An Empirical Study of Bug Report Field Reassignment

Xin Xia*♭, David Lo†, Ming Wen*, Emad Shihab‡, and Bo Zhou*§
*College of Computer Science and Technology, Zhejiang University, China
†School of Information Systems, Singapore Management University, Singapore
‡Department of Software Engineering, Rochester Institute of Technology, USA
xxkidd@zju.edu.cn, davidlo@smu.edu.sg, justinwm@zju.edu.cn, emad.shihab@rit.edu, bzhou@zju.edu.cn

***Abstract*—Bug fixing is one of the most important activities in software development and maintenance. Bugs are reported, recorded, and managed in bug tracking systems such as Bugzilla. In general, a bug report contains many fields, such as product, component, severity, priority, fixer, operating system (OS), platform, etc. Those fields provide important information for the bug triaging and fixing process. It is important to make sure that bug information is correct since previous studies showed that the wrong assignment of bug report fields could increase the bug fixing time, and even delay the delivery of the software.**

In this paper, we perform an empirical study on bug report field reassignments in open-source software projects. To better understand why bug report fields are reassigned, we manually collect 99 recent bug reports that had their fields reassigned and emailed their reporters and developers asking why these fields got reassigned. Then, we perform a large-scale empirical study on 8 types of bug report field reassignments (i.e., product, component, severity, priority, OS, version, fixer, and status field reassignments) in 4 open-source software projects (i.e., OpenOffice, Netbeans, Eclipse, and Mozilla) containing a total of 190,558 bug reports. In particular, we investigate 1) the number of bug reports whose fields get reassigned, 2) the difference in bug fixing time between bug reports whose fields get reassigned and those whose fields are not reassigned, 3) the duration a field in a bug report gets reassigned, 4) the number of fields in a bug report that get reassigned, 5) the number of times a field in a bug report gets reassigned, and 6) whether the experience of bug reporters affect the reassignment of bug report fields. We find that a large number (approximately 80%) of bug reports have their fields reassigned, and the bug reports whose fields get reassigned require more time to be fixed than those without field reassignments.

Keywords*—*Bug Report Field Reassignment, Empirical Study, Bug Fix

I. Introduction

Software maintenance consumes a large proportion of the cost of a software product. In fact, previous studies show that software maintenance consumes over 70% of the software development cost [8]. Bug fixing is one of the main activities in the software development and maintenance process, and is considered a time-consuming and costly activity. Most software projects use bug tracking systems such as Bugzilla to report, record, and manage bug reports. When reporting a bug report, a reporter needs to provide different types of information about the bug, such as the summary and description text of the observed bug, the status which shows the current status (e.g., *closed* or *resolved*), the product and component fields where the bug is detected, the priority and severity fields which mark the importance of the bug, the version, operating system (OS), and platform fields which indicate the environment affected by the bug, and the reporter and fixer fields. This information is vital for developers to triage and fix the bug [33], [34].

However, in some cases, the fields in the bug report get reassigned. Our analysis shows that approximately 80% of bug reports have their fields reassigned (see Section IV-B) and these reassignments cause delays in the bug fixing process (see Section IV). Figure 1 shows a bug report from Netbeans with BugID 227547.[1] We notice that the fixer, the priority, the product and the component fields in this bug report have been reassigned. The priority is reassigned from `P2` to `P1`, and then reassigned back to `P2`. The fixer is reassigned from `theofanis` to `jtulach`, `vv159170`, `issues`, `jtulach`, and finally it is reassigned to `alexvsimon`. The product is reassigned from `platform` to `cnd`. The component is reassigned from `Options&Settings` to `code`, and finally it is reassigned to `--Other--`.

Since various bug fields could be reassigned, we refer to this phenomenon as the *bug report field reassignment* problem. Previous studies investigated various types of bug report field reassignments. For example, Shihab *el al.* [19], [20] study reopened bugs (i.e., the reassignment of the bug status field), Jeong *el al.* [12] studied fixer reassignments, Sureka [23] and Lamkanfi *et al.* [14] studied the component reassignment in bug reports. All of the aforementioned work considered only one type of reassignment.

To further advance the state-of-the-art in this area, in this paper, we perform an empirical study on bug report field reassignments in open-source software projects. To understand the root cause of bug report field reassignments, we first collect 99 recent submitted bug reports from various open-source software projects. Then, we send emails to the bug reporters and developers of these bug reports, to ask why the fields are reassigned. A total of 21 developers replied to our email, providing useful information about why certain bug report fields get reassigned.

Next, we further investigate the problem of bug report field reassignments in 4 open-source projects (i.e., OpenOffice [4], Netbeans [3], Eclipse [1], and Mozilla [2] containing a total of

♭The work was done while the author was visiting Singapore Management University.
§Corresponding author.

[1] https://netbeans.org/bugzilla/show_bug.cgi?id=227547

190,558 bug reports, and perform an empirical study on 8 types of bug report field reassignments (i.e., product, component, severity, priority, OS, version, fixer, and status). Our study aims to answer a number of research questions: How many bug reports have their fields reassigned? When does a field in a bug report get reassigned? How many fields in a bug report get reassigned? How many times does a field in a bug report get reassigned? Whether the bug reports whose fields get reassigned need more fixing time? Whether the experience of bug reporters would affect the reassignment of bug report fields?

The main contributions of this paper are:

1) We propose the problem of bug report field reassignment, which generalizes all of the different bug report field reassignment studies, and perform a large-scale, semi-automated empirical study on this problem.
2) We manually collect the recently submitted bug reports in various open-source software projects, and ask developers why these fields got reassigned, to understand the root cause of bug report field reassignment.
3) We perform an empirical study on bug report field reassignments on 4 large-scale open-source software projects containing a total of 190,558 bug reports, which could help developers comprehensively understand bug report field reassignment.

The remainder of the paper is organized as follows. We describe the background and a motivating example in Section II. We elaborate the root cause of bug report field reassignments in Section III. We present the empirical study in Section IV. We describe related work in Section V. We present the threats to validity in Section VI. We conclude and mention future work in Section VII.

