2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)

Are All Duplicates Value-Neutral? An Empirical Analysis of Duplicate Issue Reports

Mingyang Li *‡ Lin Shi*†‡, Qing Wang*†‡[1]

* *Laboratory for Internet Software Technologies, Institute of Software Chinese Academy of Sciences, Beijing, China*

† *State Key Laboratory of Computer Sciences, Institute of Software Chinese Academy of Sciences, Beijing, China*

‡ *University of Chinese Academy of Sciences, Beijing, China*

mingyang@nfs.iscas.ac.cn, {shilin, wq}@itechs.iscas.ac.cn

I. ABSTRACT

Abstract—**In open source communities, there are numerous duplicate issue reports, considered as useless and negligible by developers. Conversely, some researches argued that duplicates deliver complementary information that could benefit issue-resolving. Considering all duplicates as value-neutral will result in either overestimation or underestimation of valuable information. It is necessary to be aware of whether all duplicates are redundant or beneficial.**

In this paper, we investigate whether duplicates have the same impacts on issue resolving and identification cost. We divide duplicates into three categories according to the statuses of master reports when duplicates are submitted. The results show duplicates in different categories play different roles in issue-resolving, and identification cost is also significantly different.

Our study reveals duplicates are different, but almost are paid equal attentions. It is promising to propose new approaches and tools to resolve the problem.

***Keywords*-Duplicate Issue Report; Master Report; Identification Cost;**

II. INTRODUCTION

In open source communities, the Issue Tracking Systems (ITSs), such as Bugzilla and Jira, act as important channels to collect and track issues from a large number of globally distributed contributors and users [1]. As issue reports can describe defects, expectations, feature requests and other issues, many software projects rely on ITSs to manage maintenance activities.

Unfortunately, there are a large number of issue reports describing the same software issues more than once in ITSs due to various reasons such as ignorance of checking similar reports before submitting and inexperience. Typically, if duplicates are identified, they will be appended to the corresponding master reports, and then the developers will no longer pay attentions to these duplicates. However, some researches, such as Bettenburg et al. [2], pointed out that duplicates may provide complementary information for issues. They gave the conclusion that the complementary information can improve automated triaging techniques such as deciding who should fix a bug. In order to better resolve the problem, it is necessary to differentiate duplicates, and gain further insights into the value neutrality of duplicates: which types are beneficial and which are useless. More specifically, can all duplicates contribute something to software maintenance? Are all duplicates value-neutral and supposed to be paid equal attentions?

In this paper, we conduct an empirical study to investigate whether duplicates submitted at different time are value-neutral from two aspects: contributions to issue-resolving and identification cost. According to the statuses of master reports when duplicates are submitted, we divide duplicates into three categories: Issue-Unresolved, Issue-Fixed and Issue-Rejected duplicates. We investigate whether the duplicates in three categories play different roles in issue-resolving. We further analyze the cost spent on identifying duplicates in the three categories from three aspects: the number of developers involved in discussions, the number of comments posted under duplicates, and the length of descriptions.

After investigating 452,707 issue reports and their corresponding change histories from seven popular open source projects, we find that taking time dimension into consideration can differentiate duplicate issue reports obviously. The duplicates in different categories have different contributions to issue-resolving. We verify the conclusion that duplicates do provide complementary information. However, the evidences also show that the complementary information in some duplicates may not make sense to issue-resolving. In addition, our study also shows that Issue-Unresolved, Issue-Fixed and Issue-Rejected duplicates differ significantly in terms of the number of involved developers and comments.

In summary, the key contributions of this paper include:

- Taking time dimension into consideration, it is meaningful to differentiate duplicate issue reports. We analyze duplicate issue reports according to the statuses of master reports when duplicates are submitted, which provides comprehensive understanding towards duplicate issue reports.
- We investigate three categories of duplicates respectively and provide evidences that not all complementary information in duplicate issue reports contribute to issue-resolving.
- We quantitatively characterize the cost spent on identifying three duplicate categories, and conduct experi-

[1]The corresponding author

978-1-7281-3927-2/19/$31.00 ©2019 IEEE
DOI 10.1109/QRS.2019.00043

ments to verify the differences among the three categories.
- Our study also prompts practical insights such as automatic tools can help reporters avoid submitting useless duplicate issue reports. Tools shipped with multi-document summarization and alarms that can remind developers to reconsider the rejected issues, are promising.

