2010 14th European Conference on Software Maintenance and Reengineering

An Initial Study on the Bug Report Duplication Problem

Yguaratã Cerqueira Cavalcanti[1,2], Eduardo Santana de Almeida[2,4],
Carlos Eduardo Albuquerque da Cunha[1,2], Daniel Lucrédio[2,3], Silvio Romero de Lemos Meira[1,2]

[1]Center for Informatics – Federal University of Pernambuco
[2]RiSE – Reuse in Software Engineering
[3]Institute of Mathematical and Computer Science - University of São Paulo
[4]Computer Science Department – Federal University of Bahia
{ycc, ceac, srlm}@cin.ufpe.br, esa@rise.com.br, lucredio@icmc.usp.br

Abstract

According to recent work, duplicate bug report entries in bug tracking systems impact negatively on software maintenance and evolution productivity due to, among other factors, the increased time spent on report analysis and validation, what in some cases take over 20 minutes. Therefore, a considerable amount of time is lost mainly with duplicate bug report analysis. This work presents an initial characterization study using data from bug trackers from private and open source projects, in order to understand the possible factors that cause bug report duplication and its impact on software development.

1 Introduction

With the objective of improving the software maintenance and evolution, some organizations use specific systems to manage, track and store bug reports. These systems are generally called *bug trackers* [1], and are useful because software changes can be identified and reported [2]. Moreover, the bug report historical databases can be used as documentation for the project's history.

Although bug trackers bring many benefits to software development, some problems may appear as a consequence of its usage, such as [3]: dynamic assignment of bug reports, change impact analysis and effort estimation, quality of bug report descriptions, software evolution and traceability, and duplicate bug reports detection. The focus of this work is on duplication issue.

The problem of duplication is characterized by the submission of two or more bug reports that describe the same change request. The main consequence of this problem is the overhead of rework when managing these bug reports. For example, the staff responsible for the search and analysis of bug reports must spend extra time with bug reports that have already been submitted.

In this context, this work presents an initial characterization study about bug repositories and the search and analysis of bug reports, in order to understand the possible factors that could cause bug report duplication and its impact on software development.

The remainder of this paper is structured as follows: Section 2 presents the definition of the study; Section 3 describes the projects and data used in the study. In Section 4, the operation is discussed, and in Section 5 the analysis and interpretation of the results are presented. Sections 6 and 7 present related work and conclusion, respectively.

2 Definition of the Study

The GQM method [4] was used to define this characterization study. The GQM consists of the definition of the study's goal, the questions to be answered, and the related metrics. For some metrics, we established a baseline based on related work, while others were explored in this work in order to serve as basis for future studies.

The goal of this study was to analyze *bug repositories* and the *activities for searching and analyzing bug reports* with the purpose of *understanding them* with respect to the *possible factors that could impact on the duplication problem and their consequences on software development*, from the point of view of the *researcher*, in the context of *software development projects*. The following questions and metrics were defined.

Question 1. *Do the analyzed projects have a considerable amount of duplicate bug reports?* This is the starting point of this study. Although the literature reports that there is a considerable amount of bug duplication in most projects, if the projects analyzed in this study do not have enough duplicate bug reports, the other questions can not be answered properly. The following metric was defined to help answering this question:

M_1: **percentage of duplicate bug reports in software development projects**. This metric can be collected by analyzing the status of the bug reports in the repository. For example, if the status of a bug report is defined by the keyword DUPLICATE, then it counts as a duplicate. Based on recent work [2, 5], we expect an average of 20% of duplicate bug reports present in bug repositories.

Question 2. *Is the submitters productivity being affected by the bug report duplication problem?* As mentioned before, duplicate bug reports have side effects, such as extra time for the analysis. In this context, the productivity is measured in terms of time that is needed to perform bug tracking activities, such as search and analysis of bug reports. We defined three metrics to understand this question:

M_2: **amount of time spent to search and analyze bug reports before opening a new bug report.** This time is measured in minutes and is counted from the moment a submitter began to investigate a bug report until it decides whether such problem is new or duplicate. Based on informal interviews with submitters from C.E.S.A.R.[1], we expect values between 10 to 20 minutes for this metric. However, we must note that such values can be biased, since it was not measured empirically;