II. Background and Motivating Example

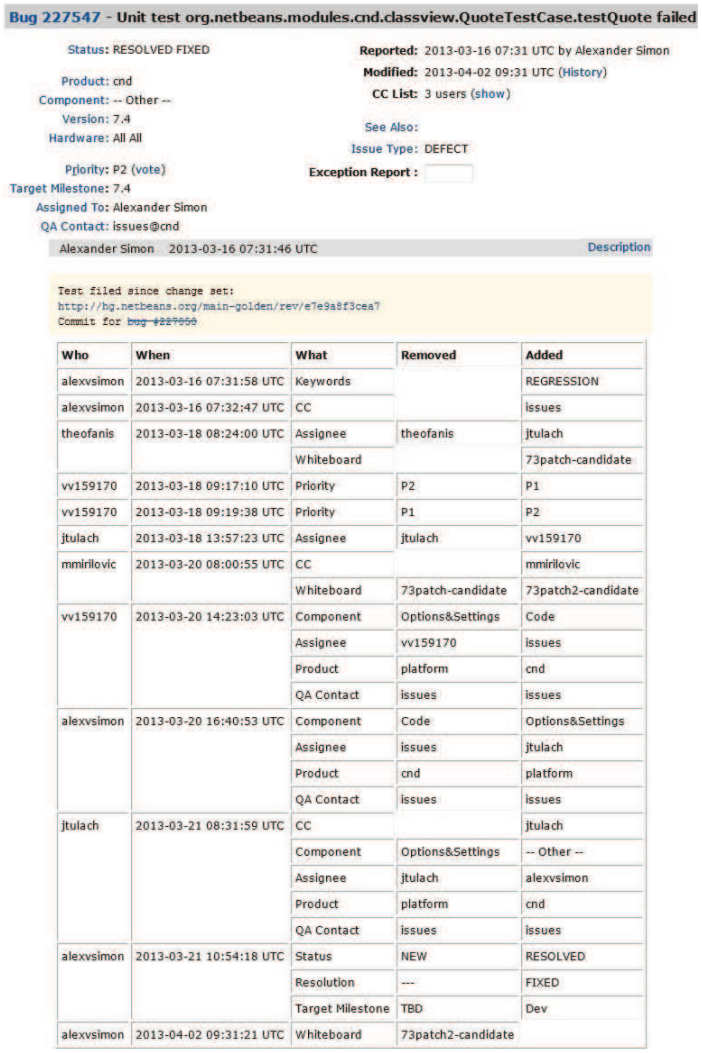
In this section, we first provide background on the different bug report fields. Then, we present a motivating example for the bug report field reassignment problem in order to better understand the bug report field reassignment process.

A. Background

A typical bug report contains many useful fields, such as status, summary, description, reporter, product, component, fixer, severity, priority, operating system (OS), version, creation time and modification time. The details of these bug report fields, illustrated based on the bug report shown in Figure 1, is presented Table I. Since the bug reports in Netbeans do not have the severity field, we set the severity fields in the example to be null in the table. For bug reports in other open-source projects, the severity field is available.

Notice that various fields of a bug report may get reassigned, e.g., product, component, fixer, severity, priority, OS, version. For example, in Figure 1, its fixer, priority, product, and component fields get reassigned. The reassignment of these fields can cause a delay in the bug fixing time.[2]

[2]For more details, please refer to Section IV.



Bug 227547 - Unit test org.netbeans.modules.cnd.classview.QuoteTestCase.testQuote failed

Status: RESOLVED FIXED
Product: cnd
Component: -- Other --
Version: 7.4
Hardware: All All
Priority: P2 (vote)
Target Milestone: 7.4
Assigned To: Alexander Simon
QA Contact: issues@cnd

Reported: 2013-03-16 07:31 UTC by Alexander Simon
Modified: 2013-04-02 09:31 UTC (History)
CC List: 3 users (show)
See Also:
Issue Type: DEFECT
Exception Report :

Alexander Simon 2013-03-16 07:31:46 UTC Description

Test filed since change set:
http://hg.netbeans.org/main-golden/rev/e7e9a8f3cea7
Commit for ~~bug #227050~~

Who	When	What	Removed	Added
alexvsimon	2013-03-16 07:31:58 UTC	Keywords		REGRESSION
alexvsimon	2013-03-16 07:32:47 UTC	CC		issues
theofanis	2013-03-18 08:24:00 UTC	Assignee	theofanis	jtulach
		Whiteboard		73patch-candidate
vv159170	2013-03-18 09:17:10 UTC	Priority	P2	P1
vv159170	2013-03-18 09:19:38 UTC	Priority	P1	P2
jtulach	2013-03-18 13:57:23 UTC	Assignee	jtulach	vv159170
mmirilovic	2013-03-20 08:00:55 UTC	CC		mmirilovic
		Whiteboard	73patch-candidate	73patch2-candidate
vv159170	2013-03-20 14:23:03 UTC	Component	Options&Settings	Code
		Assignee	vv159170	issues
		Product	platform	cnd
		QA Contact	issues	issues
alexvsimon	2013-03-20 16:40:53 UTC	Component	Code	Options&Settings
		Assignee	issues	jtulach
		Product	cnd	platform
		QA Contact	issues	issues
jtulach	2013-03-21 08:31:59 UTC	CC		jtulach
		Component	Options&Settings	-- Other --
		Assignee	jtulach	alexvsimon
		Product	platform	cnd
		QA Contact	issues	issues
alexvsimon	2013-03-21 10:54:18 UTC	Status	NEW	RESOLVED
		Resolution	---	FIXED
		Target Milestone	TBD	Dev
alexvsimon	2013-04-02 09:31:21 UTC	Whiteboard	73patch2-candidate	

Fig. 1. Reassigned Bug Report of Netbeans Project with BugID 227547.

TABLE I. Bug Report Fields

Field	Details	Example
Status	Current status of a bug report, it could be "new", "resolved", "fixed", "closed", "invalid", "duplicate", etc	"Resolved","Fixed"
Summary	Brief description of a bug	... failed
Description	Detail description of a bug	Test filed since change set: ...
Reporter	The developer who submits the bug report	Alexander Simon
Product	Product affected by the bug	cnd
Component	Component affected by the bug	–other–
Fixer	Developer who would fix the bug	Alexander Simon
Severity	After the bug report is submitted, reporter would consider its severity to the system	null
Priority	After the bug report is submited and assigned, the developers would consider the priority to fix this bug	P2
OS	The operating system affected by the bug	All
Version	Version of the source code where the bug appears	7.4
Creation Time	The time the bug is created	2013-03-16 07:31
Modification Time	The time the bug is modified	2013-04-02 09:31

B. Motivating Example

Consider bug ID 227547 in Netbeans, shown in Figure 1. To gain deeper insight into what happened with this bug report, we sent an email to the developers involved in this bug report, and discuss our findings below.

The bug was detected when "two of the C/C++ unit tests failed. However, the C/C++ product (cnd) did not change. The test failed because unexpected messages were printed in the output". To decide on the product and component of the bug, Alexander Simon investigated "change sets of other product teams and found the change set that resulted in the failed tests." Thus, he added a keyword "REGRESSION " to denote that the bug is a regression test bug, and assigned the bug to the `platform` product, `Options&Settings` component, and also the default fixer `vv159170` under the product and component. Moreover, he assigned a of priority `P2` according to the priority guideline in Netbeans.[3]

Then, "the platform team evaluated the bug and reassigned the bug to the C/C++ (cnd) product team". Moreover, since this bug is a regression bug, `vv159170` also reassigned the priority from `P2` to `P1`, since in the bug priority guidelines of Netbeans, regressions which affect the functionality or performance should be `P1`. Later, however, `vv159170` noticed that the bug only produces some warning messages, therefore, the priority is reassigned back to `P2`.

Next, Alexander Simon "slightly disagreed and proposed to downgrade logging from warning to a finer level in the platform and reassigned the issue back to platform product team", because:

1) Foreign changes should be fixed in foreign modules (he had been testing this module for several years). It is an exception when changes in platform (or other clusters) broke C/C++ functionality.
2) If a message is not important, why should it be printed as a warning? [4]

Finally, the "platform team persuaded Alexander Simon that the message is important. And reassigned this bug to the cnd product, and asked for it to be fixed on the C/C++ side".