The rest of this paper is organized as follows: Section III describes details about issue reports in open source software communities. Section IV proposes research questions. Section V and VI show the experimental setup and results respectively. Section VII provides detailed discussions of the lessons learned and threats to validity. Section IX surveys related work. Finally, we summarize this paper in Section X.

III. Issue Reports

Issue reports have been an important information carrier in issue tracking systems. As issue reports can describe defects, expectations, feature requests and other issues, many software projects rely on ITSs to manage maintenance activities.

A. The Life Cycle of an Issue Report

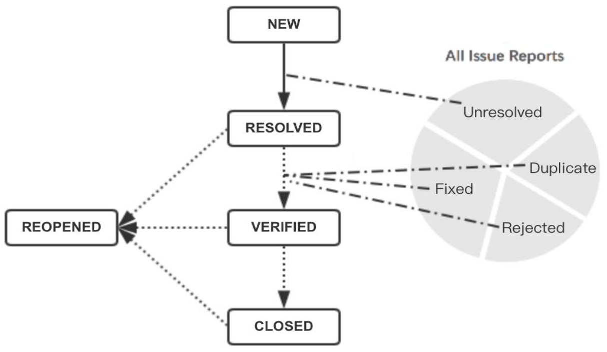

Figure 1. The Life Cycle of an Issue Report

The life cycle of an issue report is illustrated in Figure 1. When a reporter submits an issue report, the status of the report is labeled as "NEW". After that, a developer is assigned to resolve the issue. The status of the report will change to "RESOLVED" when the issue is resolved. At the same time, a resolution value (FIXED, DUPLICATE, INVALID, WONTFIX, WORKSFORME and so on) is given to the issue report in order to mark the resolution results. Next, the resolved issue report will be verified. If the issue is considered to be successfully resolved, the issue report will be closed. "RESOLVED", "VERIFIED" and "CLOSED" reports may be "REOPENED" if the issue is found not resolved correctly later.

B. Categories of Issue Reports

In order to shield the differences of resolution values among projects, we generalize all the issue reports into four categories manually: Unresolved, Fixed, Rejected and Duplicate, according to the resolution values. Specifically, four categories of issue reports are introduced as follows.

- **Unresolved**: new-coming issue reports before labeled as "RESOLVED", which indicates that the corresponding issues have not been resolved by developers.
- **Fixed**: issue reports are resolved and labeled as "FIXED", which indicates that the corresponding issues have already been fixed by developers.
- **Rejected**: issue reports labeled as "INVALID", "WORKSFORME" and so on, which indicates that developers will not fix the issues.
- **Duplicates**: issue reports labeled as "DUPLICATE", which indicates that they describe the issues reported before.

C. Master Issue Reports and Duplicate Issue Reports

Duplicate issue reports are those describing the issues which have been reported previously. If multiple reports describe the same issue, the report firstly addressed by developers is master report and others are duplicates. Duplicates correspond to "Duplicate" issue reports defined in Section III-A. And master reports belong to one of three categories ("Unresolved", "Fixed" or "Rejected") defined in Section III-A.

D. Categories of Duplicate Issue Reports

Based on the resolution values of master reports when duplicates are submitted, we divide duplicates issue reports into three categories: Issue-Unresolved, Issue-Fixed and Issue-Rejected duplicates. Several examples are presented in the following to illustrate three categories of duplicates.

*1) **Issue-Unresolved Duplicate: Duplicates submitted when master reports are not resolved.**:* Master report ISSUE #63614[1] was reported on 2006-03-25. Then, it was fixed on 2015-01-09[2]. One of its duplicate ISSUE #125953[3] was reported on 2014-12-17 at the time when the master report had not been resolved yet.

*2) **Issue-Fixed Duplicate: Duplicates submitted when master reports are fixed.**:* Master report ISSUE #127143[4] was reported on 2016-09-27. Then, it was fixed on 2016-11-07 [5] without being reopened later. The duplicate ISSUE #127209[6] was reported on 2016-11-09 at the time when the issue had already been fixed.

