The Impact of Bug Management Patterns on Bug Fixing: A Case Study of Eclipse Projects

Masao Ohira
Wakayama University
masao@sys.wakayama-u.ac.jp

Ahmed E. Hassan
Queen's University
ahmed@cs.queensu.ca

Naoya Osawa
and Ken-ichi Matsumoto
Nara Institute of Science and Technology
{*naoya-osa, matumoto*}*@is.naist.jp*

***Abstract*—An efficient bug management process is critical for the success of software projects. Prior work has focused on improving this process, for example, by automating bug triaging, detecting duplicate bugs, and understanding the rationale for re-opening bugs. This paper continues this line of work by exploring the people who are involved in the bug management process. In particular we develop four patterns that distill the different relations between the people involved in the process: the reporter, triager, and fixer of a bug. Through a case study on the Eclipse Platform and JDT projects, we demonstrate that these patterns have an impact on the efficiency of the bug management process. For example, we find that using our patterns project personnel can improve their efficiency through better communication about bugs before assigning them.**

I. Introduction

An efficient bug management process (reports, assignment and fixing) is critical for the success of software projects. Software projects use a Bug Tracking System (BTS) (e.g., Bugzilla [1]) to manage and keep track of bug management tasks efficiently. However, as the user base grows, some large open source projects such as the Eclipse and Mozilla projects have been faced with complex challenges to their bug management process, since they receive a large number of (sometimes over several hundred) bug reports on a daily basis. Developers must understand a new report, figure out if it is a real bug and whether it was reported in the past (i.e., duplicate bug), and assign it to the most appropriate person to fix the bug quickly. Correct bug triaging is very challenging. In fact, 44% of bugs in the Eclipse project are reassigned to more than one developer [2].

To improve the bug management process, prior work proposed several promising approaches, for example, automating bug triaging [2]–[4], detecting duplicate bugs [5]–[7], and understanding the rationale for the reassigning and re-opening of bugs [8]–[10]. This paper continues this line of work of exploring the bug assignment phase within the bug-fixing process. In particular we develop four different patterns that distill the different relations between the people involved in the bug management process: the reporter, triager, and fixer of a bug.

We suspect that the efficiency of the bug management process should depend on our patterns which we call *bug management patterns*. These patterns account for the different individuals assigned to the management process of a bug (reporter, triager, and fixer). To our knowledge, this is the first attempt to empirically study the impact of interaction patterns between the people in the bug management process on bug fixing, especially in terms of the time to assign bug fixing tasks and the time to fix a bug.

Through a case study on the Eclipse Platform and JDT projects, we demonstrate that these patterns have an impact on the efficiency of the bug management process as follows.

1) As the number of individuals involved in the process increases, the time to fix a bug increases. For example, the time to fix a bug (reported, triaged and fixed by three different individuals) takes around 2–3 times longer compared to other patterns. In some cases, these bugs are fixed quite fast. A manual analysis of the data shows that in such cases there was a large amount of discussion about the bug and the most suitable individuals to fix it.
2) Surprisingly if the individual triaging a bug ends up assigning it to himself for fixing, the bug assignment (to himself) takes 1.5-2 times longer. We believe this is often likely due to the triaging doing the self-assignment as a last resort. In the JDT project, we found that the team has a process in place to fast track the fixing of such bugs to offset the lengthier assignment time.

Our results highlight the importance of social and knowledge sharing factors on the bug management process. We note the need for better tools to facilitate the knowledge sharing and communication between project personnel. We also believe that researchers would benefit from integrating such social knowledge (i.e., roles and individuals) in their software quality models, in particular, models to predict the fix time of bugs.

The rest of the paper is organized as follows. Section II describes related work and motivation of this study. Section III shows the results of a pilot study that we conducted to understand the impact of different individuals playing different roles in the bug management process. Section IV introduces our bug management patterns which capture the relations among the individuals involved in the process.

978-1-4673-2312-3/12/$31.00

Section V presents the results of our case study. We discuss the contributions of the study based on our findings and the threats to validity in Section VI. Section VII concludes the paper.

II. Related Work and Motivation

Most existing studies focus on how to reduce the time to fix bugs as this time continues to increase especially in large Open Source projects. There are currently three promising approaches to improving the bug management process. In what follows, we describe the existing approaches and the motivation for our study.

A. How to make a good bug report?

A good bug report helps reduce the time to fix bugs. A good report helps developers to quickly find, replicate, and understand the bugs at hand. However developers' information needs in bug reports are often unsatisfied, since users do not know what information are required to fix a problem. Users rarely articulate the problem sufficiently for developers to fix them. For instance, users rarely report procedures to reproduce an error (e.g., sometimes they just say "This option does not work on my computer!"). Therefore, developers have to ask users to give more information again and again to identify and fix the problem.

In order to improve the cooperation on a bug report between developers and users, many studies [11]–[16] have interviewed developers to understand their information needs for effective bug fixing. For example, through interviews with over 150 developers and 300 reporters of the Apache, Eclipse and Mozilla projects, Bettenburg et al. [12], [16] found that steps to reproduce and stack traces are very essential for bug reports.

B. Detection of duplicate bug reports

Users often report problems reported by other users in the past or that have already been fixed by developers. Developers also sometimes try to resolve the same problem which had been previously resolved. This can happen because there are a large number of bug reports in BTS. Both the users and developers cannot be aware of all the reported bugs though the provided search functionalities. In this manner, the bugs are duplicated in BTS. This results in wasting developers' time and efforts.

To avoid duplicate bug reports in BTS, several studies [5]–[7], [17] have tried to detect duplicate bug reports automatically. For example, Wang et al. [7] presented an approach to detect duplicate bugs, applying a natural language processing technique to bug report data in BTS.