The above process is common in open-source software projects, and there are many bug report fields that get reassigned. Moreover, some fields could be reassigned multiple times. Understanding the root cause of bug report field reassignments could help us better understand the bug fixing process, which could help to increase software quality and productivity.

III. Root Cause of Bug Report Field Reassignment

To understand the root cause of the bug report field reassignments, we selected 99 recently published bug reports from various open-source software communities and emailed the developers of these bug reports asking why these fields were reassigned. In total, we received 21 replies. Then, we analyzed their replies to determine the root cause of the field reassignments. Note that the 99 bug reports are selected with status "Resolved", "Closed", and "Fixed", and there must be one or more fields in these bug reports that gets reassigned. Table II presents the number of emails we sent to the developers and the number of replies we receive from the developers in Freedesktop, Openoffice, Netbeans, Eclipse, and Mozilla. In the following paragraphs, we first present the general cause of bug report field reassignment, and then we analyze the root cause of each type of field reassignment separately.

A. General Root Causes for Bug Field Reassignments

A typical process of bug fixing is: 1) a reporter submits a bug report; 2) the bug is assigned to a fixer; 3) the fixer tries to fix the bug; 4) the modification of the code is confirmed and verified, and the bug is marked as resolved [29]. The bug report field reassignment could happen in all the different stages of the bug fixing process, and even if the bug is fixed, some of the fields are still reassigned by administrators. In general, we divide the root cause for the bug report field reassignments into 3 categories: new bug report correction, progressing in the process, and admin batch operations.

1) New Bug Report Correction: When a bug report is submitted, some fields could be wrongly assigned. Thus, these fields need to be reassigned, we refer to this kind of bug report field reassignment as *new bug report correction*. When a reporter is new to the open-source project, he/she might submit a bug report where some of the fields are wrongly assigned, e.g., the product and component are not correct. More specifically, one of the replies from Rob Weir, who is an expert in Openoffice project, points out:

> *"In Openoffice, the new bug reports come from two main sources: (1) bug reports submitted by members of the Apache OpenOffice project; (2) bug reports submitted by OpenOffice users. Reports that come from project members tend to be "high quality" reports, with most of the fields entered correctly. Reports that come from end users, are more "raw". They are often incompletely or incorrectly categorized. You might be able to make a distinction here if you look at how many bug reports were entered by a user. Users who enter more bug reports are more likely to be familiar with our process. Those who enter only a single report are probably users."*

2) Progressing in the Process: Bug triaging is a time-consuming and tedious task in software maintenance [5]. During bug triaging, some bug fields get reassigned. In open-source projects, to find the suitable bug fixer, a bug is first assigned to the right product and component [21]. Then, a developer under this product and component would be assigned to fix the bug. Since different developers might have different opinions on the same bug report, the bug report fields might get reassigned. Finally, after some discussion, the suitable values of the fields are determined. For example, in Figure 1, the product, component, and fixer of this bug report are reassigned multiple times. Notice that this kind of bug report field reassignment is different from that of *new bug report correction*, the fields are not wrongly assigned. After a bug report is assigned to a suitable fixer, the fixer still needs to

[3] http://wiki.netbeans.org/BugPriorityGuidelines

[4] By default messages that are severe, warning and information levels are printed in the console. Messages with fine, finer, finest levels can be printed by special IDE option. If messages are so important for the platform team, they can start the IDE with a special option to be sure that all is OK.

Who	When	What	Removed	Added
stephan.wunderlich	2004-11-17 15:00:21 UTC	Status	NEW	STARTED
stephan.wunderlich	2004-11-17 17:06:53 UTC	Status	STARTED	RESOLVED
		Resolution	---	FIXED
svante.schubert	2004-11-30 14:29:44 UTC	Status	RESOLVED	VERIFIED
stephan.wunderlich	2005-01-26 16:14:28 UTC	Status	VERIFIED	CLOSED
robweir	2013-02-24 21:09:46 UTC	Component	code	api
		Version	current	pre AOO 3.4.0
		Product	api	App Dev
		Target Milestone	OOo 2.0	---

Fig. 2. Example of Bug Report in OpenOffice Project with BugID 34887.

TABLE II. NUMBER OF EMAILS (# EMAILS) WE SEND AND NUMBER OF REPLIES (# REPLIES) WE RECEIVE IN THE 5 OPEN-SOURCE PROJECTS.

Project	# Emails	# Replies
Freedesktop	18	3
Openoffice	20	2
Netbeans	29	9
Eclipse	18	4
Mozilla	14	3
Total	99	21

consider when to fix the bug report. He would set a priority to the bug to denote the importance of it. However, some other developers (i.e., bug resolvers) also argue about the suitable priority of the bug report, which causes this field to get reassigned. We refer to the kind of bug field reassignment, which happens in the process of bug fixing as *progressing in the process*.

3) Admin Batch Operations: To help avoid the above two kinds of bug report field reassignments, administrators also reassign some fields in the bug reports to better organize the project. We refer to this kind of bug report field reassignment as *admin batch operations*. As Rob Weir states:

> *"We occasionally remove, move or combine components, causing us to re-categorize many defects in one large operation. Or we may unassign old issues that no one is currently working on. These "house cleaning" operations should not be confused with progressing in the process above. Maybe you can detect and ignore transactions from me ("robweir") when the operations are clearly batch operations, e.g., a high transaction rate for the same kind of change."*

For example, Figure 2 presents an example of bug report which has the reassignment type of *admin batch operations*. Notice this bug fixed in "2005-01-26" by `stephan.wunderlich`, but in "2013-02-24", Rob Weir reassigned the product and component.

B. Root Causes for the Reassignment of Specific Bug Report Fields

Besides the above 3 general root causes, each field of a bug report has its own reasons to be reassigned. In this paper, we mainly focus on 8 types of bug report field reassignments, i.e., product, component, severity, priority, OS, version, fixer, and status, which are common fields in bug reports of open-source projects. In the following paragraphs, we analyze the root cause for each of these 8 fields in detail.

1) Product, Component, and Fixer Field Reassignment: The root cause of product field reassignment can be due to any of the 3 above root causes, i.e., *new bug report correction, progressing in the process, or admin batch operations*. Also, the reassignment of some fields can cause other fields to be reassigned. For example, the component field gets reassigned due to a reassignment in the product field, and the fixer gets reassigned due to the fact that the component field got reassigned. In many open-source projects, there are default values for the component and default fixer for a specific product. If some developers reassign the product field, the component also gets reassigned to the default component and fixer. For example, in Figure 1, `vv159170` just reassigned product from `platform` to `cnd`. However, the component field is reassigned to a default component `Code` under `cnd`, and the fixer field is also reassigned to a default fixer `issues` under the component `Code`.