[1]https://bz.apache.org/ooo/show_bug.cgi?id=63614
[2]https://bz.apache.org/ooo/show_activity.cgi?id=63614
[3]https://bz.apache.org/ooo/show_bug.cgi?id=125953
[4]https://bz.apache.org/ooo/show_bug.cgi?id=127143
[5]https://bz.apache.org/ooo/show_activity.cgi?id=127143
[6]https://bz.apache.org/ooo/show_bug.cgi?id=127209

*3) **Issue-Rejected Duplicate: Duplicates submitted when master reports are rejected.**:* Master report ISSUE #120898[7] was reported on 2012-12-14. It described an issue "Launching fails with error message Unidentified Developer". The issue was rejected by development teams on 2012-12-18[8] and labeled as "INVALID" because developers did not think it is a bug (see Comment 2,3). Then, the issue was verified and closed on 2012-12-21. However, duplicate ISSUE #122830, #123548, #127198 and other 14 duplicates were submitted to report the same issue. At last, the master report was reopened on 2016-10-28 because developers thought it is a valid request for enhancement (see Comment 19 in #120898). ISSUE #122830, #123548, #127198 and other 14 duplicates are submitted when master report (#120898) was rejected. Developers and reporters had different attitudes towards the issue at first. And the issue was reported for several times by different reporters, developers reconsidered the master report they rejected at first.

Here, we define three categories of duplicates.

- Issue-Unresolved duplicates: duplicates submitted when corresponding master reports are not resolved yet.
- Issue-Fixed duplicates: duplicates submitted after corresponding master reports are fixed.
- Issue-Rejected duplicates: duplicates submitted after corresponding master reports are rejected by development teams.

IV. Research Questions

In order to better understand duplicates, we propose four research questions to get further insights into them.

RQ1: What is the proportion of the three categories of duplicates? In RQ1,we give an brief on the proportion of the categories of duplicates to understand which one accounts for the most.

RQ2: Are all the three categories of duplicates beneficial to issue-resolving?

- **RQ2.1: Do all the three categories of duplicates provide complementary information?**
- **RQ2.2: Is all the complementary information in the three categories beneficial to issue-resolving?**

Taking time dimension into consideration, we further investigate whether three categories can give complementary information, and whether all the complementary in three categories can contribute to issue-resolving.

RQ3: Is the cost involved in identifying the duplicates significantly different among the three categories? We measure the cost of identifying duplicates from three aspects, i.e., the number of developers who are involved in the discussions, the number of comments posted under issue reports, and the length of descriptions.

[7]https://bz.apache.org/ooo/show_bug.cgi?id=121478
[8]https://bz.apache.org/ooo/show_activity.cgi?id=121478

V. Experiment

In this section, we introduce the studied subjects and data collection, as well as how we design the experiments to investigate duplicates.

A. Subjects

We build a dataset by collecting issue reports from Bugzilla. There are seven projects selected as studied subjects: Eclipse-Platform, Mozilla-Core, Mozilla-Firefox, KDE, Open Office, LibreOffice and Linux-Kernel. We choose the seven projects due to that they are popular open source software projects and attract lots of users submitting issue reports every day. In total, there are 452,707 issue reports and corresponding change histories collected in our dataset. The details about issue reports in seven projects are shown in Table I.

Table I
Details of Studied Subjects

Project Name	Total Reports	Unresolved	Fixed	Rejected	Duplicate
Eclipse-Platform	37027	28.6%	41.1%	15.4%	14.9%
Mozilla-Core	162848	19.1%	36%	27.7%	17.2%
Mozilla-Firefox	109788	14.8%	39.6%	22.8%	22.8%
KDE	60231	21.9%	30.3%	23%	24.8%
Open Office	17286	35.4%	21%	26.9%	16.7%
LibreOffice	39089	22.6%	23.8%	34.7%	18.9%
Linux-Kernel	26438	16.7%	28.3%	46.4%	8.6%

B. Experiment Design

For RQ1, we investigate the change history of each master report, and find out its resolution value at the time when duplicates are submitted. If corresponding master report belongs to "Unresolved" when duplicates are submitted, appended duplicates are Issue-Unresolved duplicates. If corresponding master report belongs to "Fixed" when duplicates are submitted, appended duplicates are Issue-Fixed duplicates. If corresponding master report belongs to "Rejected" reports when duplicates are submitted, duplicates are Issue-Rejected. At the same time, the numbers of three categories are recorded.

For RQ2, we quantify and compare the predefined information items for master reports with the ones for duplicates. The unique values in master reports and master reports merged with corresponding duplicates are count respectively. In addition, we conduct hypothesis test to check whether the changes are significantly different before and after merging. Giving the fact that some duplicates may be submitted due to that the issue are not perfectly resolved, we study the change histories of master reports in order to figure out under what circumstances are duplicates submitted.

For RQ3, we use the following three metrics to measure cost spent on identifying the three categories. For each metric, hypothesis test is applied to verify whether the differences between every two groups are significant.