C. Re-opening and reassigning of bug reports

Even if a bug fixing task is assigned to one developer, it may not be completed by the developer instead it might get reassigned (*tossed* [2]) to other developers. This often happens because a triager assigns the task to an inappropriate developer who does not have sufficient knowledge and skill to complete the task. In the Eclipse and Mozilla projects, 37% to 44% of bugs are reassigned to another developer [2]. Preventing the bug tossing (ensuring the assignment of a bug fixing task to the most appropriate developer) is a very effective way to reduce the time needed to fix bugs.

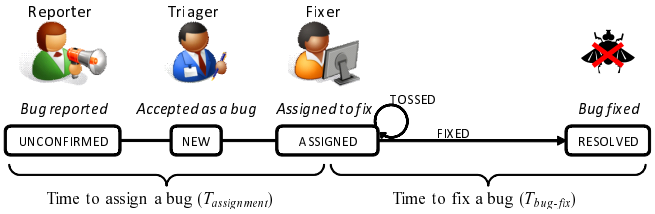

Figure 1. A bug management process

Several approaches [2]–[4], [8]–[10], [18]–[20] have tackled this issue. For instance, Anvik et al. [3] proposed an approach to assign a bug to an appropriate developer based on past bug reports data using natural language processing techniques. Jeong et al. [2] proposed a method for the bug assignment based on a social graph which captures the social relationships among developers in the bug assignment phase. Other approaches involve understanding the rational for multiple reassignments [8] and creating methods to predict which bugs are likely to be reopened [9], [10].

D. People in the bug management process: our motivation

As described above, prior studies have applied various data mining techniques to bug report data in BTS in order to support the bug management process, since the goal of existing studies, as previously described, is to reduce the time to fix bugs by providing means to write good bug reports [12], [14]–[16], finding adequate fixers for bug fixing tasks [2], [3], [18], and preventing redundant work for duplicated bug reports [5]–[7], [17].

In this study, we would like to create a new understanding of the bug management process by closely looking at the individuals involved in the process and the impact of the relations between the individuals on the efficiency of the bug fixing process. Figure 1 shows a typical bug management process with BTS such as Bugzilla [1]. In open source development with BTS, at least three types of individuals are involved in the bug management process: **bug reporter**, **triager**, and **fixer of a bug**. A **bug reporter** is a user or developer who reports a bug. A **triager** is often a senior developer who has the authority to assign a bug fixing task to other developers in the project. A **fixer** is a developer who fixes a bug that was assigned to him or her.

In this process, the triager obviously plays a very important role because she or he needs to have a good understanding of the bug report at hand and must assign

Table I
RESULTS OF OUR PILOT STUDY ON THE TIME TO ASSIGN BUG REPORTS TO FIXERS ($T_{assignment}$) IN PLATFORM AND JDT

project	Reporter = Triager ?	# of reports	ratio	average days	median days	SD	max days	min days	P-value
Platform	yes	1,000	24.2%	12.6	0.0	66.8	812.1	0.0	$< 0.01 **$
	no	3,133	75.8%	15.2	0.5	68.9	842.9	0.0	
JDT	yes	452	27,3%	10.6	0.0	57.8	713.7	0.0	$< 0.01 **$
	no	1,205	72.7%	20.0	0.5	79.6	927.0	0.0	

Table II
RESULTS OF OUR PILOT STUDY ON THE TIME TO FIX BUGS (T_{fix}) IN PLATFORM AND JDT

project	Triager = Fixer ?	# of reports	ratio	average days	median days	SD	max days	min days	P-value
Platform	yes	2,294	55.5%	23.1	1.2	65.3	776.0	0.0	$< 0.01 **$
	no	1,839	44.5%	46.9	5.9	111.1	988.2	0.0	
JDT	yes	817	49.3%	12.6	0.8	42.9	583.1	0.0	$< 0.01 **$
	no	840	50.7%	22.8	1.3	62.8	705.9	0.0	

the bug fixing task to the most appropriate developer who can fix the bug as quickly as possible. The triager can also be a reporter and/or a fixer. In such a case, the efficiency of the bug management process should be different from the case where each role is played by a different individual. Therefore we aim at establishing a clear understanding of the relation between the roles and the consequences of having different individuals in these roles on the bug management processes. While previous studies have mainly focused on reducing the time to fix bugs (T_{fix}), we are interested in both $T_{assignment}$ (time to assign a bug fixing task to a fixer) and T_{fix}.

III. PILOT STUDY

This section reports the results of our pilot study on the relation between the people involved in the bug management process and its impact on the efficiency of the bug fixing process.

A. Time to assign a bug fixing task

We first tried to answer the following research question.

RQ1: *Does the time to assign a bug fixing task depend on the fact that the same developer reports a bug and triages it?*

Motivation: Users as bug reporters occasionally cannot write good bug reports for developers. There are often mismatches between the information provided by users and the information triagers need to assign bug fixing tasks to appropriate fixers. This is one reason for lengthy $T_{assignment}$ because triagers spend considerable time to understand a filed bug.

In contrast, triagers can be bug reporters. In this case, the mismatches would be minimal and $T_{assignment}$ would be shorter than the case of bug reports by users because triagers are also developers who probably know what information is needed to fix bugs and also might know who could fix the bugs quickly. In short, we believe that $T_{assignment}$ largely depends on the reporter-triager relationship.

Approach: We collected bug reports data from two large open source projects: Eclipse Platform and Eclipse JDT projects. After cleaning the data, using the techniques described in [21], [22] to avoid counting the same individual with multiple email addresses as different individuals, we checked whether a bug reporter and a triager are the same individual or not. If an individual plays both the roles of a reporter and a triager, we considered him/her as a senior developer. If not (reporter $\neq$ triager), it is likely that a reporter is a user or a junior developer.

