International Journal on Advanced Science Engineering Information Technology

Vol.10 (2020) No. 1
ISSN: 2088-5334

How Usability Defects Defer from Non-Usability Defects? : A Case Study on Open Source Projects

Nor Shahida Mohamad Yusop[#], John Grundy[*], Jean-Guy Schneider[+1], Rajesh Vasa[+2]

[#]*Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia*
E-mail: nor_shahida@tmsk.uitm.edu.my

[*]*Faculty of Information Technology, Monash University, Melbourne, Australia*
E-mail: john.grundy@monash.edu

[+]*Faculty of Science, Engineering, and Built Environment, Deakin University, Melbourne, Australia*
E-mail: [1]jeanguy.schneider; [2]rajesh.vasa}@deakin.edu.au

***Abstract*— Usability is one of the software qualities attributes that is subjective and often considered as a less critical defect to be fixed. One of the reasons was due to the vague defect descriptions that could not convince developers about the validity of usability issues. Producing a comprehensive usability defect description can be a challenging task, especially in reporting relevant and important information. Prior research in improving defect report comprehension has often focused on defects in general or studied various aspects of software quality improvement such as triaging defect reports, metrics and predictions, automatic defect detection and fixing. In this paper, we studied 2241 usability and non-usability defects from three open-source projects - Mozilla Thunderbird, Firefox for Android, and Eclipse Platform. We examined the presence of eight defect attributes - steps to reproduce, impact, software context, expected output, actual output, assume cause, solution proposal, and supplementary information, and used various statistical tests to answer the research questions. In general, we found that usability defects are resolved slower than non-usability defects, even for non-usability defect reports that have less information. In terms of defect report content, usability defects often contain output details and software context while non-usability defects are preferably explained using supplementary information, such as stack traces and error logs. Our research findings extend the body of knowledge of software defect reporting, especially in understanding the characteristics of usability defects. The promising results also may be valuable to improve software development practitioners' practice.**

***Keywords*— defect report; open-source; software repository mining; software defect repository; usability defects.**

I. INTRODUCTION

Usability is a software quality characteristic that measures the understandability, learnability, operability and attractiveness of the software products. The lack of a systematic approach towards human-centered design in software development is seen as an obstacle in producing high usable software [1]. An effort to increase software usability is through consistent usability testing and user experience evaluation during the development cycle [2]. However, measuring usability defects is often difficult due to their subjective evaluation [3]. For example, one user might find a graphical icon in a user interface as annoying but it is not so for others. As a result, there is often a disagreement between the reporters and software developers that subsequently carry out the defect resolution. In fact, in the context of software development, usability defects are said to receive less attention from software developers [4] and due to an abundant amount of information developers have difficulties in identifying vital information to understand the defects [5].

We believe that usability defects need stronger evidence to convince software developers that the issue reported is indeed a real defect and should be treated as equally important as other high-risk defects, such as security and performance defects. In open-source software projects, users – who can be software developers or end-users or both - can directly report usability defects through defect reporting tools. However, the generic unstructured free-text defect form in Bugzilla, JIRA and GitHub, for example, is often not very helpful for capturing valuable information for usability defects [6]. This is because aspects such as likely difficulties, impact, user's feelings, emotion and "struggling" with the interface are naturally subjective and depend heavily on human judgment. Without specific prompts for this usability

defect-specific information, it can be difficult for non-technical users to provide such information.

In this paper, we present an empirical investigation of 377 usability defects and 1864 non-usability defects in Mozilla Thunderbird, Firefox for Android and Eclipse Platform open-source project to compare to what extent these non-usability defects are described and treated by these open source communities. In this research, we picked performance-related defects as a comparison benchmark since it is a non-usability type of defect that is commonly reported in open source projects. In addition to previous studies that investigate the characteristics of defects [7], [8] using predefined defect data (i.e., product, component, version) and comments, we manually examined the textual descriptions of defect reports to find out how different types of defects influence the defect resolution time and severity, and the way defect is described. In particular, this paper addresses the following four research questions (RQ):

1) RQ1: Do different defect types and projects influence the ways defects are described? The defect description is different between usability and performance defects, and across the three open-source projects. We found usability defects reported for open source projects usually contain output details, and software context, while non-usability defects are preferably explained using supplementary information.

2) RQ2: Does the presence of specific defect attributes influence the defect resolution time? The presence of *impact expected outcome* and *actual output* will eventually reduce the defect resolution time of both usability and non-usability defects. Also, when the usability and non-usability defect reports contain *supplementary information*, the total resolution time and actual fix time can be reduced.