M_3: **ratio between the average time to resolve duplicate bug reports and average time to resolve valid bug reports.** If valid bug reports take an average of x days to be resolved, then we wanted to know the percentage ($\%$) of x that is necessary to solve duplicate bug reports. Thus, to calculate M_3, we first divide it into two sub-metrics: (M_3') the average time (in days) to resolve duplicates; and (M_3'') the average time (in days) to resolve valid bug reports. Then we calculate $M_3 = M_3'/M_3''$, which is the ratio. We did not find any previous data to serve as a baseline for this metric;

M_4: **average frequency of bug reports per day.** It is the average amount of bug reports that are submitted per day to the projects' repositories. This measure is calculated dividing the amount of bug reports by the total of days that include them;

Question 3. *How are the relationships between master bug reports and duplicate bug reports characterized?* In this work, we consider three types of bug reports in a bug repository[6]: (a) *unique bug report (no duplicate reports associated)*; (b) *master bug report (the first entry of a bug reports group)*; and (c) *duplicate bug report*. It is important to understand how the groupings are characterized, because it can drive us when choosing adequate techniques to solve the bug report duplication problem.

M_6: ***grouping* types distribution.** We measured two types of grouping: (M_6') master bug reports that have only one duplicate bug report, also known as *one-to-one* relationships; and (M_6'') master bug reports that have more than one duplicate bug report, also known as *one-to-many* relationships. We could define more levels of relationships, however, we believe that these two are sufficient for our analysis.

Question 4. *Does the type of bug report influence the amount of duplicates?* There are mainly two types of bug reports: *enhancements* and *defects*. Intuitively, it may be argued that defects have more duplicates, because it is more likely that two users will notice the same defect, and it is less likely that the same enhancement is equally perceived by two different users. We wanted to analyze if this is true for the selected projects. If confirmed, this means that defect bug reports require a more careful analysis than enhancement reports, for example.

M_7: **Duplication ratio = duplicate bug reports / total bug reports.** For each bug report type we calculated the ratio between the duplicate and total bug reports. If this ratio is larger for defects, then this type of bug report causes more duplicates than enhancements.

Question 5. *Is the submitter profile an important factor that could impact on the bug report duplication problem?*

M_8: **Submitters' profile.** It is important to know what type of submitter profile is more susceptible to submitting duplicate bug reports. The values for this variable are: *sporadic* (**S**), *average* (**A**) and *frequent* (**F**). *Sporadic* is the reporter who submitted at most 10 bug reports in the analyzed period, *average* is the person who submitted between 10 and 30 bug reports in the period, and *frequent* is the person who submitted more than 30 bug reports in the period.

3 Projects and Data Selection

We chose projects from open source organizations and from a private organization (see Table 1). For open source projects, we chose eight (8) projects. For the private project, we chose bug reports from a project being developed at C.E.S.A.R. Regarding the open source projects, we collected all bug reports until the end of June/2008. For the private project, we collected all the bug reports from November/2006 to March/2008, which includes the entire life-cycle for this project.

Project	Domain	Code size	Staff size	Bugs	Life-time
Bugzilla	Bug tracker	55K	340	12829	14
Eclipse	IDE	6.5M	352	130095	7
Epiphany	Browser	100K	19	10683	6
Evolution	E-mail client	1M	156	72646	11
Firefox	Browser	80K	514	60233	9
GCC	Compiler	4.2M	285	35797	9
Thunderbird	E-mail client	310K	192	19204	8
Tomcat	Application server	200K	57	8293	8
Private Project	Mobile application	2M	21	7955	2

Table 1. Projects characteristics.

4 Study Execution

We defined a set of instrumentation assets to be used in the study, such as scripts, time-sheets, and a questionnaire. The scripts were used to read the bug reports from each

[1]Recife Center for Advanced Studies and Systems. `www.cesar.org.br`

project. The time-sheets were used to collect the time to open bug reports by the staff of the private project. A total of four people filled out the time-sheets during a period of two weeks. The time-sheets were applied only in this project because we had access to it and we could monitor this activity. For other projects, this information was gathered through a questionnaire, as described next.

The questionnaire was sent to the 20 most active submitters from each project. It was asked about time spent do search and analyze bug reports and if there is techniques being used to avoid duplicates. In total, the questionnaire was sent by e-mail to 180 submitters, with a two-weeks deadline for answer. From these emails, 24 could not be delivered to the recipients, and 141 did not respond the questionnaire in time, thus remaining only 17 answers.

The submitters had between 2 and 8 years of experience in the project in which they participated. Moreover, 14 submitters said they spent from 5 to 10 minutes performing searches for bug reports, and only 3 submitters chose the answer 10 to 15 minutes. For the private project, such time was collected using time-sheets and it was between 20 to 30 minutes.

