging all the names and email addresses belonging to a developer, but merging them with a high precision to get a lower bound of the prevalence and impact. Therefore, we do not employ the existing approaches which attempt to achieve a balance of the precision and recall. We do not intend to propose a method for other researchers to merge MNE either. Our merging is based on the assumptions proposed in Section IV-B1. The assumptions are not true for all the cases, e.g., names or email addresses of a team or a proxy may occur. The pilot developer indexes may not fully represent the developers in the GitHub repositories.In consequence, the precision of the merging may not be 100%, and there could be false positives. We have analysed the confidence of the assumptions in Section IV-B1. We also manually go through the merged data, and few questionable identities are found. We remind readers of understanding the results in this paper with the estimate precision.

Privacy issue. Because of the privacy issue for the studied developers, we do not discuss any concrete examples of the multiple names or email addresses used by them in the paper.

Impact of the MNE issue. In this study, we have not investigated the specific impact of the MNE issue on the previous studies. The manual examination of the papers conducted in Section I is a very simple investigation to motivate our study. The result does not indicate that these studies suffer from the issue.

Generalization. Because of the restrictions, we sample 450 repositories of several intuitive sub-ranges of the repository size, age and popularity, which is not based on the overall distribution. The studied repositories and developers in them may not be able to represent all, and the results may not apply to other data, e.g., the samples which only contain very large repositories. We have attempted to make the sub-ranges stretch cross the full range. The repositories cover a relatively large scope of languages and application domains.

This study is a conservative estimation of the MNE issue. The insights are preliminary. We hope it could shed more lights on the MNE issue and make us step forward on the way of a deeper understanding.

IX. Conclusions

In this study, we investigate the MNE problem in version control repositories. We reveal several patterns of MNE and quantify the impact of MNE on several basic related measures. Our findings enrich people's knowledge of the MNE issue. We hope the results of this study could inspire and promote further investigations of MNE.

In future, we intend to incorporate more data sources to increase the merging accuracy, in particular, the recall to include more MNE cases. We also plan to evaluate the impact on the results of the previous studies using version control data and offer more actionable insights.

Acknowledgments

This work is supported by the National Key R&D Program of China Grant 2018YFB1004201 and the National Natural Science Foundation of China Grant 61802378.

REFERENCES

[1] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *Proceedings of the 28th International Conference on Software Engineering*, pages 361–370. ACM, 2006.

[2] J. Benesty, J. Chen, Y. Huang, and I. Cohen. *Pearson Correlation Coefficient*, pages 1–4. Springer Berlin Heidelberg, 2009.

[3] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *Proceedings of the 2006 International Workshop on Mining Software Repositories*, pages 137–143. ACM, 2006.

[4] C. Bird, P. C. Rigby, E. T. Barr, D. J. Hamilton, D. M. German, and P. Devanbu. The promises and perils of mining git. In *Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories*, pages 1–10. IEEE, 2009.

[5] E. Bodden, W. Schäfer, A. van Deursen, and A. Zisman, editors. *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, Paderborn, Germany, September 4-8, 2017*. ACM, 2017.

[6] G. Canfora, L. Cerulo, M. Cimitile, and M. Di Penta. Social interactions around cross-system bug fixings: The case of freebsd and openbsd. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, pages 143–152. ACM, 2011.

[7] M. Chaudron, I. Crnkovic, M. Chechik, and M. Harman, editors. *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*. ACM, 2018.

[8] M. Claes, M. V. 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*, pages 705–715. ACM, 2018.

[9] H. CRAMÉR. *Mathematical Methods of Statistics (PMS-9)*. Princeton University Press, 1999.

[10] A. Cucchiara. Applied logistic regression. *Technometrics*, 44(1):81–82, 1992.

[11] J. Czerwonka, N. Nagappan, W. Schulte, and B. Murphy. Codemine: Building a software development data analytics platform at microsoft. *IEEE Software*, 30(4):64–71, 2013.