2) Severity and Priority Field Reassignment: The root cause for the severity and priority field reassignments belongs to 2 of the aforementioned root causes, i.e., *new bug report correction*, and *progressing in the process*. Severity and priority are both assigned according to some guidelines, and if the assignment disobeys the guideline, these fields would get reassigned. For example, for the bug report 68956 in Freedesktop,[5] the developer reassigns the severity and priority because "normal and medium didn't fit the bug problem. If something gets destroyed by the application this has high severity and high priority". In practice, the correct assignment of these 2 fields are difficult, as one of the developer in Eclipse states:

> *"Severity is subjective, so it gets twiddled for no good reason. Priority is used in different ways by different people."*

Thus, most of the time, the severity and priority fields get reassigned in the bug fixing process (i.e., *progressing in the process*). Moreover, there are some differences between severity and priority: severity is assigned by bug reporters, and priority is assigned by developers; the assignment of severity would affect the developers when they assign a priority to a bug report [27]. The difference means that sometimes, the priority field is reassigned due to the fact that the severity field got reassigned.

3) OS and Version Field Reassignment: The root cause for the OS and version fields to be reassigned could belong to any 3 of the above root causes, i.e., *new bug report correction, progressing in the process, and admin batch operations*. During the bug fixing process, the OS field could be reassigned to "All" if "a user reports a defect on a specific platform/os (e.g. PC/Windows) and during evaluation we (the developers) figure out that bug is platform independent". For the version field, it is the same, i.e., the version fields would be reassigned, if a user reports a defect on a specific version, and later developers figure out that the bug belongs to another version. Thus, most of the time, the root cause of the OS and version fields to be reassigned belong to *progressing in the process*.

4) Status Reassignment: In this paper, we only consider one type of status reassignment: *resolved* or *closed* to *reopen*. The root cause of the status field reassignment could be due

[5] https://bugs.freedesktop.org/show_bug.cgi?id=68956

TABLE III. STATISTICS OF COLLECTED BUG REPORTS.

Project	Time	# Reports	# Reporter	# Fixer	# Product	# Component	# Version	# OS	# Platform
OpenOffice	2002-05-17 – 2013-04-07	42,169	5,451	701	140	106	546	45	12
Netbeans	2008-01-01 – 2013-03-13	46,345	5,709	323	112	684	43	26	7
Eclipse	2008-01-01 – 2011-07-19	50,639	5,824	1,021	143	702	220	31	6
Mozilla	2009-06-23 – 2012-02-23	51,405	3,536	696	51	620	107	36	10

TABLE IV. NUMBERS AND FRACTIONS OF BUG REPORTS BELONGING TO THE 8 TYPES OF FIELD REASSIGNMENTS, I.E., PRODUCT, COMPONENT, SEVERITY, PRIORITY, OS, VERSION, FIXER, AND STATUS REASSIGNMENTS.

Project	Re-Product	Re-Component	Re-Severity	Re-Priority	Re-OS	Re-Version	Re-Fixer	Re-Status
OpenOffice	5,956 (14.12%)	5,960 (14.13%)	392 (0.93%)	4,428 (10.50%)	2,439 (5.78%)	4,688 (11.11%)	31,511 (74.73%)	11,192 (26.54%)
Netbeans	14,554 (31.40%)	29,681 (64.04%)	0 (0%)	7,576 (16.35%)	2,219 (4.79%)	2,797 (6.04%)	23,653 (51.04%)	4,926 (10.63%)
Eclipse	4,940 (9.76%)	9,338 (18.44%)	4,652 (9.19%)	5,495 (10.85%)	2,305 (4.55%)	6,251 (12.34%)	32,826 (64.82%)	4,124 (8.14%)
Mozilla	9,905 (19.27%)	12,686 (24.68%)	3,716 (7.23%)	5,857 (11.39%)	6,342 (12.34%)	4,833 (9.40%)	34,195 (66.52%)	4,787 (9.31%)

to any 2 of the aforementioned root causes, i.e., *new bug report correction*, and *progressing in the process*. Zimmermann et al. conclude that there are 6 different root causes for the reassignment of the bug status field, i.e., difficult to reproduce the bug, the misunderstanding of the root cause, insufficient information, priority increased, regression bugs, and process-related bug [32]. We also find one more root cause for the status field reassignment: the misunderstanding between developers, i.e., there is insufficient communication between developers which causes the bug get reopened. For example, for the bug report 63211 in Freedesktop,[6], a developer states the root cause of this bug is:

> *"In the case of this particular bug I think the developer originally believed that the bug was invalid and closed. I have been fortunate that another developer has picked up on this and committed a patch. I think the bug was fixed, so I close the bug without discussion of the developers. However, later a developer told me the bug still existed. In this case I have reopened it because I believe it was closed because of a misunderstanding."*

IV. EMPIRICAL STUDY

Previous sections mainly focus on the investigation of the developers in open-source projects, to understand the problem of bug report field reassignment, and the root cause of it. There has been anecdotal evidence that bug report fields get reassigned, however, we would like to empirically examine the bug report field reassignment phenomenon. In this section, we first describe the data we collect in Section IV-A, and then we elaborate on the 6 research questions we would like to investigate in Section IV-B. Finally, we present the answers for the 6 research questions.

A. Data Collection

Table III shows the statistics of the 4 projects we use to conduct our empirical study. All of the bug reports and data are downloaded from their corresponding bug tracking systems. We collected all bug reports with the status "resolved", "closed", and "fixed" following previous studies [12], [14], [20], [23], [24]. In Table III, columns **Time** and **# Report** correspond to the time periods the collected bug reports are reported and the number of collected reports, respectively. In total, we analyze 190,558 bug reports.

For the other columns, we list the number of unique values of the different fields: reporter, fixer, product, component, version, OS, and platform. Notice that we record the values of these fields at the time the bug is reported, i.e., the values of all of these fields are recorded before the bug report is reassigned. For example, in Figure 1, since the fixer, product, and component of the bug report are reassigned, we record its fixer, product, and component before the reassignment, i.e., `theofanis`, `platform`, and `Options&Settings`, respectively. In OpenOffice, the number of products is more than that of components, we double checked the dataset, and we find that indeed that is the case.

B. Research Questions

In this paper, we are interested in answering these research questions:

RQ1: *How many bug reports have their fields reassigned?*

In our study, we focus on 8 types of bug report field reassignments. Since different bug reports could have a different number of their fields reassigned, in this research question, we would like to determine the number of bug reports that have some of their fields reassigned. To answer this research question, we count the number of bug reports that have any of the 8 types of bug report field reassignments.

Table IV shows results of the number and fraction of bug reports of the 8 reassignments. We observe that the fraction of bug reports whose fixers have been reassigned are high. On average across the 4 projects, 64.28% of the bug reports have their fixers reassigned. For other bug report field reassignment types, the class imbalance phenomenon exists, i.e., the number of reassigned bug reports is much smaller than the non-reassigned ones. For example, in Eclipse and Mozilla, only 8.14% and 9.31% of the bug reports have their status reassigned to *reopen*, and 9.19% and 7.23% bug reports have their severity reassigned. Notice that for each type of field reassignment (except for the fixer reassignment), the number of bug reports whose fields have been reassigned is much smaller than the number of bug reports without reassignment, i.e., class imbalance phenomenon is observed [10].