5 Analysis and Interpretation

Question 1. The values for M_1 in Figure 1 show a high percentage of duplicates for 8 projects, the exception was the Tomcat project. Only 3 projects were below than the expected (Eclipse, Tomcat and GCC).

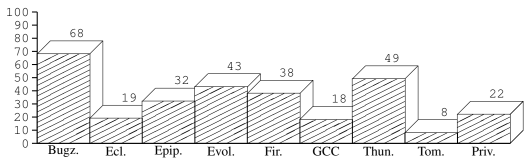

Figure 1. M_1**: percentage of duplication (%).**

For projects with major differences below the expected value (GCC and Tomcat), our intuition believe that they receive less duplicates because they have less end users. Generally, people who use GCC or Tomcat are developers with high skills on such projects.

Question 2. To answer this question, we analyzed metrics M_2, M_3 (Figure 2) and M_4 (Figure 3). For metric M_2, people from five projects stated that they spent 05–15 minutes with search and analysis, while people from one project stated that it is spent 05–10 minutes, and one project stated that it is spent 20–30 minutes. Moreover, the average time of these searches is 12.5 minutes (such mean was computed using the individual responses from reporters). In addition, metric M_4 shows an average of 231.5 bug reports submitted per day.

Thus, we have an approximate average of 48 person-hours spent per day only with searches for similar bug reports. So, if we could reduce that time in half by using some techniques to suggest similar bug reports, for example, we would save 24 person-hours per day.

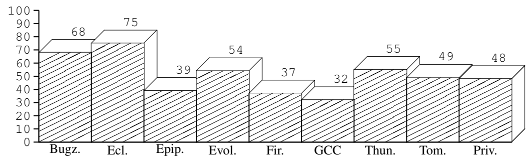

Figure 2. M_3**: unique vs. duplicates ratio (%).**

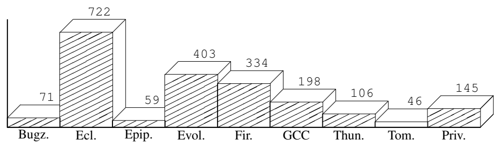

Figure 3. M_4**: bug reports frequency (daily).**

Another important point, shown by the metric M_3, is the fact that a duplicate is resolved using half the time needed to resolve a valid bug report, on average. Thus, reducing the amount of duplicate bug reports coming into a repository would spare more time to resolve valid bug reports.

Question 3. Repositories with most groups classified as one-to-many may take advantage of techniques such as clustering or machine learning[7]. This is because such techniques work better when there is more data from where they can learn. All projects presented in this study have more than 80% of bug report groups characterized as one-to-one. Such situation shows us that it is necessary to investigate more adequate techniques to treat environments where one-to-one relationships predominate. As mentioned before, machine learning and clustering techniques are not adequate for this case.

Question 4. An analysis of the distribution of the bug report types showed that more than 87% of the bug reports are related to defects. Thus, it is natural that there are more defect duplicates in the repository. We needed to understand if this difference also holds when calculating the relative percentage of duplicate bug reports for each bug report type.

Figure 4 shows the relative duplication ratio for defects and enhancement bug reports, and their difference in terms of percentages, for each open source project. We did not compute it for the private project because all bug reports from this project are about defects.

The ratios are distributed without major differences among the projects. Some projects have relatively more duplicate defects than duplicate enhancements, while in other projects the opposite is true. The mean of the sum of all differences is 3.23%, which indicates that duplicate reports for defects are slightly more frequent than duplicate reports for enhancements. But this is such a small value, and we can safely conclude that bug reports duplication is

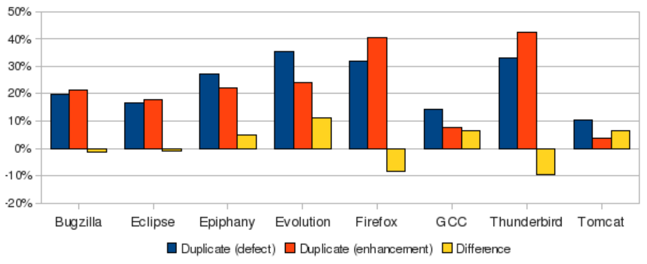

Figure 4. Duplication ratio.

not influenced by the bug report type.