- **M1: The Number of Distinct Developers Involved.** We use the number of distinct developers other than reporters involved in discussions before identification as M1. It is used to measure the resources that are involved in identifying the duplicates. The more distinct developers are involved, the more cost is spent. First, we study the change histories to find out the time when duplicates are identified. And then, we extract all comments before identification time. As last, we count distinct authors of these comments except reporters.
- **M2: The Number of Comments Posted under Issue Reports.** We use the number of comments made in discussions before identification as M2. It is used to measure the direct cost which is spent on identifying the duplicates. The more comments are made, the more cost is spent. For this metric, like M1, we find out identification time and get all comments before the time. We count comments after filtering.
- **M3: The Length of Descriptions.** We use the number of words written in the issue reports' field "descriptions" as M3. It is used to measure the cost which is spent on writing and understanding the duplicates. We consider the longer is the description, the more cost is spent. For this metric, we parse issue reports and get the descriptions. Using methods provided by [3], we filter out the traces, patches and screen shots in descriptions and count the words remaining.

VI. Results and Analysis

This section gives the analysis of results in order to answer four research questions.

A. RQ1: What is the proportion of the three categories of duplicates?

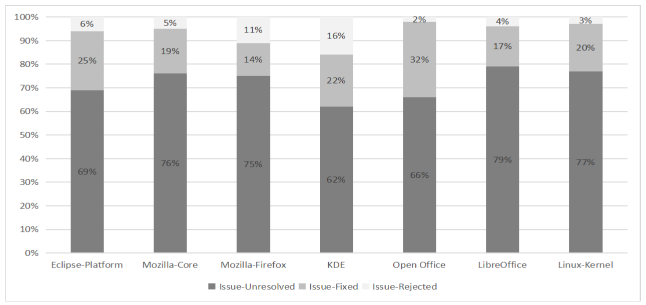

Figure 2. The Percentages of Three Categories

Figure 2 shows the percentages of the three categories in the seven software projects. In all projects, Issue-Unresolved duplicates account for most (62-79%) of the duplicates. Given the fact that an Issue-Unresolved duplicate is submitted when the master report is not resolved, the corresponding issue may still exist in the project at that time. Users who encounter the same issue will report it again. In such cases, duplicates are created mainly due to that reporters fail to retrieve the similar issue reports which have been submitted previously.

Issue-Fixed duplicates (14-32%) account for the second most of duplicates. These duplicates are still submitted even though corresponding master reports are already fixed. Issue-Fixed duplicates are created mainly due to two possible reasons: (1) the usage of out-of-date versions; (2) the corresponding issues are not perfectly fixed. Issue-Fixed duplicates related to the first reason are likely to be redundant, while those related to the second reason may benefit the software maintenance activities. Therefore, if the usage of the out-of-date versions can be detected and prompted to issue reporters, the size of redundant duplicates can be trimmed.

There are also 2-11% duplicates are Issue-Rejected duplicates, which indicates that although some issues are rejected by developers, these duplicates are still submitted to claim. This may be due to differences of opinions between reporters and developers. Issues rejected by developers may be concerned by users. If a rejected issue is resubmitted by multiple users or multiple times, re-considerations on rejecting the reports need to be launched.

Summary of RQ1: In all projects, the most duplicates are submitted when master reports are not resolved (Issue-Unresolved), then Issue-Fixed and Issue-Rejected. Duplicates submitted at different time may have different contributions on issue-resolving, and it is meaningful to distinguish them.

B. RQ2: Are all the three categories of duplicates beneficial to issue-resolving?

RQ2.1: Do all the three categories of duplicates provide complementary information?

The Table II shows the average amount of information added by duplicates. It consists of the following columns:

- The column "Information Item" presents all information items predefined by Bugzilla.
- The following three columns "Issue-Unresolved", "Issue-Fixed" and "Issue-Rejected" represent the results of the three categories respectively.
- The column "Master" lists the average number of each information item in the original master reports.
- The column "Extended" lists the average number of unique values in the duplicates merged with corresponding master reports.
- The column "Change" lists the differences between "Extended" and "Mastet" and represents the average number of information items that issue duplicates would add per master report. One-sided paired t-tests are conducted. The symbol "*" is used as suffix if the difference is significant (the change is significant at p $<.001$).