We filtered the data to focus on bug reports which were fixed without reassignments since we would like to simplify the data analysis to better capture the phenomenon. After the data filtering, we measured the average and median time (days) of $T_{assignment}$ and tested the differences of $T_{assignment}$ between individuals' roles (i.e., whether reporters are triagers or not) using the Mann–Whitney U test ($\alpha = 0.05$).

Results: Table I shows the results of our analysis. We found that there were statistically significant differences in $T_{assignment}$ in both projects. We also found that around 25% of the individuals in the projects were developers (reporter = triager) and that the average time of assignments of bug reports from developers is about 17–47% faster (2.6 days in Platform and 9.4 days in JDT), than that of assignments of bug reports from users. This result indicates that the assignment time for reports from users is a challenge for both Eclipse projects.

> The time to assign a bug fixing task depends on the fact that the same developer reports a bug and triages it.

B. Time to fix a bug

We were also interested in answering the following research question on the time to fix a bug (T_{fix}).

RQ2: *Does the time to fix a bug depend on the fact that the*

same developer triages a bug and fixes it?

Motivation: Even if a triager were to quickly find a fixer to assign a bug fixing task (i.e., $T_{assignment}$ is short), T_{fix} can be lengthy because there might exist other mismatches between the knowledge and/or the skills of the fixer and those needed to fix the bug. Jeong et al. [2] studied the "tossing (reassignment)" process to explore this issue. We further suspected the mismatches also happen even if a bug is not tossed to several fixers (i.e., if a bug is fixed by a single fixer). To better understand the impact of the mismatches on the time to fix bugs, we analyze T_{fix} without reassignments in the bug management process.

Approach: Using the cleaned and filtered bug report data in RQ1, we looked at whether a triager and a fixer are the same individual (triager = fixer) or not (triager $\neq$ fixer). We then measured the average and median time (days) of T_{fix} and tested the differences of T_{fix} with the Mann–Whitney U test ($\alpha = 0.05$). Note that our data did not include the "tossing" process where reassignment of a bug fixing task is required several times, in order to focus on the influence of the triager-fixer relationship. We only used bug reports that were "FIXED" by a single fixer.

Results: Table II shows the results of our analysis on T_{fix} for both projects. We found that there were significant differences in $T_{assignment}$ for both projects. Around 50% of the individuals in the projects played both roles of triager and fixer. The average time of bug fixing by triagers was about two times faster (23.8 days in Platform and 10.2 days in JDT), than that by fixers (regular developers).

> The time to fix a bug depends on the fact that the same developer triages a bug and fixes it.

These results of our pilot study motivate us to dig deeper in the relationships among the individuals (reporter, triager, and fixer) involved in the bug management process.

IV. BUG MANAGEMENT PATTERNS

In our pilot study from the previous section, we studied the two types of pairs of individuals (i.e., reporter-triager and triager-fixer) in the bug management process and analyzed the relationships between the types and the time ($T_{assignment}$ and T_{fix}). Although the results of the analysis provided clear understandings of the relationships between them, the results encouraged us to investigate the combined consequence of all three roles on $T_{assignment}$ and T_{fix}.

In this section we introduce the bug management patterns which provide us with a framework to study the bug management process and its impacts on bug fixing (i.e., $T_{assignment}$ and T_{fix}). The patterns are defined by combinations of the individual roles (i.e., bug reporter, triager, and fixer). Figure 2 illustrates the four patterns.

Pattern #1: Reporter=Triager=Fixer [R=T=F]

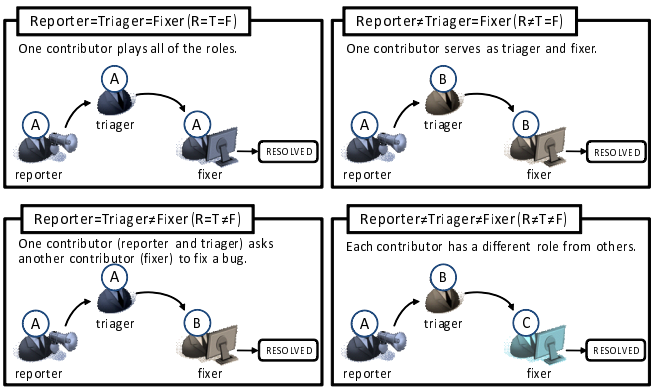

Figure 2. Four patterns of bug management process

In this pattern, a single individual plays all the roles (i.e., reporter, triager, and fixer of the bug). The patterns [R=T=F] is assumed to make bug fixing faster since the individual likely knows the bug source and has good confidence in his ability to fix the bug.

Pattern #2: Reporter=Triager≠Fixer [R=T≠F]

In this pattern, one individual (A) has the roles of both reporter and triager, and another individual (B) fixes the bug as a fixer. The pattern [R=T≠F] is assumed to make the bug assignment faster as shown previously in the result of RQ1 because individual (A) would be a developer who is authorized to assign the bug. In contrast, the pattern [R=T≠F] might make bug fixing inefficient such as the case of the Eclipse projects in RQ2 because the individual (B) as a fixer might be just entrusted by the individual (A) who does not have the knowledge nor the skills of the individual (B). Of course, even in this pattern, bug fixing can be done efficiently if the individual (A) is capable of correctly assigning bug reports to a fixer, that means the individual (A) is well aware of the knowledge and skills of individual (B) and that the skills match the ones needed for fixing the bug.