[12] 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*, pages 1277–1286. ACM, 2012.

[13] L. K. Dillon, W. Visser, and L. Williams, editors. *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*. ACM, 2016.

[14] 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*, pages 153–162. ACM, 2011.

[15] W. Fu and T. Menzies. Revisiting unsupervised learning for defect prediction. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 72–83. ACM, 2017.

[16] M. Goeminne and T. Mens. A comparison of identity merge algorithms for software repositories. *Science of Computer Programming*, 78(8):971 – 986, 2013. Special section on software evolution, adaptability, and maintenance & Special section on the Brazilian Symposium on Programming Languages.

[17] J. M. González-Barahona, A. Hindle, and L. Tan, editors. *Proceedings of the 14th International Conference on Mining Software Repositories, MSR 2017, Buenos Aires, Argentina, May 20-28, 2017*. IEEE Computer Society, 2017.

[18] G. Gousios, M. Pinzger, and A. v. Deursen. An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering*, pages 345–355. ACM, 2014.

[19] A. E. Hassan. The road ahead for mining software repositories. In *Proceedings of the 2008 Frontiers of Software Maintenance*, pages 48–57. IEEE, 2008.

[20] M. Joblin, S. Apel, C. Hunsen, and W. Mauerer. Classifying developers into core and peripheral: An empirical study on count and network metrics. In *Proceedings of the 39th International Conference on Software Engineering*, pages 164–174. IEEE, 2017.

[21] D. Kim, Y. Tao, S. Kim, and A. Zeller. Where should we fix this bug? a two-phase recommendation model. *IEEE Transactions on Software Engineering*, 39(11):1597–1610, 2013.

[22] M. Kim, R. Robbes, and C. Bird, editors. *Proceedings of the 13th International Conference on Mining Software Repositories, MSR 2016, Austin, TX, USA, May 14-22, 2016*. ACM, 2016.

[23] E. Kouters, B. Vasilescu, A. Serebrenik, and M. G. J. van den Brand. Who's who in gnome: Using lsa to merge software repository identities. In *Proceedings of the 2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 592–595. IEEE, 2012.

[24] C. Kuo and R. Nevatia. How does person identity recognition help multi-person tracking? In *CVPR 2011*, pages 1217–1224. IEEE, 2011.

[25] G. T. Leavens, A. Garcia, and C. S. Pasareanu, editors. *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*. ACM, 2018.

[26] W. Ma, L. Chen, X. Zhang, Y. Zhou, and B. Xu. How do developers fix cross-project correlated bugs?: A case study on the github scientific python ecosystem. In *Proceedings of the 39th International Conference on Software Engineering*, pages 381–392. IEEE, 2017.

[27] C. Mayr-Dorn and A. Egyed. Does the propagation of artifact changes across tasks reflect work dependencies? In *Proceedings of the 40th International Conference on Software Engineering*, pages 397–407. ACM, 2018.

[28] A. Mockus. Amassing and indexing a large sample of version control systems: Towards the census of public source code history. In *Proceedings of the 2009 6th IEEE International Working Conference on Mining Software Repositories*, pages 11–20. IEEE, 2009.

[29] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3):309–346, July 2002.

[30] A. Mockus and J. D. Herbsleb. Expertise browser: A quantitative approach to identifying expertise. In *Proceedings of the 24th International Conference on Software Engineering*, pages 503–512. ACM, 2002.

[31] M. Nagappan, T. Zimmermann, and C. Bird. Diversity in software engineering research. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, pages 466–476. ACM, 2013.

[32] G. A. Oliva, F. W. Santana, K. C. M. de Oliveira, C. R. B. de Souza, and M. A. Gerosa. Characterizing key developers: A case study with apache ant. In V. Herskovic, H. U. Hoppe, M. Jansen, and J. Ziegler, editors, *Proceedings of the 2012 International Conference on Collaboration and Technology*, pages 97–112. Springer Berlin Heidelberg, 2012.