[6]https://bugs.freedesktop.org/show_bug.cgi?id=63211

TABLE V. NUMBERS AND PERCENTAGES OF FIELDS IN BUG REPORTS THAT GET REASSIGNED.

Projects	0	1	2	3	≥4
OpenOffice	7,921 (18.78%)	15,036 (35.66%)	11,509 (27.29%)	3,836 (9.10%)	3,867 (9.17%)
NetBeans	9,091 (19.62%)	9,833 (21.22%)	12,775 (27.57%)	9,947 (21.46%)	4,699 (10.14%)
Eclipse	11,402 (22.52%)	21,299 (42.06%)	9,873 (19.50%)	4,580 (9.04%)	3,485 (6.88%)
Mozilla	10,124 (19.69%)	185,68 (36.12%)	11,485 (22.34%)	6,378 (12.41%)	4,850 (9.43%)

In Netbeans, the fraction of product and component reassignments are higher than the other 3 projects: 31.40% and 64.04% bug reports have their product and component fields reassigned. Moreover, we notice that no bug report had its severity changed in Netbeans, however, the fraction of bug reports which have their priority reassigned is higher than the other 3 projects.

The fractions of bug reports whose product, component, severity, priority, OS, version, fixer and status fields get reassigned vary from 9.76%-31.40%, 14.13%-64.04%, 0%-9.19%, 10.50%-16.35%, 4.55%-12.34%, 6.04%-12.34%, 51.04%-74.73%, and 8.14%-26.54%, respectively. Except for the fixer reassignment, we notice that for most types of bug report field reassignment only a minority of bug reports have their fields reassigned.

RQ2: *How many fields in a bug report get reassigned?*

Since each bug report could have multiple types of bug report field reassignment, in this research question, we would like to compare the number of bug reports that do not have any field reassigned to those that have some of their fields reassigned. To answer this research question, we count the number of fields in a bug report that get reassigned.

Table V presents the numbers and percentages of fields in bug reports that get reassigned in OpenOffice, Netbeans, Eclipse, and Mozilla. The number of bug reports which do not have any type of field reassignments is low. There are only 18.78%, 19.62%, 22.52%, and 19.69% bug reports in OpenOffice, Netbeans, Eclipse, and Mozilla respectively where none of their fields are reassigned.

We notice that most of the bug reports have at least 1 or 2 types of field reassignment, i.e., 62.95%, 48.79%, 61.56%, and 58.46% bug reports in OpenOffice, Netbeans, Eclipse, and Mozilla, respectively. We also notice that the percentage of bug reports which have more than 4 types of field reassignment is around 10%. From Table V, we could conclude that bug report field reassignments is a common activity in open-source software development.

Most of the bug reports have 1 or 2 types of field reassignments, and on average, across the 4 projects, these bug reports are around 58% of the total number of bug reports. The number of bug reports with no field reassignment is low, which is around 20% of the total number of bug reports in the 4 projects.

RQ3: *How many times does a field in a bug report get reassigned?*

We observe that a field in a bug report may be reassigned multiple times. Therefore, in this research question, we would like to investigate whether some fields get reassigned more times than others. To answer this research question, for each field, we record the number of bug reports for which the field is reassigned a given number of times (i.e., 1, 2, 3, and ≥ 4).

TABLE VII. MEAN, MEDIAN, MAX, AND MIN VALUES (HOURS) FOR THE TIME DURATION BETWEEN BUG REPORT CREATION AND THE FIRST REASSIGNMENT IN OPENOFFICE.

Reassignment	Mean	Median	Max	Min
Re-Product	32,942	31,144	94,281	0
Re-Component	34,398	34,334	94,281	0
Re-Severity	9,428	10,456	73,999	0
Re-Priority	1	1	1	0
Re-OS	2,116	82	54,042	0
Re-Version	34,758	37,719	94,281	0
Re-Fixer	1,235	105	89,783	0
Re-Status	2,599	823	68,640	0

Table VI presents the number of bug reports for which various fields are reassigned a given number of times. For most bug reports, most fields are only reassigned once. For example, in Eclipse, there are 4,940 and 9,338 bug reports whose product and component are reassigned respectively, but only 386 and 1,333 bug reports reassign their product and component more than once, respectively.

For fixer reassignment, the number of bug reports which have been reassigned more than 4 times is high, i.e., 3,735, 1,079, 636, and 719 for OpenOffice, Netbeans, Eclipse, and Mozilla, respectively. These numbers indicate that bug triaging work is not an easy job. Thus, automated methods are needed to better recommend fixers [5], [12], [24]. However, for the other types of field reassignment, the number of bug reports whose fields are reassigned is low. For example, there are only 1, 7, 1, and 14 bug reports whose OS field is reassigned more than 4 times in OpenOffice, NetBeans, Eclipse, and Mozilla, respectively.

Among bug reports requiring field reassignments, to decide the right value for each bug report field, most of the bug reports just need to have their fields reassigned once. For fixer reassignment, the number of bug reports whose fixers are reassigned more than once is high compared to other types of field reassignment.

RQ4: *When does a field in a bug report get reassigned?*

We conjuncture that different bug report fields get reassigned at different times. Therefore, in this research question, we would like to investigate whether some fields get reassigned later/earlier than other fields. To answer this research question, for each type of field reassignment, we extract bug reports whose fields are reassigned. Next, we extract their creation timestamps, and the timestamps when one of their fields are reassigned for the first time. The difference between these two timestamps is the time duration that has passed till the field got reassigned.

TABLE VI. NUMBER OF TIMES THE FIXER, PRODUCT, COMPONENT, SEVERITY, PRIORITY, OS, AND VERSION FIELDS GET REASSIGNED.

	OpenOffice				Netbeans				Eclipse				Mozilla			
Reassignment	**1**	**2**	**3**	**≥4**	**1**	**2**	**3**	**≥4**	**1**	**2**	**3**	**≥4**	**1**	**2**	**3**	**≥4**
Re-Product	5,504	420	26	6	13,433	843	191	87	45,54	333	39	14	6,945	2,776	151	33
Re-Component	5,567	367	25	1	23,986	4,311	979	405	8,005	1,085	181	67	8,783	3,469	321	116
Re-Severity	387	4	1	0	0	0	0	0	4,220	361	56	15	3,166	435	79	36
Re-Priority	3,825	517	66	20	6,048	1,146	260	122	4,893	473	79	50	5,095	563	145	54
Re-OS	2,353	80	5	1	2,139	55	18	7	2,274	24	6	1	5,984	265	79	14
Re-Version	4,449	208	29	2	2,563	202	27	5	5,472	673	73	33	4,294	417	98	24
Re-Fixer	15,415	8,041	4,320	3,735	15,868	5,038	1,668	1,079	25,450	5,273	1,467	636	28,198	3,865	1,413	719

TABLE VIII. MEAN, MEDIAN, MAX, AND MIN VALUES (HOURS) FOR THE TIME DURATION BETWEEN BUG REPORT CREATION AND THE FIRST REASSIGNMENT IN IN NETBEANS.