Pattern #3: Reporter≠Triager=Fixer [R≠T=F]

In this pattern, one individual (A) reports a bug and another individual (B) assigns the bug report to herself as to fix. As opposed to the pattern [R=T≠F], the pattern [R≠T=F] is assumed to make the bug assignment difficult as shown previously in the result of RQ1. The pattern [R≠T=F], however, would make bug fixing itself faster if the individual (B) has a good understanding of the bug based on the report by the individual (A); otherwise it would make bug fixing difficult because the individual (B) has to spend the time to investigate the bug.

Pattern #4: Reporter≠Triager≠Fixer [R≠T≠F]

In this pattern, three different individuals play each of the roles in the process. This pattern is assumed to make both the bug assignment and bug fixing most difficult because the mismatches of knowledge and skills between reporters and

Table III
DATA SETS COLLECTED FROM THE ECLIPSE PLATFORM AND JDT PROJECTS

project	study period	# of bug reports in total	# of fixed bug reports with reassignment	# of fixed bug reports without reassignment	# of reporters	# of triagers	# of fixers
Platform	Jan. 2007 - Dec. 2009	21,308	8,434	4,133	811	54	85
JDT	Jan. 2007 - Dec. 2009	8,110	3,343	1,657	369	23	33

Table IV
RATIO OF BUG MANAGEMENT PATTERNS IN PLATFORM AND JDT

project	pattern	ratio
Platform	R=T=F	17% (719/4,133)
	R=T≠F	7% (281/4,133)
	R≠T=F	38% (1,575/4,133)
	R≠T≠F	38% (1,558/4,133)
JDT	R=T=F	14% (241/1,657)
	R=T≠F	13% (211/1,657)
	R≠T=F	35% (576/1,657)
	R≠T≠F	38% (629/1,657)

triagers and between triagers and fixers would be larger than that of the other patterns. The pattern [R≠T≠F] depends heavily on the collective and collaborative efforts in an open source project. In practice, it is well known that open source systems are created by a considerable amount of efforts from a minority of individuals [23]. In this paper we wish to confirm the difficulty of "team work" in the bug management process and to find hints to overcome it.

V. A CASE STUDY ON BUG MANAGEMENT PATTERNS

This section describes our analysis on the bug management patterns in Eclipse projects. We use the same data set used in our pilot study.

A. Data sets

Table III shows the basic statistics of the bug report data sets we collected. The bugs were reported and fixed from 2007 to 2009. We did not use any data of bug reports which had been still OPEN (= not fixed) during the studied period. We got fixed bugs with reassignments (8,434 in Platform and 3,343 in JDT) and fixed bugs without reassignments (4,133 and 1,657), though we did not use fixed bugs with reassignment for our case study due to the reason mentioned in Section III. We can see that about one-third of reported bugs were fixed without reassignment. We can also see that the number of bug reporters is about 10-15 times larger than the number of triagers and fixers.

Table IV shows the ratio of the bug management patterns in each project. [R≠T=F] and [R≠T≠F] are the dominant bug management patterns in both projects. Surprisingly, both patterns exhibit similar ratio around 35–38%. Only 17% in Platform and 14% in JDT are fixed through [R=T=F]. We expect the pattern [R=T=F] is the most optimal circumstance of the bug management process as there is no need for knowledge sharing between the involved individuals.

B. Bug management patterns and time for task assignment

Using our patterns, we aim to answer the following research question.

RQ3: *How do the bug management patterns impact the time to complete bug assignments?*

Motivation: As we showed in Table I, the time to assign a bug fixing task ($T_{assignment}$) depended on whether a bug reporter played the role of a triager. However, in our pilot study we did not consider if the triager also fixed the bug (i.e., we did not distinguish between [R=T=F] and [R=T≠F] and between [R≠T=F] and [R≠T≠F].)

Approach: Similar to our pilot study, we measure the average and median time (days) of $T_{assignment}$ in each pattern and test the differences of $T_{assignment}$ between the patterns using the Mann–Whitney U test ($\alpha = 0.05$). We also calculate the cumulative percentage of assigned bug reports to look at how bug report assignments in each pattern are completed over time.

Results: Table V and Figure 3 are the analysis result of the relationship between the bug management patterns and $T_{assignment}$. As we expected before the analysis, [R=T=F] showed the best performance in assigning bug fixing tasks in the Platform project, but the best pattern in JDT was [R=T≠F]. However we could not confirm the statistically significance of the difference between [R=T=F] and [R=T≠F] in JDT.

Against our expectations, there were no differences between [R≠T=F] and [R≠T≠F] in the Platform project. Furthermore we could confirm the statistically significant difference in JDT, surprisingly this means that the performance in [R≠T≠F] was significantly superior to that in [R≠T=F]. Figure 4 shows the required days to assign all the bug fixing tasks to individuals in Platform and JDT. In fact, [R≠T≠F] did not show the worst performance, rather [R≠T=F] is around 48%–58% slower than [R≠T≠F] on average days.

This result is also complemented by Table VI which shows the days required to assign the majority (80%) of bug reports to individuals in each pattern. We suspect that this phenomena likely happened when a triager in [R≠T=F] tried to ask a fixer to take a bug fixing task but cannot get any response from the fixer (or cannot find an adequate fixer), then he or she finally assigns the task to himself or herself.

> The bug management patterns impact the performance of assigning a bug-fixing task.