Question 5. Figure 5 shows the contribution of each type of submitter to the number of duplicates of each project. As it can be seen, most of the duplicates are submitted by sporadic submitters, followed by average and frequent submitters. The average and frequent submitters contribute very little to the problem of duplication.The sporadic submitters are more prone to submit duplicate bug reports.

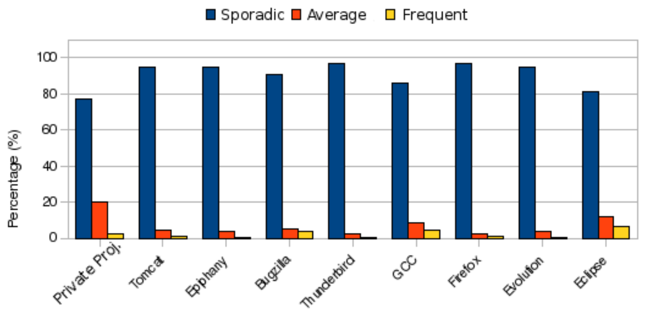

Figure 5. Submitter profiles and duplication.

6 Related Work

In work of Jhon Anvik [2], it was analyzed potential problems raised by bug repositories from open source projects. Hiew's work [8] presented an approach to detect duplicate bug reports using clustering techniques. Runeson's work [5] addressed the bug report duplication problem using Natural Language Processing (NLP). While Wang et al. [9] proposed an approach to detect duplicate bug reports using NLP and execution information.

Jalbert and Weimer [7] build a system to automatically classify duplicate bug reports as they arrive. Bettenburg et al. [6] approched the same problem by providing developers with additional information, such as *tracebacks*.

Sandusky et al. [10] proposed a method to identify and visualize bug report networks. With such networks, it was possible to analyze relationships and dependencies among bug reports.

7 Conclusion

In this work, we presented an initial characterization study, using the GQM method [4], about the bug report duplication problem. The study was performed using 8 open source projects and one private project.

According to our analysis, all 9 projects are being affected by the bug report duplication problem. Furthermore, we found evidences that the productivity is being impacted because of the same problem.

For future work, we are planning to investigate some projects' characteristics that could be factors to the duplication problem, such as: staff size, number of submitters, software size, software life-time, bug repository size, quality of bug report descriptions, and submitters' profile.

Acknowledgement

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[2]), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08.

References

[1] N. Serrano and I. Ciordia. Bugzilla, itracker, and other bug trackers. *IEEE Soft.*, 22(2):11–13, March-April 2005.

[2] J. Anvik, L. Hiew, and G. C. Murphy. Coping with an open bug repository. In *Proc. of the 2005 OOPSLA workshop on Eclipse technology eXchange*, pages 35–39, New York, NY, USA, October 2005. ACM Press.

[3] H. Kagdi, M. L. Collard, and J. I. Maletic. A survey and taxonomy of approaches for mining software repositories in the context of software evolution: Survey articles. *Journal of Soft. Maint. and Evol.*, 19(2):77–131, 2007.

[4] V. Basili, R. Selby, and D. Hutchens. Experimentation in soft. eng. *IEEE Trans.*, 12(7):733–743, July 1986.

[5] P. Runeson, M. Alexandersson, and O. Nyholm. Detection of duplicate defect reports using natural language processing. In *Proc. of the 29th Inter. Conf. on Soft. Eng.*, pages 499–510. IEEE Press, May 2007.

[6] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim. Duplicate bug reports considered harmful? In *Inter. Conf. on Soft. Maintenance*, pages 337–345. IEEE Press, 2008.

[7] N. Jalbert and W. Weimer. Automated duplicate detection for bug tracking systems. In *The 38th Annual IEEE/IFIP Inter. Conf. on Dependable Systems and Networks*, pages 52–61. IEEE Press, June 2008.

[8] L. Hiew. Assisted detection of duplicate bug reports. Master's thesis, The University of British Columbia, 2006.

[9] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun. An approach to detecting duplicate bug reports using natural language and execution information. In *Proc. of the 13th Inter. Conf. on Soft. Eng.*, pages 461–470. ACM Press, 2008.

[10] R. J. Sandusky, L. Gasser, and G. Ripoche. Bug report networks: Varieties, strategies, and impacts in a f/oss development community. In *Proc. of the 1st Inter. Workshop on Mining Soft. Repositories*, pages 80–84, University of Waterloo, Waterloo, May 2004.

[2]INES - `http://www.ines.org.br`