Reassignment	Mean	Median	Max	Min
Re-Product	6,155	5,191	39,019	0
Re-Component	5,766	4,688	27,815	0
Re-Severity	0	0	0	0
Re-Priority	1	1	1	0
Re-OS	1,437	65	39,019	0
Re-Version	3,175	76	39,019	0
Re-Fixer	872	23	42,943	0
Re-Status	1,625	296	38,755	0

TABLE IX. MEAN, MEDIAN, MAX, AND MIN VALUES (HOURS) FOR THE TIME DURATION BETWEEN BUG REPORT CREATION AND THE FIRST REASSIGNMENT IN ECLIPSE.

Reassignment	Mean	Median	Max	Min
Re-Product	12,151	6,406	44,734	0
Re-Component	7,262	319	44,734	0
Re-Severity	917	22	39,068	0
Re-Priority	1	1	1	0
Re-OS	1,894	51	42,737	0
Re-Version	5,167	271	44,734	0
Re-Fixer	846	16	40,890	0
Re-Status	1,949	378	45,283	0

Tables VII, VIII, IX and X present the time durations of various reassignments happening in the bug reports of the 4 projects, OpenOffice, Netbeans, Eclipse, and Mozilla, respectively. We record the mean, median, maximum and minimum values in hours.

The product and component reassignments happen latter as compared to other field reassignment types. Detecting the proper product and component is not an easy job, and previous studies show that changes in the component of a bug report is positively correlated with fixer reassignments [9], [21]. If we could detect whether the product and component of a bug report would be reassigned, we could potentially save a large amount of time during the bug fixing process. In OpenOffice, the durations are longer, i.e., the median values are 31,144 and 34,334 hours (3.55 and 3.92 years) for product and component reassignments, respectively. We double checked the dataset, and we notice that the product and component fields of some bug reports in OpenOffice changed even after the bug reports are fixed and closed by quality assurance personnel. Figure 2 shows an example, we notice this bug fixed in "2005-01-26" by `stephan.wunderlich`, but in "2013-02-24", `robweir` reassigned the product and component. This reassignment is due to admin batch operations, i.e., these fields are reassigned to better organize the project.

TABLE X. MEAN, MEDIAN, MAX, AND MIN VALUES (HOURS) FOR THE TIME DURATION BETWEEN BUG REPORT CREATION AND THE FIRST REASSIGNMENT IN MOZILLA.

Reassignment	Mean	Median	Max	Min
Re-Product	10,060	8,619	34,003	0
Re-Component	7,225	3,340	34,003	0
Re-Severity	712	23	28,701	0
Re-Priority	1	1	1	0
Re-OS	1,117	44	29,942	0
Re-Version	1,726	36	28,631	0
Re-Fixer	969	28	31,018	0
Re-Status	1,499	331	30,965	0

Aside from the product and component reassignments, we notice that status reassignments (i.e., to *reopen*) also takes a long time to be detected. The median time to detect the status reassignments are 823, 296, 378, and 331 hours for OpenOffice, Netbeans, Eclipse, and Mozilla, respectively, which is much longer than the median time of fixer, OS, severity, version, and priority reassignments.

We also observe that the priority field is reassigned within a short period of time. We sampled many bug reports to see the reason, and we find that in most of the bug reports whose priority fields are reassigned, the priority fields are first set to the default value. Later, the developers would decide the proper priority, i.e., reassign the priority from the default value to a proper value.

> *Product and component reassignments take a very long time to be detected. Aside from them, status reassignments also take a long time to be detected. On the other hand, priority reassignments is made within a short period of time.*

RQ5: *Do bugs that have their fields reassigned need more time to fix?*

Various factors could affect the bug fixing time, e.g., bug owners and bug types [30]. In this research question, we investigate whether the bug report field reassignment would increase the bug fix time. To answer this research question, we first divide the bug reports into two disjoint sets: bug reports whose fields get reassigned (reassigned bugs), and bug reports where none of the fields get reassigned (non-reassigned bugs). Next, we compute the bug fixing time in these two sets. The bug-fixing time is measured as the time duration from the

TABLE XI. MEAN, MEDIAN, MAXIMUM, AND MINIMUM BUG FIX TIME (HOURS) FOR REASSIGNED AND NON-REASSIGNED BUGS. THE LAST COLUMN SHOWS THE P-VALUE OF MWW TEST.

Project	Reassigned Bugs				Non-reassigned Bugs				P-value
	Mean	Median	Max	Min	Mean	Median	Max	Min	
Openoffice	13,888	4,672	94,281	0	7,922	3,410	90,408	0	$2.2e^{-16}$
Netbeans	3,429	952	47,059	0	1,248	201	30,584	0	$2.2e^{-16}$
Eclipse	5,979	1,868	46,734	0	4,009	669	46,489	0	$2.2e^{-16}$
Mozilla	6,020	1,913	34,003	0	2,886	501	32,894	0	$2.2e^{-16}$

creation of a bug report to the time the bug is resolved as fixed. Then, we compute the mean, median, maximum, and minimum bug fixing time across the two sets, and perform a Mann-Whitney-Wilcoxon (MWW) test [17] to compare the bug fixing time of the bug reports in the two sets.

Table XI presents the mean, median, maximum, minimum bug fix time for the reassigned and non-reassigned bugs across the 4 projects. The mean and median bug fix time for reassigned bugs are much more than these of non-reassigned bugs. For example, in Netbeans, the mean and median bug fix time for reassigned bugs are 3,429 and 952 hours respectively, but for the non-reassigned bugs they are 1,248 and 201 hours respectively. The bug fix time for reassigned bugs are almost 3 times more than these of non-reassigned bugs in Netbeans. Moreover, the median bug fix time for reassigned bugs are 4,672, 952, 1,868, and 1,913 hours in Openoffice, Netbeans, Eclipse, and Mozilla, respectively; and these values of non-reassigned bugs are 3,410, 201, 669, and 501 hours, respectively. The median bug fix time for reassigned bugs are almost 3 times more than the values of non-reassigned bugs.

The Mann-Whitney-Wilcoxon (MWW) test [17] showed that the differences in bug fix time of reassigned and non-reassigned bugs are statistically significant, for all of the 4 projects, the p-values are $2.2e^{-16}$. Thus, we conclude that bugs with reassigned fields take more time to fix than bug with no reassigned fields.

The time required to fix a reassigned bug is almost 3 times that of a non-reassigned bug. The difference of fix time between reassigned bugs and non-reassigned bugs are statistically significant, the p-values are $2.2e^{-16}$ in all of the 4 projects.

RQ6: *Does the experience of bug reporters affect the field reassignments?*

As described in Section III, there are some "raw" users who do not have enough experience to submit bug reports. In this research question, we would like to investigate whether the experience of reporters affects the field reassignment. To answer this research question, we measure the experience of reporters as the number of bug reports they submitted in one year. For example, if we have a bug report submitted in "2013-10-14", and the reporter is `Tom`. We would compute the number of bug reports submitted by `Tom` from "2012-10-14" to "2013-10-14". The reason we do this setting is that we want to remove the noise due to a long time spans. For example, `Tom` maybe quite active from "1998-10-14" to "1999-10-14", and submit 10,000 bug reports. But after that time, he leaves the project, and then he is back in "2013-10-14". If we measure the experience of `Tom` across the entire time spans of the project, then `Tom` would be considered as an experienced developer. However, he did not work for the project more than 10 years.