Table II
AVERAGE AMOUNT OF INFORMATION ADDED BY DUPLICATES PER MASTER REPORT

Information Item	**Issue-Unresolved**			**Issue-Fixed**			**Issue-Rejected**		
	Master	Extended	Change	Master	Extended	Change	Master	Extended	Change
project	1	1.13	+0.13*	1	1.08	+0.08*	1	1.27	+0.27*
component	1	1.34	+0.34*	1	1.27	+0.27*	1	1.35	+0.35*
operating system	0.84	0.96	+0.12*	0.74	0.90	+0.16*	0.83	1.12	+0.29*
reported platform	0.51	0.65	+0.14*	0.31	0.41	+0.10*	0.39	0.52	+0.13*
version	0.96	1.58	+0.62*	0.81	1.47	+0.66*	0.93	1.64	+0.71*
reporter	1	3.14	+2.14*	1	2.31	+1.31*	1	5.45	+4.45*
priority	1	1.09	+0.09	1	1.02	+0.02	1	1.03	+0.03

Significantly, three categories of duplicates provide more information about project, component, operating system, reported platform, version and reporters compared with the master report. However, all the three categories of duplicates do not provide significantly more complementary information about priority. It may due to that reporters tend to use the default priority provided by Bugzilla if the issue is not particularly urgent.

Particularly, in the row "reporter", change in Issue-Rejected (+4.45) is much larger than those in Issue-Unresolved (+2.14) and Issue-Fixed (+1.31), which indicates that compared to the other two categories, the number of reporters submitting duplicates to claim the rejected issues is larger. This phenomenon may relate with the inappropriate rejection of the master reports. To deliver better user experiences, it is suggested to reconsider the rejected issues followed by Issue-Rejected duplicates.

RQ2.2: Is all the complementary information in the three categories beneficial to issue-resolving? After known that all the three categories of duplicates can provide complementary information, we further investigate if all those complementary information contribute to issue-resolving.

Issue-Fixed and Issue-Rejected duplicates are both submitted after the master reports are resolved. Considering time dimension, both may not contribute to issue-resolving. Some may argue that Issue-Fixed and Issue-Rejected may be submitted if corresponding issues are not resolved perfectly. on the fact that issue reports will be reopened if the issues are not resolved perfectly, we further investigate if master reports of Issue-Fixed and Issue-Rejected duplicates are reopened after corresponding duplicates are submitted.

Table III shows the results of how many master reports are reopened after their corresponding Issue-Fixed and Issue-Rejected duplicates are submitted.

For Issue-Fixed duplicates, less than 1% master reports are reopened in the later 16 months after being fixed on average, which indicates that almost all these issues have already been fixed correctly. It is a potential evidence that complementary information in Issue-Fixed duplicates may not contribute to issue-resolving. For Issue-Rejected duplicates, 17.18% master reports are reopened in the later 16 months after being rejected on average. The results indicate that Issue-Rejected duplicates may motivate developers to reconsider the issues they rejected at first. Without these duplicates, the rejected master reports may remain closed, and the corresponding issues will not be resolved.

Table III
REOPENED RATE OF MASTER REPORTS IN ISSUE-FIXED AND ISSUE-REJECTED DUPLICATES

Project Name	**Issue-Fixed**	**Issue-Rejected**
Eclipse-Platform	0.17%	23.73%
Mozilla-Core	0.21%	23.89%
Mozilla-Firefox	1.23%	13.31%
KDE	0.78%	19.67%
Open Office	0.55%	18.70%
LibreOffice	0.54%	13.56%
Linux-Kernel	0.21%	7.41%
Average	0.53%	17.18%

Summary of RQ2: Within all the three categories, duplicates can provide complementary information which does not exist in master reports. However, the evidences also show that not all the complementary information can contribute to issue-resolving, which indicates that duplicates are not value-neutral. Specifically, the complementary information provided by Issue-Unresolved duplicates may contribute to issue-resolving. Almost all Issue-Fixed duplicates are submitted when corresponding issues are resolved perfectly, and the complementary information in Issue-Fixed may not make sense to issue-resolving. Issue-Rejected duplicates are valuable which may motivate developers to reconsider the rejected issues.

C. RQ3: Is the cost involved in identifying the duplicates significantly different among the three categories?

We quantify the identification cost and give analysis of results in the following.

The Number of Distinct Developers Involved (M1).

Table IV shows the means of distinct developers involved in discussing duplicates. The column "p-value" represents the p-value of t-test between every two categories which give the mean values in each row.

