Differences of Time between Modification and Re-modification: An Analysis of a Bug Tracking System

Akinori Ihara, Masao Ohira, and Ken–ichi Matsumoto

Graduate School of Information Science, Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara, JAPAN 630-0192
{akinori-i,masao,matumoto}@is.naist.jp

Abstract. Managers of open source projects need to understand time to resolve bugs, which are reported into a bug tracking system on a daily basis, to make a release plan. Hewett et al. proposed an empirical approach to predicting time required to repair bugs. However, the predictive model did not distinguish between time to modify a new bug and time to modify a reported bug. In this paper, toward predicting time to resolve bugs with accuracy, we identify such the differences of time between bug modifications and re-modifications. We have conducted a case study using Firefox project data. As a result of this case study, we have confirmed that time to modify a reopened bug bug was shorter than time to modify a new bug.

Key words: open source software development, bug tracking system, time to resolve bugs, bug modification process, Firefox

1 INTRODUCTION

As open source software with a large number of users increases, it is required to release a new feature or a bug fix on regular basis. Therefore, managers of OSS projects need to understand time to resolve bugs which are reported into a bug tracking system on a daily basis, in order to make a release plan. Hewett et al.[2] proposed an empirical approach to predicting time for repairing bugs. The study presented a bug modification process using a bug tracking system as a state transition diagram and predicted time spent for transition to each state. However, the predictive model did not distinguish between time to modify a new bug and time to modify a reported bug. Time required for modifying a reported bug would be shorter than time required for modifying a new bug, since a problem in source codes must be more clear, compared to the modification of a new bug which may contain unknown problems. It will also take a longer time to resolve bugs, if developers in charge of bug modifications frequently change. In this paper, toward predicting time to resolve bugs with accuracy, we identify such the differences of time between bug modifications and re-modifications. We have conducted a case study using Firefox project data.

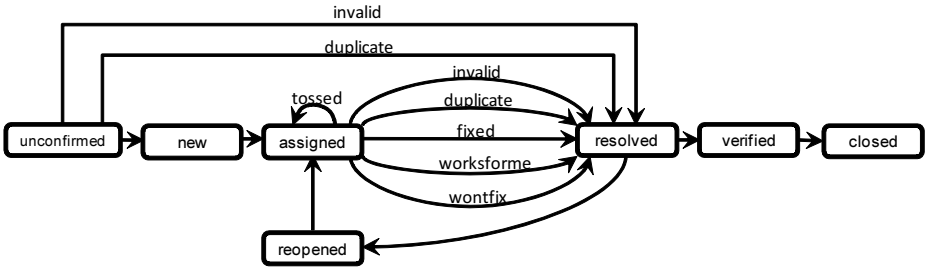

Fig. 1. A bug modification process[3]

2 RELATED WORK

There are many studies on bug modification processes with bug tracking systems in open source projects[1][3][4][5]. Focusing on time to resolve bugs in the bug modification process, we have proposed an analysis method to understand a factor which results in prolonging the bug modification process[4]. This analysis method represents a bug modification process as a state transition diagram and calculates the amount of time required to transit between states. We have conducted two case studies of the reported bugs in Apache and Mozilla projects. As a result of our analysis, we have found that the both projects needed long time to resolve bugs in the modification phase and verification phase. However, we could not achieve a clear understanding on the differences of time between bug modifications and re-modifications.

3 BUG MODIFICATION PROCESS

3.1 Bug Modification Process with a Bug Tracking System

Most open source projects use bug tracking systems to unify management of bugs found and reported by developers and users in their projects. A bug tracking system helps open source project's managers to know the progress of bug modifications, to avoid leaving unmodified bugs and so forth. Popular bug tracking systems include Bugzilla, Mantis, RedMine, Trac and so on.

Figure 1 represents a bug modification process using a bug tracking system. Although a bug modification process using a bug tracking system slightly differs among individual bug tracking systems, it substantially can be represented as a state transition diagram in Figure 1.

3.2 Modification Work

This section describes differences of the modification work flow between modification of a new bug and modification of a reopened bug. Figure 1 shows the

bug modification work flow. The modification work flow for a firstly reported bug is as follows. First, developers understand contents of bug reports. Second, developers understand a source code containing bugs. System names containing bugs are written in many bug reports. One developer often faces with the difficulty in modifying bugs by oneself, because most software systems work with other systems. Furthermore, developers need to consider the influence of their modifications on other source codes, due to such as the dependency of software modules. In contrast, re-modifying bugs would be finished by a shorter time, because analyzing the prior modification helps developers identify the reason and/or location of bugs. For the reasons mentioned above, we consider that time to resolve is affected by the presence or absence of the history of modifications.