Table V
RESULTS OF THE ANALYSIS ON THE BUG MANAGEMENT PATTERNS AND $T_{assignment}$ IN THE ECLIPSE PLATFORM AND JDT PROJECTS

project	Assignment pattern	median days	average days	SD	max days	min days	P-value R=T≠F	P-value R≠T=F	P-value R≠T≠F
Platform	R=T=F	0.00	12.28	71.50	812.05	0.00	$< 0.01 **$	$< 0.01 **$	$< 0.01 **$
	R=T≠F	0.00	13.27	53.06	512.98	0.00	-	$< 0.01 **$	$< 0.01 **$
	R≠T=F	0.48	20.41	82.35	842.93	0.00	-	-	0.61
	R≠T≠F	**0.53**	9.99	51.00	842.93	0.00	-	-	-
JDT	R=T=F	**0.00**	14.33	75.48	713.66	0.00	0.59	$< 0.01 **$	$< 0.01 **$
	R=T≠F	**0.00**	**6.32**	25.18	237.06	0.00	-	$< 0.01 **$	$< 0.01 **$
	R≠T=F	0.69	25.45	93.67	927.05	0.00	-	-	$< 0.01 **$
	R≠T≠F	0.32	14.96	63.79	638.23	0.00	-	-	-

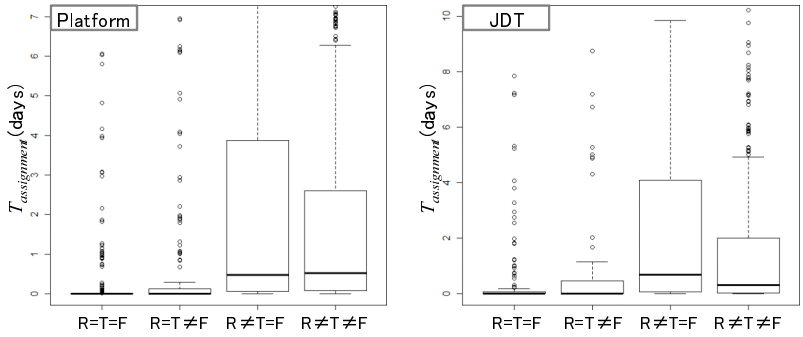

Figure 3. Boxplot of the bug management patterns and $T_{assignment}$ in Platform and JDT

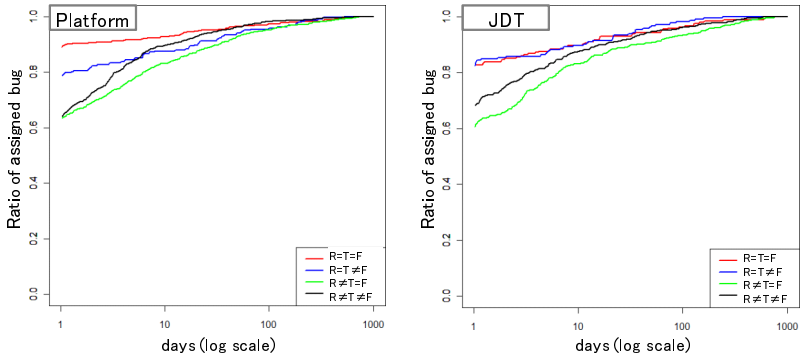

Figure 4. Required days to assign bug reports in Platform and JDT

Table VI
REQUIRED DAYS TO ASSIGN THE MAJORITY (80%) OF BUG REPORTS

	R=T=F	R=T≠F	R≠T=F	R≠T≠F
Platform	0.02	1.23	6.15	3.45
JDT	0.56	0.82	6.34	3.38

C. Bug management patterns and time for bug fixing

Next, we aim to answer the following research question.
RQ4: *How do the bug management patterns impact the time to fix bugs?*
Motivation: In RQ2, we did not consider who reports a bug when studying T_{fix}. If a bug reporter is an end-user, he might not produce a good bug report to expedite the bug fixing. We expect that using our bug management patterns we would bring us better understanding of the bug management process in Eclipse projects.
Approach: Same as our pilot study, we measure the average and median time (days) of T_{fix} in each pattern and then test the differences between the patterns using the Mann–Whitney U test ($\alpha = 0.05$). We also calculate the cumulative percentage of fixed bugs over time.
Results: Table VII and Figure 5 show the analysis result of the relationship between the bug management patterns and T_{fix}. We can see that there are the differences at 1% significant level between [R≠T≠F] and other patterns in Platform and JDT. This implies that "team work" in the bug management process is much difficult due to communication overhead and/or miscommunication among stakeholders.

Figure 6 shows the days required to fix all the bugs in the

Table VII
RESULTS OF THE ANALYSIS ON THE BUG MANAGEMENT PATTERNS AND T_{fix} IN THE ECLIPSE PLATFORM AND JDT PROJECTS

project	Assignment pattern	median days	average days	SD	max days	min days	P-value R=T≠F	P-value R≠T=F	P-value R≠T≠F
Platform	R=T=F	1.00	18.17	50.75	434.80	0.00	$< 0.01 **$	0.11	$< 0.01 **$
	R=T≠F	0.10	21.28	80.64	889.05	0.00	-	$< 0.01 **$	$< 0.01 **$
	R≠T=F	1.30	25.41	70.86	775.96	0.00	-	-	$< 0.01 **$
	R≠T≠F	**7.13**	51.56	115.19	988.21	0.00	-	-	-
JDT	R=T=F	0.64	12.91	36.92	377.84	0.00	0.19	0.32	$< 0.01 **$
	R=T≠F	0.83	13.01	32.98	281.03	0.00	-	0.49	$< 0.01 **$
	R≠T=F	0.79	12.48	45.23	583.11	0.00	-	0.49	$< 0.01 **$
	R≠T≠F	**2.11**	26.02	69.71	705.90	0.00	-	-	-

