A New Perspective on the Socialness in Bug Triaging: A Case Study of the Eclipse Platform Project

Masao Ohira
Faculty of Systems Engineering, Wakayama University
930, Sakaedani, Wakayama, JAPAN
masao@sys.wakayama-u.ac.jp

Hayato Yoshiyuki
Faculty of Systems Engineering, Wakayama University
930, Sakaedani, Wakayama, JAPAN
s151054@sys.wakayama-u.ac.jp

ABSTRACT

This paper explores how social relationships among developers impact on the efficiency of bug fixes. From the case study of the Eclipse Platform project, we found that (1) past achievements of bug triaging by particular pairs of assignors and fixers do not necessarily impact on the time to fix bugs, (2) rather, the time required to fix a bug can depend on who assigns the bug fixing task to a fixer. These findings would imply that we need to not only consider *who should fix this bug?* but also take into account a fixer's perspective *who should assign this bug?* or *who should ask to whom?*, in order to better support the bug triaging process.

Categories and Subject Descriptors

K.6.3 [**Management of Computing and Information Systems**]: Software Management—*software development, software maintenance, software process*

General Terms

Management

Keywords

bug triaging, social perspective, open source software development

1. INTRODUCTION

Triaging bugs in a large-scale software development is very important to keep providing quality products. To improve the bug triaging process, prior work proposed several promising approaches, for example, automated bug triaging methods [2, 5, 7], duplicate bugs detection [10], understandings of the rationale for the reassigning and re-opening of bugs [6, 9] and so forth. Our study in this paper continues this line of work of exploring efficient bug triaging in a bug-fixing process, by closely looking at the individuals involved in the process.

SSE'13, August 18, 2013, Saint Petersburg, Russia
Copyright 2013 ACM 978-1-4503-2313-0/13/08...$15.00
http://dx.doi.org/10.1145/2501535.2501542

The existing approaches to bug triaging are based on the notion *who should fix this bug?* [1] that implies an assignor needs a help to correctly assign a bug fixing task to an appropriate developer (fixer). In fact, 44% of bugs in the Eclipse project are reassigned to more than one developer [7]. In our understanding, the approaches are come up with to support assignor's bug triaging tasks (i.e., assignor's perspective). In this study, we investigate how the relations between the individuals involved in the process impact on the efficiency of the bug fixing, from the fixer's perspective.

Figure 1 shows the differences of perspectives between existing studies and our study. We believe that we need the fixer's perspective to better support the bug triaging process, because most of bug modification activities rely on the minority of developers in the project [8] and the developers often need to concurrently deal with many bugs which are assigned by multiple assignors. Even if an assignor can find an appropriate developer (fixer) to fix a particular bug, s/he might be too busy to start working on the bug. Moreover, his/her bug fixing performance (i.e., the time required to fix the bug) might not only depend on the technical level of fixing the bug, but also might be influenced by social relationships between the assignor and the fixer, since s/he might have to decide *which task should be completed first?* after taking care of her/his working hours. To better understand the impact of the both perspectives on the time to fix bugs, we would like to answer the following research questions in this study.

RQ1: *Does social relationships between assignors and fixers impact on the time to fix bugs?*
RQ2: *Does the bug fixing performance of a fixer depend on who assigns tasks to the fixer?*

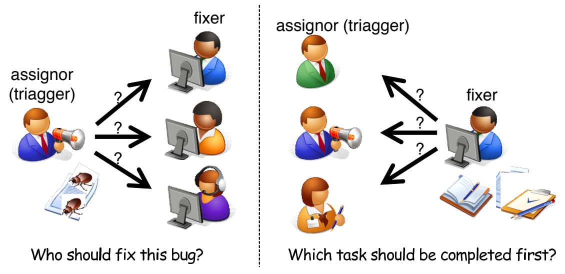

Figure 1: Difference of perspectives between existing studies and our study.