4 ANALYSIS METHOD

This section describes a method for identifying the differences of time between bug modifications and re-modifications. At first, we describe a method to calculate time to modify a new bug and time to modify a reopened bug. Time to modify a new bug is defined by the mean time from acceptance of the bug (**new**) to resolution of the bug (**resolved**). In contrast, time to modify a reopened bug is defined by time from the decision of re-modification of the bug (**reopen**) to resolution of the bug (**resolved**). Some reported bugs are often needed to be re-modified several times. In this case, we calculate time for each re-modification of bugs. For example, if a bug are re-modified twice, we count time for the two re-modifications Therefore, time required to re-modify a bug depends on the number of changes of developers in charge [3]. In this paper, time to modify a new bug and time to modify a reopened bug are respectively calculated by the number of modifications of each developer in charge.

5 CASE STUDY

5.1 Target Projects and Data

In the Mozilla Firefox project, Bugzilla is used to manage reported bugs. The Mozilla Firefox project has been developing a web browser product with a rapidly increasing share. The product is very popular due to the extensibility of functions (i.e., add-ons). The project has been using Bugzilla since 2001. In the case study, history data of Bugzilla in Firefox version 1.0, 2.0, and 3.0 had been examined. In this paper, we analyzed 10,917 bug reports. Their bug reports is closed bugs from 2003 to 2008. In the bug reports, the number of re-modified bugs were 434 and the number of firstly reported bugs were 969.

5.2 Result

Table 1 respectively shows time to modify a new bug and time to modify a reopened bug, by the number of changes of assigned developers. If an assigned

developers never changed, the number of changes of assigned developers is shown as zero. Time to modify new bugs with zero, once and twice assigned developers is longer than time to modify reopened bugs with zero, once and twice assigned developers. In addition, the number of new bugs excepting zero assigned developers is 426 of 969 (44%). The number of re-modified bugs excepting zero assigned developers is 107 of 434 (25%).

Table 1. Time to firstly-modify and re-modify bugs

	new bugs					reopened bugs				
number of assigned developers	zero assigned	once	twice	third times	more than four times	zero assigned	once	twice	third times	more than four times
median(days)	11.0	24.3	82.7	129.3	429.2	0.6	8.2	75.7	163.2	227.4
average(days)	78.2	113.2	197.3	363.5	331.2	28.6	94.8	298.8	201.8	286.6
variance(days)	167.4	227.5	348.0	439.3	212.1	98.8	170.9	627.9	160.6	210.1
number of bugs	543	353	47	19	7	386	74	20	8	5

6 DISCUSSIONS

Based on the results of our case study, this section discusses the necessity of analysis on the differences of the mean time to resolve bugs due to kinds of bug modification process toward building a predictive model of bug resolution time.

As a result of this study, we observed that time to modify new bugs was shorter than time to modify a reopened bug in spite of the number of assigned developers. We also found that the number of assigned developers in modifying new bugs was larger than that in modifying reopened bugs. In addition, the number of assigned developers in re-modifications is less than that in modifications of firstly reported bugs. Therefore, we think that time to understand bug reports and source codes with bugs is less required in re-modifying bugs.

In this paper, we did not analyze time to resolve bugs, considering every developer's skills and priority and/or severity of bugs. If a developer has high skill, time to resolve bugs would be shorten in nature. Also, if high severity bugs are reported, developers would modify such the bugs by priority. In the future, we would analyze time to resolve bugs, considering such the skills and priority and/or severity of bugs.

7 CONCLUSION AND FUTURE WORK

In this paper, toward predicting time to resolve bugs with accuracy, we identify such the differences of time between bug modifications and re-modifications. We have conducted a case study using Firefox project data. As a result of this case

study, we have confirmed that time to resolve a firstly reported bug was shorter than time to re-modify a bug.

We think that the verification work also different between time for a first bug modification and time for re-modification of a bug. Finally, we would like to build a predictive model of bug resolution time, analyzing differences of time to resolve bugs modification considering bug modification processes.

8 ACKNOWLEDGEMENT

We appreciate the anonymous reviewrs giving insightful comments and helpful suggestions. This research is being conducted as a part of the Next Generation IT Program and Grant-in-aid for Young Scientists (B), 20700028, 2009 by the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

1. I. Herraiz, D. M. German, J. M. Gonzalez-Barahona, and G. Robles. Towards a simplification of the bug report form in eclipse. In *Proceedings of the 2008 international working conference on Mining software repositories (MSR'08)*, pages 145–148, 2008.
2. Hewett. R, and Kijsanayothin. P. On modeling software defect repair time. *Empirical Softw. Engg.*, pages 165–186, 2009.
3. Jeong. G, Kim. S, and Zimmermann. T. Improving bug triage with bug tossing graphs. In *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering on European Software Engineering Conference and Foundations of Software Engineering Symposium* , pages 135–144, 2009.
4. Ihara. A, Ohira. M, and Matsumoto. K. An analysis method for improving a bug modification process in open source software development. In *Proceedings of the Joint international and Annual ERCIM Workshops on Principles of Software Evolution (IWPSE) and Software Evolution (EVOL) Workshops* , pages 111–120, 2009.
5. A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346, 2002.