TABLE XII. SPEARMAN'S RHO AND CORRELATION LEVEL [11].

Spearman's Rho	Correlation Level
0.0 - 0.1	None
0.1 -0.3	Small
0.3 -0.5	Moderate
0.5 - 0.7	High
0.7 - 0.9	Very High
0.9 - 1.0	Perfect

Moreover, for each bug report, we also record the number of fields that get reassigned.

To answer this question, we use Spearman's ρ, which is a non-parametric measure used to measure the strength of monotonic relationship between sets of data [22], is used to evaluate whether there would be correlation between the experience of reporters and the number of bug report fields get reassigned. The ρ value ranges from -1 to 1, where these extreme values depict a perfect monotonic correlation. A value of 0 shows that the variables are independent of each other. Table XII describes the meaning of various Spearman's rho values and their corresponding correlation level [11].

Table XIII presents the p-value, Spearman's ρ value, and correlation level between the experience of reporters and number of fields get reassigned in the 4 projects. Column Time corresponds to the time period of the selected bug reports. Notice that we set the begin date as one year later than the bug reports we collected in Table III. For example, in Netbeans, our collected bug reports is from "2008-01-01" in Table III, and we set the begin date as "2009-01-01". Thus, we can compute the one-year experience of bug reporters.

From Table XIII, we observe that the difference between the experience of reporters and the number of fields that get reassigned is significant, since all of the p-values in the 4 projects are $2.2e^{-16}$. And the Spearman's rho values int the 4 projects are negative, which mean there would be negative correlation relationship, i.e., the more experience a reporter has, the less bug report fields would get reassigned when he submits a bug report. However, the correlation relationship is very weak. According to table XII, the correlation level for Netbeans and Eclipse are none, and these for Openoffice and Mozilla are small. Thus, our findings do not suggest any relationship between reporter experience and bug report field reassignments.

In Netbeans and Eclipse, there is no correlation between the experience of reporters and the number of bug report fields reassignments. In Openoffice and Mozilla, the correlation is very small.

TABLE XIII. THE P-VALUE, SPEARMAN'S RHO VALUE, AND CORRELATION LEVEL BETWEEN EXPERIENCE OF REPORTERS AND NUMBER OF FIELDS GET REASSIGNED.

Project	Time	# Reports	p-value	Spearman's Rho	Correlation Level
Openoffice	2003-05-17–2013-04-17	38,799	$2.2e^{-16}$	-0.1380	**Small**
Netbeans	2009-01-01–2013-03-13	31,512	$2.2e^{-16}$	-0.0931	**None**
Eclipse	2009-01-01–2011-07-09	27,748	$2.2e^{-16}$	-0.0864	**None**
Mozilla	2010-06-23–2012-02-23	23,649	$2.2e^{-16}$	-0.1139	**Small**

V. RELATED WORK

A. Bug Report Field Reassignments

There have been a number of studies on bug report field reassignments. Guo *et al.* perform an empirical study on fixer reassignments, and they find five primary reasons for fixer reassignments, i.e., difficulty to identify the root cause, ambiguous ownership of components, poor bug report quality, difficulty to determine the proper fix, and workload balancing [9]. Shihab *et al.* propose a machine learning based method to predict reopened bugs; they extract 4 groups of features, related to work habits, bug report fields, bug fix, and people, containing a total of 24 features [19], [20]. Lamkanfi *et al.* propose the usage of Naive Bayes to predict whether the component of a bug report would be reassigned, and their method achieves precision and recall between 0.58-0.94 and 0.54-1 for bug reports of several products in Eclipse and Mozilla [14]. Jeong *et al.* investigate fixer reassignments in Mozilla and Eclipse, and they propose a method to use fixer reassignment graph to improve the performance of bug triaging [12]. Bhattacharya *el at.* extend Jeong *et al.'s* work to improve the accuracy of bug triaging by using multi-feature fixer reassignment graph [6]. Our work generalizes the above studies; previous studies focus on single bug report field reassignment, while our work considers all field reassignments simultaneously.

B. Bug Report Field Prediction

There have been many studies on bug report field prediction. Bug triaging predicts the fixer field in a bug report, and there are a number of machine learning and information retrieval approaches proposed for bug triaging [5], [6], [12], [24]. There are also many studies that predict the severity labels of bug reports [15], [18], [26], and priority labels of bug reports [13], [27]. Recently, several studies predict the components of bug reports [21], [23]. Somasundaram *et al.* propose the usage of a topic model to predict the component of bug reports [21]. Sureka proposes the usage of a TF.IDF classifier and a dynamic language model classifier to predict the component of a bug report [23]. Our work is orthogonal to the above studies; in our study, we perform an empirical study on the reassignment of the fields studied in the aforementioned work, e.g., fixer, severity, priority, component.

C. Empirical Studies

There have been many empirical studies on bug report management. Thung *et al.* [25] perform an empirical study of bugs in machine learning systems. They investigate three open source machine learning system: Apache Mahout, Lucene, and OpenNLP. Lu *et al.* [16] investigate concurrency bugs in MySQL, Apache Web Server, Mozilla, and OpenOffice. Bhattacharya et al. perform an empirical study on bug reports and bug fixing in open source Android applications [7]. Our work is orthogonal to the above studies; in our study, we perform an another empirical study which focuses on the bug report field reassignment.

VI. THREATS TO VALIDITY

There are several threats that may potentially impact the validity of our empirical study. Threats to internal validity relates to experimenter bias and errors. Our study investigate the bugs in 4 large-scale open-source software projects, and all of the bug reports are collected from their corresponding bug tracking systems. Moreover, the process of bug fixing are under strict control. Thus, we believe the historical actions for the bug reports are the actual actions for the bug reports. Also we select the recent submitted bug reports, and confirm that there are reassignment activities in these bug reports, and send emails to the developers who are related to these reports. Thus, our study is done under the fact of bug reports, and none of the developers in open-source projects have any motivation to influence our results in either way.

Threats to external validity relates to the generalizability of our study. We have analyzed 4 open-source software projects: Openoffice, Netbeans, Eclipse, and Mozilla. To improve the generalizability of our study, we collect a large number of bug reports which contains a total of 190,558 bug reports across a long time period. Moreover, the bug reports are all in the status of "resolved", "fixed" and "closed", and the type of bug reports is "defect". By doing this, we remove the potential noise caused by duplicate bug report, invalid bug reports, or feature request and enhancement. We believe that we collected enough bug reports to prove the findings in our study. We plan to reduce this threat to external validity in the future by analyzing more bug reports from more open-source software projects, and industrial projects.