2. CASE STUDY

2.1 Dataset: Eclipse Platform Project

For our case study, we collected bug reports from the Eclipse Platform project where all the bug information is managed by one of the most popular bug tracking systems called Bugzilla. Our original data set included 52,593 bugs reported from October 2001 to October 2012. After collecting the original data set using our web crawler program, we carried our data cleaning and several preprocessing filters. The data cleaning is required to identify a developer who uses several email addresses. According to [4], we cleaned the data set and also checked it manually. Then we filtered the data set to focus on bug reports which were FIXED without reassignments although reassignments to other developers often happen in a large-scale project [9], since we would like to simplify the data analysis to better capture the phenomenon. We also remove the data if we can only count the same pair of assignor-fixer less than three times. After cleaning and filtering, our data set has 20,422 bugs in total.

2.2 Assignor's Perspective

We first try to answer the following research question from the assignor's perspective.

RQ1: *Does social relationships between assignors and fixers impact on the time to fix bugs?*

Motivation: In case a particular pair of an assignor and a fixer has substantial experience in working together for fixing bugs, the time to fix the bugs would be shorter than other pairs since the assignor is likely to know what the fixer is knowledgeable about. If it is true, bug triaging might not matter actually because *who should fix this bug?* would be easily resolved if an experienced assignor ask an experienced fixer to fix the bug. If the time required to fix the bugs dose not depend on the assignor-fixer relationship, bug triaging would still matter and need to be improved.

Approach: The term "social relationship" can have various meanings in different contexts. In this paper, we regard it as the strength of the partnership between an assignor and a fixer, which can be partly captured by past achievements of bug triaging between the two developers. To investigate the social relationship between developers, we calculate how many tasks were assigned from whom (assignor) to whom (fixer) in the past.

From our data set, we extracted the top 5 assignors who most assigned bug-fixing tasks to fixers and then extracted most assigned 5 fixers by each assignor. We counted the number of assignments by each assignor (i.e., how many bugs did each assignor ask fixers to fix?) and calculated median days to fix the assigned bugs.

Results: Table 1 shows the total number of assignments of bug fixing tasks by the top 5 assignors in the Eclipse Platform project. We can see assignments from assignor A_a and A_b were completed in a surprisingly-short time (less than one hour) while assignor A_c's assignments requires about 26 days to be completed.

Table 2 shows the number of assignments to the top 5 fixers who most fixed bugs in the past and median days to fix the bugs. For assignor A_a, A_d and A_e, the median days to spend to fix bugs which were assigned to the top 5 fixers is almost the same as or less than the median days (shown in Table 1) to spend to fix all the bugs.

In contrast to A_a, A_d and A_e, assignments from A_b and A_c to the top 5 fixers tends to need a longer time than the median days (shown in Table 1) to fix all the bugs. For instance, the median days for all the bugs of A_c is 8.27 days, but the median days to spend to fix bugs by the top 5 fixers exceeded 8.27 days, except $F11$. The same thing can be applied to A_d.

These imply that past achievements of bug triaging by a particular pair of developers (i.e., assignor and fixer) do not necessarily impact on the time to fix bugs. In other words, a bug is not necessarily resolved soon even if it is triaged by a particular pair of developers who had co-worked to resolve a large number of bugs in the past. We also could not confirm that there are strong correlations between the number of assignments and the median days to fix bugs in Table 1.

Past achievements of bug triaging by a particular pair of developers (i.e., assignor and fixer) do not necessarily impact on the time to fix bugs.