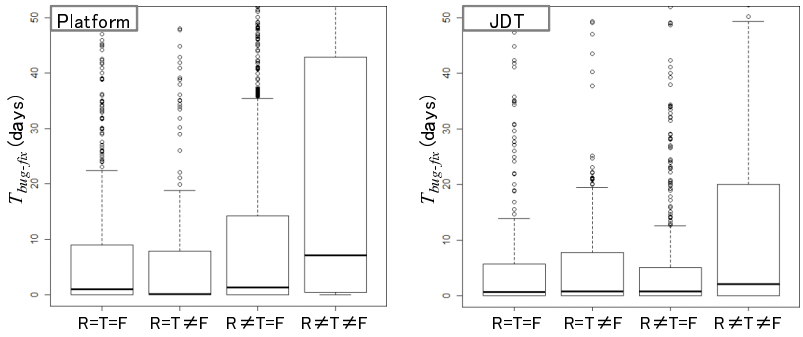

Figure 5. Boxplot of the bug management patterns and T_{fix} in Platform and JDT

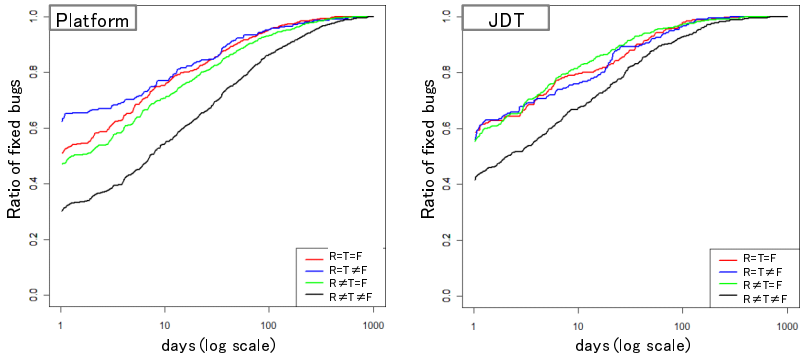

Figure 6. Required days to fix bugs in Platform and JDT

Table VIII
REQUIRED DAYS TO FIX THE MAJORITY (80%) OF BUGS

	R=T=F	R=T≠F	R≠T=F	R≠T≠F
Platform	14.87	12.94	22.92	63.13
JDT	10.79	16.30	8.00	18.19

projects. In the both projects, the bug fixing performance in the pattern [R≠T≠F] is apparently the worst. This result is also complemented by Table VIII which shows the days required to fix the majority (80%) of bugs in each pattern.

From Figure 6 and Table VIII, we see that surprisingly the pattern [R≠T=F] in the JDT project has the best performance. We inspected our data and found that several "super" developers from IBM dedicate considerable efforts to bug fixing in the pattern [R≠T=F]. As we described in RQ3, [R≠T=F] is the worst for $T_{assignment}$. The triagers in JDT might assign tasks to themselves to fix bugs soon.

The bug management patterns impact the speed of bug fixing. The pattern [R≠T≠F] makes the performance worst even if we do not account for bugs that are reassigned to more than one developer.

VI. DISCUSSIONS

We now discuss our findings and their impact on overcoming the difficulty of the bug management process. We also presents the threats to validity of our study.

A. Summary of our findings

Through our research questions RQ 1–4, we can confirm the impact of the individuals involved in bug fixing. In summary, we observe the following consequences of our bug management patterns on the bug management process.

- When a triager makes a bug report as a reporter, the time to assign a bug fixing task to a fixer is 17–47% faster than when a regular reporter does so [RQ1].
- However, surprisingly when the triager assigns a bug fixing task to himself, he needs 48%–58% longer time for the bug assignment than when he assigns it to other developers [RQ3].
- When a triager assigns a bug fixing task to himself, he can fix the bug around two times faster than when he assigns it to other developers [RQ2].
- Conclusively, the pattern [R≠T≠F] exhibits the worst performance in bug fixing (at least two times slower than the other patterns on average days) [RQ4].

Our results highlight the importance of social and knowledge sharing factors on the bug management process. We note the need for better tools to facilitate the knowledge sharing and communication between project personnel. We also believe that researchers would benefit from integrating such social knowledge (i.e., roles and individuals) in their software quality models, in particular, models to predict the fix time of bugs.

B. The impact of discussions among developers

As we expected, the pattern [R≠T≠F] leads to the worst performance in bug fixing (T_{fix}). However, we noticed that the boxplot of [R≠T≠F] in Figure 5 had the widest distribution. This implies that in some cases the pattern [R≠T≠F] works better than other patterns. We sought to explore this closely to overcome the challenge associated with [R≠T≠F]. We manually checked 100 randomly selected bug reports from the whole bug report data (1,558 for Eclipse and 629 for JDT) in that pattern. We then compared quickly-fixed bug reports with slowly-fixed bug ones. We found that discussions about bugs before bug report assignment made a difference in the bug-fixing performance.

Figure 7 shows the relationship between the bug-fixing time and the number of discussions comments before the bug assignment in the Platform project. We see T_{fix} is gradually reduced as comments (discussions about a bug in Bugzilla) increase. We also manually read the bug reports and found that developers including triagers discussed how to reproduce and fix the bugs, and who would be the most appropriate to fix the bugs.

From this finding, we can say that project personnel can improve their efficiency through better communication about bugs before assigning them. Discussions among developers are still very important not only for efficient bug-fixing but also for preventing reassignments, although the time to assign bug fixing tasks to appropriate fixers ($T_{assignment}$) might be slightly prolonged due to the discussions.