Generally, Issue-Unresolved duplicates involve significantly more developers than duplicates submitted after master issues are resolved (Issue-Fixed and Issue-Rejected), and

Table IV
THE MEANS OF DEVELOPERS INVOLVED IN DISCUSSIONS

Project	I-Unresolved	I-Fixed	I-Rejected	p-value
Eclipse-Platform	1.94	1.72	-	<0.001*
	1.94	-	1.80	<0.001*
	-	1.72	1.80	0.179
Mozilla-Core	2.47	1.96	-	<0.001*
	2.47	-	2.01	<0.001*
	-	1.96	2.01	0.234
Mozilla-Firefox	1.77	1.52	-	<0.001*
	1.77	-	1.59	<0.001*
	-	1.52	1.59	0.152
KDE	1.99	1.71	-	<0.001*
	1.99	-	1.79	<0.001*
	-	1.71	1.79	0.210
Open Office	2.55	2.47	-	0.223
	2.55	-	2.50	0.147
	-	2.47	2.50	0.243
LibreOffice	2.41	1.85	-	<0.001*
	2.41	-	1.97	<0.001*
	-	1.85	1.97	0.078
Linux-Kernel	2.37	1.95	-	<0.001*
	2.37	-	2.00	<0.001*
	-	1.95	2.00	0.097

the differences between Issue-Fixed and Issue-Rejected are not significant for all projects. The results indicate that deciding duplicates to unresolved issues is more difficult than the resolved issues. As the issues are already resolved, the developers are much more familiar with those issues. Thus when duplicates are reported, they can identify them in a short time. Improving the clarity and readability of unresolved issue reports may help developers identify duplicates faster [4].

The Number of Comments Posted under Duplicates (M2).

Table V
THE MEANS OF COMMENTS

Project	I-Unresolved	I-Fixed	I-Rejected	p-value
Eclipse-Platform	1.49	1.92	-	<0.001*
	1.49	-	1.99	<0.001*
	-	1.92	1.99	0.279
Mozilla-Core	3.70	2.35	-	<0.001*
	3.70	-	2.58	<0.001*
	-	2.35	2.58	0.050
Mozilla-Firefox	1.98	1.19	-	<0.001*
	1.98	-	1.38	<0.001*
	-	1.19	1.38	0.013
KDE	1.13	0.51	-	<0.001*
	1.13	-	0.84	<0.001*
	-	0.51	0.84	<0.001*
Open Office	3.86	3.61	-	0.013
	3.86	-	3.77	0.111
	-	3.61	3.77	0.063
LibreOffice	3.22	2.13	-	<0.001*
	3.22	-	2.45	<0.001*
	-	2.13	2.45	0.014
Linux-Kernel	5.97	3.19	-	<0.001*
	5.97	-	3.97	<0.001*
	-	3.19	3.97	<0.001*

Table V shows the means of comments posted under issue reports and the results of t-test. The column "p-value" represents the p-value of t-test between every two categories which give the mean values in each row.

We can find that the most comments are made in Issue-Unresolved category, and the results of t-test show that it is significantly different from the other two categories except for the Open Office project. The results of M2 provide the same evidences with M1 that if issues are not resolved, more discussions are made to identify the duplicates. On the contrary, if corresponding issues are resolved, developers could identify the duplicates confidently without lots of discussions. Particularly, in KDE and Linux-Kernel, significantly more comments are made in Issue-Rejected than Issue-Fixed, which indicates that developers are more cautious to reject issues in the two projects.

The Length of Descriptions (M3).

Table VI
THE MEANS OF WORDS IN DESCRIPTIONS

Project	I-Unresolved	I-Fixed	I-Rejected
Eclipse-Platform	120.21	151.75	143.47
Mozilla-Core	116.49	129.86	118.16
Mozilla-Firefox	114.28	124.77	118.26
KDE	664.85	695.62	670.84
Open Office	97.87	106.87	99.63
LibreOffice	100.01	108.64	100.7
Linux-Kernel	330.33	320.45	328.87

Table VI shows the means of words written in descriptions in the three categories of duplicates. We find that there are no obvious differences among the three categories. Compared to other five projects, much more words are written in the descriptions of reports in KDE and Linux-Kernel, which indicates much more words are used to describe issues in the two projects. It may relate to the internal characters of the two projects.

Summary of RQ3: In most of the projects, there are significant differences between Issue-Unresolved and Issue-Fixed or Issue-Rejected duplicates in terms of the number of involved developers and comments. This finding indicates that more resources and cost are spent on identifying duplicates when the corresponding issues are not resolved. There are no significant differences among duplicates in identification cost regardless corresponding master reports are fixed or rejected. Automatic tools should pay more attentions to the Issue-Unresolved duplicates which are potentially difficult for developers to identify.