Table 1: Total number of assignments by the top 5 assignors and median days to complete the assignments

Assignor	Num. of assignments	Median days to fix bugs
A_a	1,529	0.03
A_b	1,500	26.09
A_c	1,343	8.27
A_d	1,186	0.00
A_e	895	8.08

Table 2: Top 5 assignors' task assignments and median days to be fixed by assignees (fixers)

Assignor $\rightarrow$ *Fixer*	Num. of assignments	Median days to fix bugs
$A_a \rightarrow F1$	290	0.00
$A_a \rightarrow F2$	224	0.00
$A_a \rightarrow F3$	242	0.02
$A_a \rightarrow F4$	457	0.04
$A_a \rightarrow F5$	187	0.08
$A_b \rightarrow F6$	428	20.54
$A_b \rightarrow F7$	288	26.09
$A_b \rightarrow F8$	138	27.33
$A_b \rightarrow F9$	158	35.04
$A_b \rightarrow F10$	103	85.10
$A_c \rightarrow F11$	262	2.93
$A_c \rightarrow F12$	266	13.88
$A_c \rightarrow F13$	71	15.26
$A_c \rightarrow F14$	64	17.12
$A_c \rightarrow F15$	64	28.81
$A_d \rightarrow F16$	180	0.00
$A_d \rightarrow F17$	270	0.00
$A_d \rightarrow F18$	92	0.00
$A_d \rightarrow F19$	255	0.00
$A_d \rightarrow F20$	91	1.00
$A_e \rightarrow F21$	45	2.94
$A_e \rightarrow F22$	53	4.15
$A_e \rightarrow F23$	89	4.71
$A_e \rightarrow F24$	144	5.50
$A_e \rightarrow F25$	77	5.62

2.3 Fixer's Perspective

The next research question is our main focus in this study.

RQ2: *Does the bug fixing performance of a fixer depend on who assigns tasks to the fixer?*

Motivation: We can see that there are 25 fixers in Table 1. Each fixer is also asked to fix other bugs by other assignors. We would like to know If each fixer shows the same performance in fixing bugs (i.e., the time to fix a bug is not so different due to assignors.) If not, bug triaging becomes much more complex because we need to consider both *who should fix this bug?* and *who should assign this bug?* to improve the time to fix bugs.

Approach: From our data set, we extracted the top 5 fixers who most fixed bugs and then extracted top 5 assignors by each fixer. We counted the number of assignments to each fixer (i.e., how many bugs did each fixer fix?) and calculated median days to fix the assigned bugs.

Results: Table 3 shows the total number of assignments to the top 5 fixers in the Eclipse Platform project. We can see assignments to fixer F_a and F_d were completed in a short time while fixer F_c requires about 20 days to fix an assigned bug.

Table 4 shows the number of assignments to the top 5 fixers who most fixed bugs in the past and median days to fix the bugs. We can see that each fixer shows better or worse performance that depends on assignors (i.e., who assigned tasks to fixers). For instance, as shown in Table 3, fixer F_b takes 8.17 median days if s/he fixes all the assigned (1,088) bugs. However, when bugs are assigned by assignor $A9$ who most assigned to F_b and so would know about F_b, her/his bug fixing performance become worse (13.88 median days) than others. Similarly assignments from assignor $A22$ made F_e's bug fixing performance worse. Although assignor $A22$ assigned the largest number of bug fixing tasks to F_e, the time to complete the assignments takes extra 13 days (26.09 days), compared to F_e's median days (13.18 days) to fix all the bugs. We discuss these phenomena later.

We also could not confirm that there were strong correlations between the number of assignments and the median days to fix bugs in Table 1. A fixer does not necessarily better perform even when particular assignors ask her/him to fix bugs.

The time required to fix a bug depends on who assigns tasks to whom (fixer).