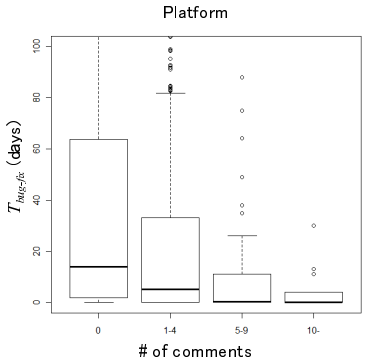

Figure 7. T_{fix} and discussions before the bug report assignment in the pattern [R≠T≠F] of the Platform project

C. Other factors that would impact the time to fix bugs

Although in this paper we focused on the bug management patterns, we can also consider many other factors that would impact the time to fix bugs: *bug (content of a bug report), day and time, stakeholders)* [10], [23]–[25] and so on. For instance, a bug report with high severity might be fixed sooner than other bugs. We extracted metrics associated with such factors from our dataset as listed in Table IX.

In order to demonstrate the impact of the factors on time to fix bugs, we created a prediction model based on logistic regression to quantify the relationships between the factors. The procedure for creating the prediction model is as follows:

1) Top five categories frequently appeared in each independent variables with nominal scale are transformed into dummy variables since the existence of many dummy variables are likely to create the issue of multicollinearity. Note that we also calculated VIF (Variance Inflation Factor) to confirm the issue of multicollinearity between any two independent variables and eliminated variables if the VIF exeeded 10.
2) The value of the dependent variable is set as 1 if bug fixing (T_{fix}) is completed within the three kinds of specified period of time (in a day, in a week, and in a month) and otherwise it is set as 0 (i.e., T_{fix} exeeds the specified period of time.)
3) We divide our dataset used in this study into nine fit data ($fit_1, \cdots, fit_9$) and one test data ($test$).
4) Using the nine fit data and a logistic regression model, we build a model to predict whether T_{fix} is completed in each specified period of time.
5) Using the test data, we calculate the prediction accuracy for each specified period of time.
6) After repeating Step 3) – Step 5) ten times, we calculate the average prediction accuracy.

Table IX
METRICS ASSOCIATED WITH FACTORS THAT WOULD AFFECT TIME TO FIX BUGS

factor	metrics (variable name)	scale	descriptions
bug	Component	nominal	component name specified in the bug report
	Priority	nominal	priority for fixing the bug
	Severity	nominal	severity of the reported bug
	Milestone	nominal	whether or not a milestone is specified in the bug report
	DescriptionWords	interval	number of words in "Description" in the bug report
	CommentsCount	interval	number of comments in the bug report
	CommentsWords	interval	number of words in comments
	AttachmentsCount	interval	number of attachments (e.g., patches and screen shots)
	DependsOnCount	interval	number of bugs which must be resolved before the reported bug
	BlocksCount	interval	number of other bugs which are blocked by the reported bug
	CCCount	interval	number of users who might be interested in the bug report
day and time	AssignTime	interval	time to assign the bug fixing task to a developer (i.e., $T_{assignment}$)
	AssignedMonth	interval	month in which the bug fixing task was assigned to a developer
	AssignedDay	interval	day in which the bug fixing task was assigned to a developer
	AssignedWeekEnd	nominal	whether or not the bug fixing task was assigned in the weekend
stakeholder	Reporter	nominal	email address of the reporter (who reports the bug)
	Triager	nominal	email address of the triager (who triages the bug)
	Fixer	nominal	email address of the fixer (who resolves the bug)
	Pattern	nominal	bug management pattern used in fixing the bug (**main scope of this paper**)

Table X
PREDICTION ACCURACY OF OUR PREDICTION MODEL FOR PLATFORM

	prediction period	precision	recall	F1-value
prediction accuracy of our logistic regression model	in a day	68.14	38.22	48.97
	in a weak	67.90	76.66	72.02
	in a month	76.67	98.77	86.33
improvement rate against random prediction	in a day	66.68%	-6.50%	19.80%
	in a week	14.22%	28.95%	21.14%
	in a month	2.35%	31.86%	15.24%

Table XI
IMPORTANCE OF EACH INDEPENDENT VARIABLE IN OUR PREDICTION MODEL FOR PLATFORM

factor	metrics (variable name)	deviance residuals <a day	<a week	<a month
bug	Component	**263.06**	**177.64**	**132.95**
	Priority	9.22	5.98	1.41
	Severity	1.38	3.70	3.88
	Milestone	6.31	5.04	4.96
	DescriptionWords	0.42	1.67	2.54
	CommentsCount	3.39	11.84	16.32
	CommentsWords	4.56	1.52	1.17
	AttachmentsCount	3.84	0.04	0.99
	DependsOnCount	6.26	2.72	0.67
	BlocksCount	0.70	0.61	1.40
	CCCount	12.08	8.33	5.04
day and time	AssignTime	4.71	11.54	11.48
	AssignedMonth	9.87	9.08	22.23
	AssignedDay	0.61	1.90	0.05
	AssignedWeekEnd	0.10	0.09	1.75
stakeholder	Reporter	7.23	14.63	22.22
	Triager	8.60	7.60	9.65
	Fixer	**76.26**	**72.27**	**53.10**
	Pattern	**154.49**	**123.81**	**89.50**

Table X shows the calculated results of the prediction accuracy for each specified period. It also shows the improvement rate against a prediction result which was calculated 1,000 times by randomly selecting one independent variable. Almost all results with the three evaluation measures (precision, recall and F1-value) tend to be better when predicting a longer period of time. All the F1-values of our results also perform the result which is predicted by using randomly selected independent variable.

We analyzed which independent variable contributed to our prediction model building. We used deviance residuals as the measures to check the model fit. If deviance residual value for one independent variable is larger than others, it means that the model fit well due to the independent variable. Table XI is the results of deviance residuals. We can see that Component (i.e., which component is fixed) is the most important factor to predict T_{fix} in each period of time. Pattern (i.e., the bug management patterns) and Fixer (i.e., who fixes the bug) are second and third important factors respectively. The results of Component and Fixer are consistent with the existing studies [3], [10], [25]. Contrary to our expectation, Triager (i.e., who triages the bug report) does not contribute to the prediction model for Platform. From these results, we can conclude that the bug management pattern is an unignorable, important factor which has an impact on the time to fix bugs and should be carefully managed by project personnel.