VII. DISCUSSION

Issue-Unresolved duplicates. Issue-Unresolved is the category with the most duplicates. It can provide complementary information to help issue-resolving. The cost to identify the duplicates of this category is significantly larger than the other two categories, which indicates that if an issue is not resolved, it is difficult for developers to identify corresponding duplicates. Given that more efforts are spent on identifying Issue-Unresolved duplicates, automatic duplicate retrieval tools are better to pay more attentions to Issue-Unresolved duplicates. Successfully identifying those duplicates can largely alleviate the burden of duplicates in ITSs. In addition, the information provided by Issue-Unresolved duplicates are useful for issue-resolving. These information is written in natural language. Natural Language Processing (NLP) techniques, such as automatic summarization from multi-documents can be applied to combine the information in duplicates and master issues in order to provide a synthesis and completed understanding of the issues.

Issue-Fixed duplicates. On average, 21% duplicate issue reports are submitted even after the corresponding issues have already been fixed. It may result from that the versions used by reporters are out of date, thus the associated issue reoccur. Our study finds that the Issue-Fixed duplicates result in few reopening of the issue, which indicates that they may not make contributions to issue-resolving. Furthermore, duplicate retrieval tools could also consider not only issues that have been reported before, but also the release notes to check if reported issues have already been resolved in the latest versions. In addition, automatic tools for Issue-Fixed identification can reduce the number of this kind of duplicates, and prevent developers from spending cost on discussing and identifying these duplicates.

Issue-Rejected duplicates. Our study finds that the reopening rate of Issue-Rejected duplicates is much higher than Issue-Fixed duplicates, which indicates that developers are supposed to be more cautious about rejections of issues annoying lots of users. There are multiple reasons for rejection of master reports, such as lack of key information, different opinions of developers and users, and so on. Developers could handle Issue-Rejected duplicates in different ways, depending on the reasons for rejection. For example, if the master reports are rejected due to lack of key information to reproduce the issues, approaches or automatic tools to understand and summary the information in master and duplicate reports are useful as well. Furthermore, if a rejected issue is reported for many times, it should be paid attentions. The more times the issue is reported, the higher priority the issue is considered with. Unlike Issue-Fixed duplicates, Issue-Rejected duplicates may be beneficial for issue-fixing.

VIII. Threats to Validity

In this section, we discuss the threats to the validity of our conclusions.

A. External Threats

Our study is conducted on seven projects. Different projects are owned by different organizations and follow different processes for managing their issues. There are also various statuses and resolution values in different projects, which may result in that our work may not generalize to the other projects. We try to shield these differences by generalization. Specially, we check documentations in every project and merge many resolution values into four categories (Unresolved, Fixed, Rejected and Duplicate).

B. Internal Validity

There is a assumption that activities of master reports can indicate if issues are resolved in our study. Some developers may not update statuses of master reports even if the issues are resolved, which results in more duplicates are divided into Issue-Unresolved duplicates. However, if an issue has already been resolved, the number of duplicates which report same issue will be small, and thus the influence is small.

In RQ1, we use the time when duplicated are submitted to divide duplicates. However, the time recorded by ITSs is not the time when reporters start to write reports. Reporters maybe spend more time searching for master reports and preparing to write reports. We do not think this process will last too long. PAMS Neto et al. [5] reported reporters spend in average 12.5 minutes searching for duplicates. We think that the time recorded by ITSs will not have much influence on the final results except some extreme cases.

In RQ2, we think that if an issue report is fixed and not reopened in the next 16 months, it is fixed correctly. The threshold (16 months) may introduce errors to our study because we can not make sure all fixed reports will not be reopened after the time when our dataset is build. We try to reduce the errors by investigating the intervals between the time when reports are labeled as 'RESOLVED' and 'REOPENED'. The maximum time interval in our data is about 16 months.

In RQ3, in order to measure identification cost of duplicate issue reports, we use three metrics. Each of three metrics could only measure the identification cost from a single aspect, and can not be used as an accurate indicators of identification cost.