Table 3: Total number of bugs assigned to the top 5 fixers and median days to fix the bugs

Fixer	Num. of assigned bugs	Median days to fix bugs
F_a	1,231	0.17
F_b	1,088	8.17
F_c	952	20.06
F_d	901	0.13
F_e	864	13.18

Table 4: Top 5 fixers' assigned bugs and median days to fixe the bugs

Fixer $\leftarrow$ *Assignor*	Num. of assigned bugs	Median days to fix bugs
$F_a \leftarrow A1$	457	0.04
$F_a \leftarrow A2$	127	0.16
$F_a \leftarrow A3$	120	0.21
$F_a \leftarrow A4$	171	0.78
$F_a \leftarrow A5$	156	0.94
$F_b \leftarrow A6$	80	3.50
$F_b \leftarrow A7$	144	5.50
$F_b \leftarrow A8$	154	6.01
$F_b \leftarrow A9$	266	13.88
$F_b \leftarrow A10$	69	36.06
$F_c \leftarrow A11$	89	6.97
$F_c \leftarrow A12$	428	20.54
$F_c \leftarrow A13$	71	21.97
$F_c \leftarrow A14$	171	22.00
$F_c \leftarrow A15$	115	39.29
$F_d \leftarrow A16$	290	0.00
$F_d \leftarrow A17$	255	0.00
$F_d \leftarrow A18$	159	4.13
$F_d \leftarrow A19$	67	7.89
$F_d \leftarrow A20$	52	8.04
$F_e \leftarrow A21$	44	5.44
$F_e \leftarrow A22$	288	26.09
$F_e \leftarrow A23$	12	38.67
$F_e \leftarrow A24$	118	60.38
$F_e \leftarrow A25$	72	61.15

3. DISCUSSIONS

This section further discusses the results obtained from the case study and the threats to validity in this study.

3.1 Achievements vs. Efficiency of Bug Fixes

In answering RQ2, we found that there were several pairs of assignors and fixers who made the bug fixing performance worse than their median days to fix bugs as a whole.

For instance, although fixer F_b (= $F12$ in Table 2) is most assigned by $A9$ (= A_c in Table 2) and assignor A_c assigned the largest number of tasks to F_b, the result (i.e., the time to fix bugs) is not so good each other. Figure 2 shows all the fixers who were assigned by A_c and all the assignors who assigned to F_b and their performance. The top of Figure 2 indicates that F_b is the 9th fastest fixer among the 24 fixers who are assigned by A_c. The bottom of Figure 2 indicates that F_b completed bug fixing tasks which were assigned by A_c in the 12th fastest among the 18 assignors. The contributions (i.e., the number of bug fixes) from the pair of A_c and F_b is large enough and so can be considered as very productive. However, as we see in Figure 2, the bug fixing performance of the pair of A_c and F_b is not so efficient.

From these findings, we can say that the current support for bug triaging is not sufficient because the proposed approaches in existing studies are based on the notion *who should fix this bug?* and then focus on finding an appropriate developer for each bug. We suggest that we need to consider a fixer's perspective *who should assign this bug?* or *who should ask to whom?* in order to better support the efficient bug triaging.

3.2 Threats to Validity

In this study we used the bug report data without reassignments in the Eclipse Platform projects to obtain a clear understanding of socialness in the bug triaging process. Such data selection (including the limited term of data and