3) RQ3: Do different defect types and projects influence the defect resolution time, wait time and actual fix time? The total resolution time, wait time and actual fix time differed significantly between the different types of defects and projects. We found that non-usability defects take less time to correct than usability defects. In terms of projects, the Thunderbird project took the largest time to resolve usability defects, followed by the Eclipse platform and Firefox for Android.

4) RQ4: Is there any relationship between defect types and defect severity? The relationship between defect types and severity is moderately significant. Usability defects are more dominantly rated low-severity compared to non-usability defects.

5) RQ5: How fast do open source communities respond to usability defects in comparison to non-usability defects? On average, we found non-usability defects are responded to 5 times faster than usability defects in Mozilla Thunderbird, and 3 times faster than usability defects in Firefox for Android and Eclipse Platform.

6) RQ6: Do different defect types and projects affect the number of comments? Defect types do not statistically affect the number of comments. However, the number of comments differs between projects. The significant difference of total comments received among the three projects suggested that the contributor community of the Thunderbird project is the most active, followed by Firefox for Android and Eclipse Platform. The rest of the paper is organized as follows. In Section 2, our experimental method is presented. Defect sources, data extraction, and statistical analysis used are discussed. In Section 3, the study results are presented and discussed. Section 6 concludes the paper with future works.

II. Materials and Method

A. Defect Sources

In this research, we studied the Mozilla Thunderbird, Firefox for Android, and Eclipse platform projects. Across the three projects, only 23373 defect reports are available to download from Bugzilla defect tracking system. For usability defects, we looked for Bugzilla usability keywords as listed in Table I, while for non-usability defects, we decided to include defects that were tagged with performance-related issues such as perf, crash, top crash, hang, intermittent-failure. As shown in Table I, 1206 and 6345 of downloaded defect reports were tagged with usability and performance-related keywords, respectively. However, in our study, we only included 2241 usability and non-usability defect reports that were resolved as FIXED (a defect has been fixed by developers, or being fixed by another defect fix). We limit our analysis to FIXED defect reports in order to reduce selection bias, as the software developers already completed the resolution process and have reach agreement on the actual types of defects. We extracted sample defect reports for each project using the methods described in [9].

TABLE I
DATASETS STUDIED

Project	Types	Total	Other resolution	Resolved/ Verified						
				Fixed	Duplicate	Incomplete	Invalid	Wontfix	Worksforme	Expired
Mozilla Thunderbird	Usability	384	185	**88**	64	4	9	16	17	1
	Non-usability	2223	212	**349**	406	453	158	7	591	47
Firefox for Android	Usability	292	62	**101**	59	3	11	36	20	0
	Non-usability	2855	643	**822**	422	44	43	66	815	0
Eclipse Platform	Usability	530	78	**188**	46	-	68	103	47	-
	Non-usability	1267	200	**693**	98	-	74	108	94	-
Total		7551	1380	**2241**	1095	504	363	336	1584	48

Other resolution – New, unconfirmed, assigned, and reopened
Non-usability - perf, crash, topcrash, hang, intermittent-failure
Usability – ue, uiwanted, useless-UI, ux-affordance, ux-consistency, ux-control, ux-discovery, ux-efficiency, ux-error-prevention, ux-error-recovery, ux-implementation, ux-interruption, ux-jargon, ux-minimalism, ux-mode-error, ux-natural-mapping, ux-tone, ux-trust, ux-undo, ux-userfeddback,

TABLE II
RESEARCH QUESTIONS AND CORRESPONDING STATISTICAL TEST

Research Questions	Statistics
RQ1: Do different defect types and projects influence the ways defects are described?	Frequency, Shapiro-Wilk test, Chi-Square test
RQ2: Does the presence of certain defect attributes influence the defect resolution time?	Frequency, Shapiro-Wilk test, Spearman's rank correlation
RQ3: Do different defect types and projects influence the defect resolution time, wait time and actual fix time?	Frequency, Shapiro-Wilk test, Kruskal–Wallis H test
RQ4: Is there any relationship between defect types and defect severity?	Frequency, Shapiro-Wilk test, Chi-Square test
RQ5: How fast open source communities respond to usability defects in comparison to performance defects?	Frequency, Shapiro-Wilk test, Kruskal–Wallis H test
RQ6: Do different defect types and projects affect the number of comments?	Frequency, Shapiro-Wilk test, Kruskal–Wallis H test

B. Research Questions and Related Statistics Computation

We performed different statistical analyses for different research questions, depending on the types of data and analysis we wanted to use. Since, the *Total Resolution Time (TRT), Wait Time (WT), Actual Fix Time (AFT),* and *First Comment Response Time (FCRT)* were not normally distributed (as assessed with Shapiro-Wilk's test for normality), non-parametric statistics were used for analysis. Table II presents the list of research questions and their respective statistical procedures applied. We used SPSS (version 23) to conduct all tests.