IX. Related Work

Duplicate issue reports are very popular in issue tracking systems and previous studies showed that lots of efforts are spent on duplicate issue reports. Anvik et al. [6] reported 20-30% of the issue reports in Eclipse and Firefox respectively are duplicates and described the duplicate bug and bug triage problems. Cavalcanti et al. [7] pointed out many projects had duplicate bug reports and a considerable amount of time was wasted by duplicates. In addition, the efforts spent on duplicates were quantified in previous studies. Rakha et al. [8] quantified the efforts spent on duplicates in terms of that the median identification delay, involved developers, comments etc. Davidson et al. and Cavalcanti et al. [9], [10] also examined the effort that is associated with closing a duplicate report. There are also many studies to analysis the information items in issue reports. Bettenburg et al. [2] reported that duplicate bug reports provide developers with information that was not present in the original report and the additional information provided by duplicates helps to resolve bugs quicker. Bettenburg et al. [3] reported that steps to reproduce and stack traces are most useful in bug reports. Chaparro et al. [11] revealed that the difference between the lexical agreement of duplicate and non-duplicate pairs is a good predictor for the performance of TR-based duplicate detection. Mohamed et al. [12] studied Just-In-Time duplicate retrieval feature provided by Bugzilla and reported that less duplicate reports end up in an ITS and duplicates are more difficult to retrieval after the activation of the Just-In-Time duplicate retrieval feature. In previous studies, duplicate issue reports are treated in the same way. Xia et al. [13] reported 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.

In practice, we find that duplicates differ in not only values, but also needed efforts. Our study could complement the existing issue report studies.

X. Conclusion

Duplicate issue reports are submitted at different time. Taking time dimension into consideration, different duplicates play different roles in issue-resolving and vary in identification cost, which indicates that not all duplicates value-neutral.

Duplicates submitted when master reports have not been resolved (Issue-Unresolved) give complementary information and may contribute to issue-resolving with the complementary information. However, they also involve significantly more developers and provoke significantly more comments.

There are also many duplicates submitted when master reports have already been resolved (Issue-Fixed and Issue-Rejected). Although both of them can also give complementary information, the complementary information may not contribute to issue-resolving. Further, they potentially cause significantly less identification cost than the duplicates submitted when master reports are not resolved.

Our study gives a new perspective that not all duplicate issue reports are not value-neutral. Different duplicates play different roles in issue-resolving and potentially need different cost to utilize. It is beneficial for automatic tools to take advantages of the insights of duplicates in order to better solve the problem.

Acknowledgment

This work is supported by the National Key Research and Development Program of China under grant No.2018YFB1403400, National Natural Science Foundation of China under grant No.61602450, No.6143201, No.61802374.

References

[1] D. Bertram, A. Voida, S. Greenberg, and R. Walker, "Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams," in *ACM Conference on Computer Supported Cooperative Work*, 2010, pp. 291–300.

[2] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Duplicate bug reports considered harmful ... really?" pp. 337–345, 2008.

[3] ——, "Extracting structural information from bug reports," in *International Working Conference on Mining Software Repositories*, 2008, pp. 27–30.

[4] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?" *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 618–643, 2010.

[5] E. A. S. M. PAMS Neto, T Vale, "The bug report duplication problem: an exploratory study," in *Software Quality Journal*, 2013, pp. 39–66.

[6] J. Anvik, L. Hiew, and G. C. Murphy, "Coping with an open bug repository," in *OOPSLA Workshop on Eclipse Technology Exchange, Etx 2005, San Diego, California, Usa, October*, 2005, pp. 35–39.

[7] Y. C. Cavalcanti, D. Lucrdio, T. Vale, E. S. D. Almeida, and S. R. D. L. Meira, "The bug report duplication problem: an exploratory study," *Software Quality Journal*, vol. 21, no. 1, pp. 39–66, 2013.

[8] M. S. Rakha, W. Shang, and A. E. Hassan, "Studying the needed effort for identifying duplicate issues," *Empirical Software Engineering*, vol. 21, no. 5, pp. 1960–1989, 2016.

[9] J. L. Davidson, N. Mohan, and C. Jensen, "Coping with duplicate bug reports in free/open source software projects," in *Visual Languages and Human-Centric Computing*, 2011, pp. 101–108.

[10] Y. C. Cavalcanti and D. Lucredio, "One step more to understand the bug report duplication problem," in *Software Engineering*, 2010, pp. 148–157.

[11] O. Chaparro, J. M. Florez, and A. Marcus, "On the vocabulary agreement in software issue descriptions," in *IEEE International Conference on Software Maintenance and Evolution*, 2017, pp. 448–452.

[12] M. S. Rakha, C. P. Bezemer, and A. E. Hassan, "Revisiting the performance evaluation of automated approaches for the retrieval of duplicate issue reports," *IEEE Transactions on Software Engineering*, vol. PP, no. 99, pp. 1–1, 2017.

[13] X. Xia, D. Lo, M. Wen, E. Shihab, and B. Zhou, *An empirical study of bug report field reassignment*, 2014.