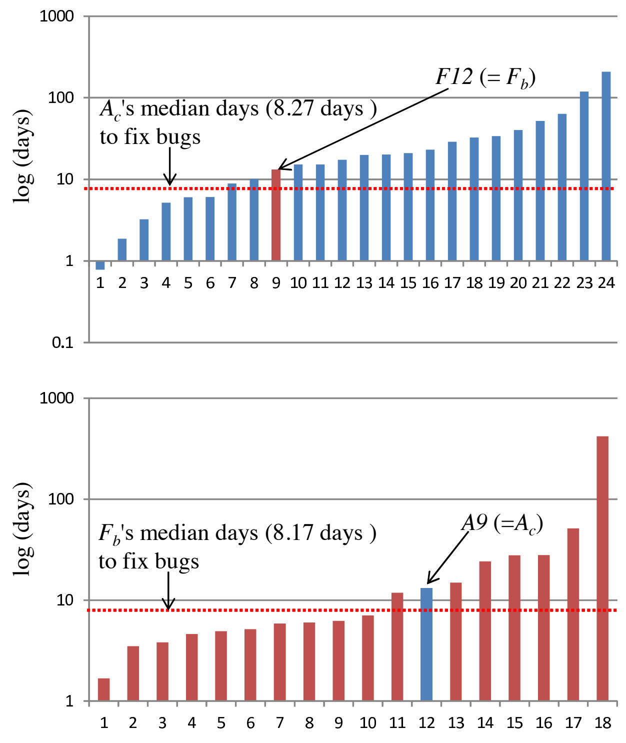

Figure 2: Overall performance of A_b (= $A9$) and fixer F_b (= $F12$). The top figure indicates all the fixers who were assigned byA_b and their bug fixing performance (log (days)). The bottom figure shows that all the assignors who assigned tasks to F_b and F_b's bug fixing performance (log (days)).

excluding reassignments) might bring bias [3] against the complete picture of open source development, we need to conduct exhaustive analyses to make our results much more valuable in the future.

We only studied the Eclipse Platform project. The project is large scale, successful and so they have sufficient bug report data to validate results of our case study. However they have many developers who are fully employed by IBM and who dedicate considerable efforts to development of the Eclipse products since the Eclipse projects originally started as a corporate project of IBM. Our findings in this study might not be applicable to any other open source projects.

In this study, we did not analyze contents and properties (e.g., such as priority and severity) of bug reports. We need not only to analyze other OSS projects but also to analyze qualitative aspects of bug reports in the future to strengthen the conclusions of our study.

4. CONCLUSION AND FUTURE WORK

In this paper, we explored the bug triaging process from a social perspective. In particular, we studied the impact of the fixer's perspective on the efficiency of the bug triaging. We found that:

1. past achievements of bug triaging by a particular pair of developers (i.e., assignor and fixer) do not necessarily impact on the time to fix bugs.
2. Rather, the time required to fix a bug can depend on who assigns tasks to the fixer.
3. In order to better support the efficient bug triaging, we need to not only consider *who should fix this bug?* but also take into account a fixer's perspective *who should assign this bug?* or *who should ask to whom?*.

In future work, we wish to investigate whether our findings hold when we integrate other OSS projects. We also wish to propose a new prediction model to predict *who should assign this bug?* and the time to be fixed, based on our findings.

5. ACKNOWLEDGMENT

This research is conducted as part of Grant-in-Aid for Scientific Research (B) 23300009 and (C) 24500041 by Japan Society for the Promotion of Science (JSPS).

6. REFERENCES

[1] J. Anvik, L. Hiew, and G. C. Murphy. Who should fix this bug? In *Proceedings of ICSE '06*, pages 361–370, 2006.

[2] J. Anvik and G. C. Murphy. Reducing the effort of bug report triage: Recommenders for development-oriented decisions. *ACM Trans. Softw. Eng. Methodol.*, 20(3):10:1–10:35, Aug. 2011.

[3] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu. Fair and balanced?: bias in bug-fix datasets. In *Proceedings of ESEC/FSE '09*, pages 121–130, 2009.

[4] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *Proceedings of MSR '06*, pages 137–143, 2006.

[5] D. Cubranic and G. C. Murphy. Automatic bug triage using text categorization. In *Proceedings of SEKE2004*, pages 92–97, 2004.

[6] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy. "Not my bug!" and other reasons for software bug report reassignments. In *Proceedings of CSCW '11*, pages 395–404, 2011.

[7] G. Jeong, S. Kim, and T. Zimmermann. Improving bug triage with bug tossing graphs. In *Proceedings of ESEC/FSE '09*, pages 111–120, 2009.

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

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

[10] C. Sun, D. Lo, X. Wang, J. Jiang, and S.-C. Khoo. A discriminative model approach for accurate duplicate bug report retrieval. In *Proceedings of ICSE '10*, pages 45–54, 2010.