VII. CONCLUSION AND FUTURE WORK

In this paper, we perform an empirical study on bug report field reassignments. To understand the root cause of bug report field reassignment, we first send emails to the developers in open-source software projects to ask the root cause for the bug report field reassignment, and based on their replies, we conclude 3 general root cause: *new bug report correction*, *progressing in the process*, and *admin batch operations*. Next, we analyze bug reports on 4 open-source projects, i.e, OpenOffice, Netbeans, Eclipse, and Mozilla, which contain a total of 190,558 bug reports. By analyzing these 4 projects, we investigate 6 research questions such as the bug fix time between the bug reports whose fields get reassigned and those whose fields are not reassigned, the number of bug reports whose fields get reassigned, the time duration a field in a bug report gets reassigned, the number of fields in a bug report that get reassigned, the number of times a field in a bug report gets reassigned, and whether the experience of bug reporters affect the reassignment of bug report fields. We find that bug

report field reassignments could cause a delay in the bug fix, and it is a common phenomenon in open-source projects, approximately 80% of bug reports have their fields reassigned. Moreover, the experience of reporters could limit affect the bug report field reassignment when they submit bug reports.

In the future, we plan to evaluate our results from more bug reports in more software projects, both from open-source projects, and industrial projects. We also plan to develop an automated tool to detect which fields in a bug report would get reassigned. Since multiple fields in a bug report would get reassigned, we would refer to multi-label learning algorithms [28] to solve the problem. For example, we can leverage ML.KNN [31], which is a state-of-the-art multi-label learning algorithm, to predict which fields would get reassigned.

ACKNOWLEDGMENT

This research is sponsored in part by NSFC Program (No.61103032) and National Key Technology R&D Program of the Ministry of Science and Technology of China (No2013BAH01B03). The authors would thank Alexander Simon, Rob Weir, Jaroslav Havlin, Ferry Toth, Jaroslav Havlin, Boris Zbarsky, Jesse Glick, Regina Henschel, Mark Wilmoth, Marek Fukala, Thomas Arnhold, Milos Kleint from the development teams of open-source software projects, to provide us valuable suggestions and comments for the root cause of bug report field reassignment. All the replies of the emails can be download from https://www.dropbox.com/s/ws8xu4wdese07sc/Emails.zip.

REFERENCES

[1] Eclipse bug tracking system. *https://bugs.eclipse.org/bugs/*.

[2] Mozilla bug tracking system. *https://bugzilla.mozilla.org/*.

[3] Netbeans bug tracking system. *http://netbeans.org/bugzilla/*.

[4] Openoffice bug tracking system. *https://issues.apache.org/ooo/*.

[5] 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.

[6] P. Bhattacharya and I. Neamtiu. Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. In *Software Maintenance (ICSM), 2010 IEEE International Conference on*, pages 1–10. IEEE, 2010.

[7] P. Bhattacharya, L. Ulanova, I. Neamtiu, and S. C. Koduru. An empirical analysis of bug reports and bug fixing in open source android apps. In *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, pages 133–143. IEEE, 2013.

[8] B. Boehm and V. Basili. Software defect reduction top 10 list. *Foundations of empirical software engineering: the legacy of Victor R. Basili*, 2005.

[9] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy. Not my bug! and other reasons for software bug report reassignments. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, pages 395–404. ACM, 2011.

[10] H. He and E. A. Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[11] W. G. Hopkins. *A new view of statistics*. Will G. Hopkins, 1997.

[12] G. Jeong, S. Kim, and T. Zimmermann. Improving bug triage with bug tossing graphs. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 111–120. ACM, 2009.

[13] F. Khomh, B. Chan, Y. Zou, and A. E. Hassan. An entropy evaluation approach for triaging field crashes: A case study of mozilla firefox. In *Reverse Engineering (WCRE), 2011 18th Working Conference on*, pages 261–270. IEEE, 2011.

[14] A. Lamkanfi and S. Demeyer. Predicting reassignments of bug reports-an exploratory investigation. In *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, pages 327–330. IEEE, 2013.

[15] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals. Predicting the severity of a reported bug. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, pages 1–10. IEEE, 2010.

[16] S. Lu, S. Park, E. Seo, and Y. Zhou. Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. In *ACM Sigplan Notices*, volume 43, pages 329–339. ACM, 2008.

[17] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 18(1):50–60, 1947.

[18] T. Menzies and A. Marcus. Automated severity assessment of software defect reports. In *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*, pages 346–355. IEEE, 2008.

[19] E. Shihab, A. Ihara, Y. Kamei, W. M. Ibrahim, M. Ohira, B. Adams, A. E. Hassan, and K.-i. Matsumoto. Predicting re-opened bugs: A case study on the eclipse project. In *Reverse Engineering (WCRE), 2010 17th Working Conference on*, pages 249–258. IEEE, 2010.

[20] E. Shihab, A. Ihara, Y. Kamei, W. M. Ibrahim, M. Ohira, B. Adams, A. E. Hassan, and K.-i. Matsumoto. Studying re-opened bugs in open source software. *Empirical Software Engineering*, pages 1–38, 2012.

[21] K. Somasundaram and G. C. Murphy. Automatic categorization of bug reports using latent dirichlet allocation. In *Proceedings of the 5th India Software Engineering Conference*, pages 125–130. ACM, 2012.

[22] C. Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101, 1904.

[23] A. Sureka. Learning to classify bug reports into components. In *Objects, Models, Components, Patterns*, pages 288–303. Springer, 2012.

[24] A. Tamrawi, T. T. Nguyen, J. M. Al-Kofahi, and T. N. Nguyen. Fuzzy set and cache-based approach for bug triaging. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 365–375. ACM, 2011.

[25] F. Thung, S. Wang, D. Lo, and L. Jiang. An empirical study of bugs in machine learning systems. In *Software Reliability Engineering (ISSRE), 2012 IEEE 23rd International Symposium on*, pages 271–280. IEEE, 2012.

[26] Y. Tian, D. Lo, and C. Sun. Information retrieval based nearest neighbor classification for fine-grained bug severity prediction. In *Reverse Engineering (WCRE), 2012 19th Working Conference on*, pages 215–224. IEEE, 2012.

[27] Y. Tian, D. Lo, and C. Sun. Drone: Predicting priority of reported bugs by multi-factor analysis. *International Conference on Software Maintenance*, pages 200–209, 2013.

[28] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[29] F. Zhang, F. Khomh, Y. Zou, and A. E. Hassan. An empirical study on factors impacting bug fixing time. In *Reverse Engineering (WCRE), 2012 19th Working Conference on*, pages 225–234. IEEE, 2012.

[30] H. Zhang, L. Gong, and S. Versteeg. Predicting bug-fixing time: an empirical study of commercial software projects. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 1042–1051. IEEE Press, 2013.

[31] M.-L. Zhang and Z.-H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[32] T. Zimmermann, N. Nagappan, P. J. Guo, and B. Murphy. Characterizing and predicting which bugs get reopened. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 1074–1083. IEEE, 2012.

[33] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss. What makes a good bug report? *Software Engineering, IEEE Transactions on*, 36(5):618–643, 2010.

[34] T. Zimmermann, R. Premraj, J. Sillito, and S. Breu. Improving bug tracking systems. In *Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on*, pages 247–250. IEEE, 2009.