We used Chi-square analysis to answer RQ1 and RQ4. We consider textual information - steps to reproduce (STR), impact (IMP), software context (SC), expected output (EO), actual output (AO), assumed cause (AC), solution proposal (SP) and supplementary information (SI) as dependent variables, measured on nominal scale and defect types as independent variable. Alternately, in Spearman's Rho correlation analysis (to answer RQ2) STR, IMP, SC, EO, AO, AC, SP, and SI were used as the independent variable while TRT, WT, and AFT were used as the dependent variable. For RQ3, we used Kruskal-Wallis H Test to compare whether *Total Resolution Time (TRT), Wait Time (WT)* and *Actual Fix Time (AFT),* measured on a continuous scale (in days), differed based on defect types and projects (nominal data). We consider *TRT, WT* and *AFT* as the dependent variable, and defect types (usability or non-usability defect) and project types (Mozilla Thunderbird, Firefox for Android and Eclipse Platform project) as independent variables. We also consider the *First Comment Response Time (FCRT)* and *Total Comments (TC)* as the dependent variable in examining RQ5 and RQ6. Since the distribution of the scores for each group in defect types and project types are identical, we used the medians value of the dependent variables to compare the characteristics of different groups of the independent variable.

C. Information Extraction

A software defect reports have many attributes, some of which are filled at the time of reporting and others are filled during the fixing process [10]. In our study, we only focused on the attributes supplied during the initial reporting of a defect, not the subsequent discussion about the problem and its possible solution in the comments sections. Our primary textual analysis is focused on the defect report *title, description* and *attachment* fields. Since the existing Bugzilla defect report template was in unstructured plain text, we were unable to automatically extract and assess the presence of *STR, IMP, SC, EO, AO, AC,* SP, and *SI*. Thus, we manually read all the 2241 defect reports and used the criteria suggested by Capra [11].

The first author read all the 2241 defect descriptions. This process requires the author to (1) interpret the reported problem, (2) classify the information into one of the predefined attributes, and (3) give score if the information is presented - 1 implies that the "information exists", and 0 implies that the "information does not exist." In order to validate the rating process and minimize the misclassification error, a random sample of 10 defect reports was rated by the second author and then compared in a review meeting. Whenever the rating differed, such differences were discussed until consensus reached.

For *IMP, AC,* and *SP* we assessed the presence of this information usin3 the following criteria:

- *AC* - defect report number, in which the reporter felt the current issues were likely due to the previous fixed.
- *IMP* – user difficulty, number of reproducibility's, high *numbers* of users encountered the same problem and severity.
- *SP*: justification of the proposed solution or fragmental/ modification of affected code/ patch description on how to fix the problem.

III. RESULTS AND DISCUSSION

Each subsection below discusses the three research questions that we studied using Mozilla Thunderbird, Firefox for Android, and Eclipse Platform data. For each question, we present the results and a discussion of our findings.

A. The Essence of Defect Report Content

To better understand how the usability defects are described in comparison to other non-usability defects, we examined the following research question: *RQ1: Do different defect types and projects influence the ways defects are described?"*

Table III reveals several observations from the Chi-square test to answer RQ1. First, a large fraction of usability defects in Mozilla Thunderbird, Firefox for Android and Eclipse Platform contains *SC, AO,* and *EO*. Second, Only a few non-usability defects contain *EO* (19.5%, 9.5%, 30.9%, for Mozilla Thunderbird, Firefox for Android, and Eclipse Platform, respectively). Third, *SI* is mostly attached with non-usability defects (67.0%, 89.4%, 15.0% for Mozilla Thunder, Firefox for Android, and Eclipse Platform, respectively) than usability defects (13.6.0%, 26.7%, 5.3%

for Mozilla Thunderbird, Firefox for Android, and Eclipse Platform, respectively). Fourth, *IMP*, *AC*, and *SP* are rarely reported for both usability and non-usability defects across projects. Fifth, for the three projects, the *STR* is more often described in the usability defect reports than in non-usability defects. At a significant level p=0.05, we found the relationship between defect types and the presence of defect information are significant for all attributes in the three projects, accept for *STR* (df=1, p=0.420) and *SC* (df=1, p=0.162) in Firefox for Android, and *AO* (df=1, p=0.708) in Eclipse Platform.