[33] T. Rausch, W. Hummer, P. Leitner, and S. Schulte. An empirical analysis of build failures in the continuous integration workflows of java-based open-source software. In *Proceedings of the 14th International Conference on Mining Software Repositories*, pages 345–355. IEEE, 2017.

[34] P. C. Rigby, D. M. German, L. Cowen, and M.-A. Storey. Peer review on open-source software projects: Parameters, statistical models, and theory. *ACM Trans. Softw. Eng. Methodol.*, 23(4):35:1–35:33, Sept. 2014.

[35] G. Robles and J. M. Gonzalez-Barahona. Developer identification methods for integrated data from various sources. *Acm Sigsoft Software Engineering Notes*, 30(4):1–5, 2005.

[36] B. Rosner, R. J. Glynn, and M. L. Lee. The wilcoxon signed rank test for paired comparisons of clustered data. *Biometrics*, 62(1):185–192, 2010.

[37] I. Steinmacher, G. Pinto, I. S. Wiese, and M. A. Gerosa. Almost there: A study on quasi-contributors in open source software projects. In *Proceedings of the 40th International Conference on Software Engineering*, pages 256–266. ACM, 2018.

[38] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida. Revisiting code ownership and its relationship with software quality in the scope of modern code review. In *Proceedings of the 38th International Conference on Software Engineering*, pages 1039–1050. ACM, 2016.

[39] P. Thongtanunam, C. Tantithamthavorn, R. G. Kula, N. Yoshida, H. Iida, and K. Matsumoto. Who should review my code? a file location-based code-reviewer recommendation approach for modern code review. In *Proceedings of the 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering*, pages 141–150. IEEE, 2015.

[40] S. Uchitel, A. Orso, and M. P. Robillard, editors. *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017*. IEEE / ACM, 2017.

[41] I. S. Wiese, J. T. d. Silva, I. Steinmacher, C. Treude, and M. A. Gerosa. Who is who in the mailing list? comparing six disambiguation heuristics to identify multiple addresses of a participant. In *Proceedings of the 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 345–355. IEEE, 2016.

[42] T. Wolf, A. Schröter, D. Damian, L. D. Panjer, and T. H. D. Nguyen. Mining task-based social networks to explore collaboration in software teams. *IEEE Software*, 26(1):58–66, 2008.

[43] C. B. Wright. Chi-square test. *Annals of Thoracic Surgery*, 33(6):642–643, 1982.

[44] X. Xia and D. Lo. An effective change recommendation approach for supplementary bug fixes. *Automated Software Engineering*, 24(2):455–498, Jun 2017.

[45] Y. Yang, Y. Zhou, J. Liu, Y. Zhao, H. Lu, L. Xu, B. Xu, and H. Leung. Effort-aware just-in-time defect prediction: Simple unsupervised models could be better than supervised models. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 157–168. ACM, 2016.

[46] Y. Yu, H. Wang, G. Yin, and C. X. Ling. Who should review this pull-request: Reviewer recommendation to expedite crowd collaboration. In *Proceedings of the 2014 21st Asia-Pacific Software Engineering Conference*, volume 1, pages 335–342. IEEE, 2014.

[47] A. Zaidman, Y. Kamei, and E. Hill, editors. *Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018*. ACM, 2018.

[48] M. Zhou, Q. Chen, A. Mockus, and F. Wu. On the scalability of linux kernel maintainers' work. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 27–37. ACM, 2017.

[49] M. Zhou and A. Mockus. Who will stay in the floss community? modeling participant's initial behavior. *Software Engineering, IEEE Transactions on*, 41(1):82–99, Jan 2015.

[50] J. Zhu, M. Zhou, and A. Mockus. Effectiveness of code contribution: From patch-based to pull-request-based tools. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 871–882. ACM, 2016.

[51] T. Zimmermann, J. Cleland-Huang, and Z. Su, editors. *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016*. ACM, 2016.