D. Threats to Validity

In this study we only used three years (from 2007 to 2009) bug report data without reassignments in the Eclipse projects to obtain a clear understanding of the bug management process. Such data selection (including the limited term of data and excluding reassignments) might bring bias [26] 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 studied only the two open source projects: Eclipse

Platform and JDT projects. These projects are large scale, successful projects and so they have sufficient bug report data to validate our patterns. 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. So, our findings in this study might not be applicable to any other open source projects. In addition, the user base of the Eclipse products (i.e., the knowledge and skill of the individuals involved in the Eclipse projects) is different from that of other products such as the Mozilla products (e.g., Firefox and Thunderbird). We need to apply our patterns to other projects in the future to verify the generality of our findings.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we explored the bug management process from a social perspective. In particular, we studied the impact of the roles associated with the process and the different individuals playing these roles. We found that:

1) There is a need for better ways to communicate and share knowledge between the different individuals. In cases where all roles where played by different individuals, the efficiency of the bug fixing was negatively impacted.
2) One project (i.e., JDT) put careful attention on prioritizing the fixing of bugs that have been slowed down due to inefficiencies early on in the bug management process (i.e., bug assignment taking too long).
3) Communication appears to have a positive impact on speeding up bug fixing time even when every role is played by different individuals.

In future work, we wish to investigate whether our findings hold when we integrate other confounding factors (e.g., complexity of the bug and its associated fix). In particular, we wish to explore whether social (i.e., our patterns) have a stronger impact on the bug management process than technical factors.

ACKNOWLEDGMENTS

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).

REFERENCES

[1] "Bugzilla," http://www.bugzilla.org/.
[2] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with bug tossing graphs," in *Proc. of ESEC/FSE'09*, 2009, pp. 111–120.
[3] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *Proc. of ICSE'06*, 2006, pp. 361–370.
[4] D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization," in *Proc. of SEKE'04*, 2004, pp. 92–97.
[5] C. Sun, D. Lo, X. Wang, J. Jiang, and S.-C. Khoo, "A discriminative model approach for accurate duplicate bug report retrieval," in *Proc. of ICSE'10*, 2010, pp. 45–54.
[6] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing," in *Proc. of ICSE'07*, 2007, pp. 499–510.
[7] 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 ICSE'08*, 2008, pp. 461–470.
[8] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy, ""not my bug!" and other reasons for software bug report reassignments," in *Proc. of CSCW'11*, 2011, pp. 395–404.
[9] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy, "Characterizing and predicting which bugs get fixed: an empirical study of microsoft windows," in *Proc. of ICSE'10*, 2010, pp. 495–504. [Online]. Available: http://doi.acm.org/10.1145/1806799.1806871
[10] E. Shihab, A. Ihara, Y. Kamei, W. Ibrahim, M. Ohira, B. Adams, A. Hassan, and K. Matsumoto, "Predicting reopened bugs: A case study on the eclipse project," in *Proc. of WCRE'10*, 2010, pp. 249–258.
[11] N. Bettenburg, S. Just, A. Schröter, C. Weiß, R. Premraj, and T. Zimmermann, "Quality of bug reports in eclipse," in *Proc. of Eclipse'07*, 2007, pp. 21–25.
[12] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?" in *Proc. of FSE-16*, 2008, pp. 308–318.
[13] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Extracting structural information from bug reports," in *Proc. of MSR'08*, 2008, pp. 27–30.
[14] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: improving cooperation between developers and users," in *Proc. of CSCW'10*, 2010, pp. 301–310.
[15] P. Hooimeijer and W. Weimer, "Modeling bug report quality," in *Proc. of ASE'07*, 2007, pp. 34–43.
[16] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?" *IEEE Trans. on Software Engineering (TSE)*, vol. 36, no. 5, pp. 618–643, 2010.
[17] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Duplicate bug reports considered harmful ... really?" in *Proc. of ICSM'08*, 282008-oct.4 2008, pp. 337 –345.
[18] P. Bhattacharya and I. Neamtiu, "Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging," in *Proc. of ICSM'10*, 2010, pp. 1–10.
[19] J. Anvik, L. Hiew, and G. C. Murphy, "Coping with an open bug repository," in *Proc. of Eclipse'05*, 2005, pp. 35–39.
[20] D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug reports using a vocabulary-based expertise model of developers," in *Proc. of MSR'09*, 2009, pp. 131–140.
[21] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *Proc. of MSR'06*, 2006, pp. 137–143.
[22] C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, and G. Hsu, "Detecting patch submission and acceptance in oss projects," in *Proc. of MSR'07*, 2007, p. 26.
[23] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and mozilla," *ACM Trans. on Software Engineering and Methodology (TOSEM)*, vol. 11, no. 3, pp. 309–346, 2002.
[24] I. Herraiz, D. M. German, J. M. Gonzales-Barahona, and G. Robles, "Towards a simplification of the bug report form in eclipse," in *Proc. of MSR'08*, 2008, pp. 145–148.
[25] M. D. Syer, B. Adams, Y. Zou, and A. E. Hassan, "Studying the fix-time for bugs in large open source projects," in *Proc. of PROMISE'11*, 2011.
[26] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu, "Fair and balanced?: bias in bug-fix datasets," in *Proc. of ESEC/FSE '09*, 2009, pp. 121–130.