TABLE III
CHI-SQUARE TEST RESULTS TO EXAMINE THE INFLUENCE OF DEFECT TYPES AND DEFECT ATTRIBUTES

Defect Report Attributes	**Mozilla Thunderbird (%)**			**Firefox for Android (%)**			**Eclipse Platform (%)**		
	Usability	***Non-usability***	***p-val***	***Usability***	***Non-usability***	***p-val***	***Usability***	***Non-usability***	***p-val***
STR	46.6	24.1	0.000	18.8	15.7	0.420	32.4	21.1	0.001
IMP	**38.6**	**15.2**	0.000	**27.8**	**2.2**	0.000	**28.2**	**7.1**	0.000
SC	**73.9**	49.0	0.000	**47.5**	54.9	0.162	**42.0**	50.4	0.042
EO	**76.1**	19.5	0.000	**52.5**	9.5	0.000	**60.6**	30.9	0.000
AO	**83.0**	45.6	0.000	**56.4**	29.8	0.000	**85.1**	84.0	0.708
AC	**2.3**	**9.2**	0.031	**2.0**	**9.7**	0.010	**2.7**	**19.5**	0.000
SP	**17.0**	**3.7**	0.000	**18.8**	**3.3**	0.000	**8.5**	**14.0**	0.046
SI	13.6	**67.0**	0.000	26.7	**89.4**	0.000	5.3	**15.0**	0.000

We have observed most open-source defect reports, regardless of different defect types, contain software context and the actual outcome, and by contrast, relatively few defect reports describe the impact, assumed cause and have a solution proposal. Our findings confirm our previous findings [12], where reporters seldom provide the assumed cause. Since existing defect report forms in Bugzilla defect repositories do not have separate fields for each of these attributes, the provision of this information relies on the reporters' ability to explain in the free format textual description.

According to software developers [12], the presence of assumed cause is much appreciated as they can directly investigate what caused the problems, rather than guessing and investigating the problems from scratch. This is evidence, where the non-usability defect reports containing assumed cause has been resolved faster than those without this information. This is likely because logs and stack traces in non-usability defect reports provide technical information to directly determine the root cause of the problems, rather than textual information that needs subjective interpretation and assumption. From our observations, we found assumed cause often originated from the changes made to the previous code. Reporters often include the defect report number of other defect reports in which they suspect the fixed causes of the current defects. For steps to reproduce, even software developers considered them to be useful for fixing software defects [12], [13], in real-world practice, however, we observed this information is much less reported for both usability and non-usability defect reports.

As expected, the attachment of snapshots, stack traces, logs, external links and explanation of recovery steps appear to be very helpful for software developers in order to fix defects, as this kind of information contains objective evidence for defects. However, supplementary information was only found for less than 15% of usability defect reports – mostly in the form of screenshots and video. A closer inspection of usability defect reports showed that many reporters like to attach screenshots, video and logs in the later comments section. This is likely since such materials are not readily available when a defect is submitted. In Bugzilla, for example, there are insufficient features to collect hypermedia data automatically, thus making it a convoluted exercise for reporters to provide such information when a defect is first reported [14]. We postulate that more convenient defect reporting tools to support usability defect reporting issues could be developed for editing attachments, capturing UI event traces, user interactions, usage context and problematic screens, and recording current user roles and tasks.

B. Usability Defect Resolution

To investigate the influences of defect resolution time, we studied three aspects: (1) the content of defect reports, (2) types of defects and projects, and (3) the relationship between types of defects and defect severity. The aforementioned are addressed n the following research questions: *"RQ2: Does the presence of certain defect attributes influence the defect resolution time?"*, *"RQ3: Do different defect types and projects influence the defect resolution time, wait time and actual fix time?"*, and *"RQ4: Is there any relationship between defect types and defect severity?"*

Defect resolution time is a time taken for actual time spent to correct a defect and the wait time [15],. In many cases, the time needed to correct a defect is short, but the resolution time is longer due to the long waiting time to find a resource and the urgency of the defect to be fixed. To examine RQ2 and RQ3, we used the following three metrics to measure defect resolution time:

- Total resolution time (*TRT*) in days – the total time taken to resolve an issue includes the actual time spent on defect resolution and the wait time. We measured the defect resolution time of each closed defect as *ResolutionTime = DateResolved – DateOpened*.
- Wait time (*WT*) in days – the time a reported defect awaits the assignment of a resource. We measured the defect wait time as the difference between the time a defect is reported and the defect get assign before they fixed, *WaitTime = DateAssigned – DateOpened.*
- Actual fix time (*AFT*) in days - the time the assigned resource takes to start working on the defect until the defect is closed as resolve, *ActualFixTime = DateResolved – DateAssigned.*

To answer RQ2, as shown in Table IV, Spearman's Rho correlation results show that at significance level p =0.05, there is a positive relationship between *TRT* and *IMP*, *EO* and *AO* for both defect types. At the same level of significance, we also see a negative relationship between *TRT*, *AFT* and *SI* for non-usability defects and negative relationships between *TRT*, *WT*, *AFT* and *SI* for usability defects. Meanwhile, the positive relationships suggest that the presence of *IMP*, *EO* and *AO* makes the resolution time slower. For non-usability defects, there are also positive relationships between *ST* and *IMP*, and between *AFT* and *STR*, *EO*, *AO* and *SP*. While for usability defects, a positive relationship between *WT* and *IMP* is also observed.

TABLE IV
SPEARMAN'S RHO CORRELATION TO INVESTIGATE THE INFLUENCE OF DEFECT INFORMATION ON TRT, WT, AND AFT

	STR	**IMP**	**SC**	**EO**	**AO**	**AC**	**SP**	**SI**
Non-usability defects								
TRT	0.092*	**0.080****	0.051*	**0.088****	**0.165****	0.026	0.044	**-0.187****
WT	0.042	**0.095****	0.041	-0.008	-0.008	-0.055*	-0.032	0.005
AFT	**0.081****	0.011	0.018	**0.094****	**0.213****	0.072*	**0.067****	**-0.224****
Usability defects								
TRT	0.120*	**0.217****	0.025	**0.142****	**0.150****	-0.026	0.003	**-0.152****
WT	0.070	**0.181****	-0.048	0.096	0.100	-0.021	0.000	**-0.135****
AFT	0.050	0.085	0.097	0.109*	0.036	0.010	0.001	**-0.135****
* Correlation is significant at the 0.05 level (2-tailed) ** Correlation is significant at the level 0.01 level (2 tailed)								

TABLE V
DISTRIBUTION OF TRT, WT, AND AFT FOR USABILITY AND NON\-USABILITY DEFECTS

Project	**TRT**				**WT**				**AFT**			
	Usability		**Non-usability**		**Usability**		**Non-usability**		**Usability**		**Non-usability**	
	Mean	***Median***	***Mean***	***Median***	***Mean***	***Median***	***Mean***	***Median***	***Mean***	***Median***	***Mean***	***Median***
Mozilla Thunderbird	717.84	423.00	96.77	12.00	560.68	248.50	69.65	4.00	157.16	22.50	27.12	1.00
Firefox for Android	204.23	51.00	51.56	10.00	153.14	26.00	38.34	3.00	51.09	11.00	13.23	3.00
Eclipse Platform	367.40	166.00	181.87	42.00	224.84	105.00	68.42	2.00	142.56	39.00	113.45	19.00

TABLE VI
CONTIGENCY TABLE. DEFECTS TYPES BY PROBABILITY OF RECEIVING DEFECT SEVERITY

Defect Type	**Defect Severity**							
	Blocker	***Critical***	***Major***	***Minor***	***Normal***	***Trivia***	***Enhancement***	***Total***
Usability	0.3 (1)	**1.9 (7)**	9.3 (35)	**65.8 (248)**	4.2 (16)	2.4 (9)	**16.2 (61)**	100.00
Non-usability	1.3 (25)	**39.8 (742)**	8.7 (163)	**47.4 (883)**	0.6 (12)	0.00 (0)	2.1 (39)	100.00
Total	1.2 (26)	33.4 (749)	8.8 (198)	50.5 (1131)	1.2 (28)	0.4 (9)	4.5 (100)	(2241)
$\chi 2$ (6) = 375.920, p=0.000,								
Cramer's V = 0.410								
Note: Number in parentheses is the frequency of defects in each of severity.								

For RQ3, we calculated the mean and median of *TRT*, *WT*, and *AFT* for both usability and non-usability defects. Table V shows the results. These numbers show that the resolution time for the usability defect is the longest. The wait time for usability defects takes the most amount of time in defect resolution for all three open-source projects. ***Considering the actual time, the developer spends to correct a defect, we find that non-usability defects take less time to correct than usability defects***. A Kruskal-Wallis H test showed that the total resolution time ($\chi 2$ (1) = 263.408, p=0.000), wait time ($\chi 2$ (1) = 273.269, p=0.000) and actual fix $\chi 2$ (1) = 69.464, p=0.000) time differed significantly between the different types of defects.

We also considered the median time to see how fast the response of different open source project communities is in terms of correcting a defect. As shown in Table V, the Thunderbird project took the largest time to resolve usability defects, followed by the Eclipse platform and Firefox for Android. In terms of actual time used by the software developer to correct usability defects, Firefox for Android shows the shortest time. We also observed that non-usability defects are resolved slower in the Eclipse Platform. ***Across the three open-source projects, Firefox for Android project takes less time to resolve usability and performance defects***. A Kruskal-Wallis H test showed a significant difference between the project types and TRT ($\chi 2$ (2) = 183.273, p=0.000), WT ($\chi 2$ (2) = 16.143, p=0.000), and AFT ($\chi 2$ (2) = 222.928, p=0.000).

For RQ4, Table VI summarizes the distribution of defects types and severity. ***It shows that usability defects are dominantly rated low-severity compared to non-usability defects***. The percentage of low-severity for usability defects is 88.6%, much larger than the 50.1% for non-usability defects. Similarly, the high-severity impacts that were rated for non-usability defects, i.e. blocker, critical and major, account for 49.8% as compared to 11.5% for usability defects, which is a considerable difference. ***Also, we found usability defects are more likely to be reported as enhancements*** (16.2%). To further understand the influence of defect types on the severity, we present the Chi-square

test results in Table VI. The relation between defect types and severity was significant, $\chi 2$ (6) = 375.92, p= 0.000 < 0.05. Cramer's V value between 0.4 and 0.6 indicates that the relationship between defect types and severity is moderate.

Fixing usability defects is often take a much longer time than fixing non-usability defects [4]. Common reasons were due to incomplete information in defect reports, low severity rating to consider usability defects as significant issues, and misunderstanding and misinterpretation about the usability issues by developers [4], [13]. In this research, we examined the influence of defect resolution time from three perspectives: (1) types of defects and projects, (2) defect severity, and (3) the presence of specific defect attributes.

From our study, we found usability defects in Mozilla Thunderbird, Firefox for Android, and Eclipse Platform take a longer time to resolve than non-usability defects. In all these projects, the waiting time for both usability defects and non-usability defects took the most significant time in the resolution process. Based on our observations on the studied defect report comments, most of the waiting time is used to reproduce the defects, discuss the rationale of the issues, propose ideas, and await the assignment of a resource. Possibly, as open-source projects involve volunteers with a variety of different commitment levels, levels of involvement and degree of technical expertise, no dedicated resources and fix-time can be assigned to work on each defect.

The other possible explanation for a long resolution time of usability defects is due to the low-severity rate. We observed about 90% of usability defects are rated as low-severity (minor, trivia, and enhancement). Within limited resources and time constraints, high-severity defects will very likely to get more attention. When software crashes or a user interface is hanging, for example, software developers will fix the issue as soon as they can so that the software is usable again. Conversely, when someone experiences difficulty to find certain menus or is confuses in performing certain tasks, that issue may not be an important issue to the software developers. The software developers may see these kinds of usability defects as a low priority issue since the software can still be used. In many cases, unconvincing issues often closed as WORKSFORME [3]. Perhaps, reporters should develop a strategy to write better usability defect descriptions in more convincing and informative ways, so that software developers can treat usability defects as important as functional defects. We discuss this further in the next section.

In contrast to our expectation, the presence of textual information such as actual output expected output and impact do not appear to speed up the defect resolution process. Similarly, previous studies [16], [17] also found that there is no significant influence between the number of defect attributes with the resolution time. Karim et al. [17], for example, who studied high-impact defects (i.e., security, performance, breakage, and dormant defects) found there is no stable relationship exists between provided defect attribute and the defect fixing time even though defect fixing time can be reduced when defect report contains at least four main attributes. One possible reason for that may be due to "reporters' effect" as they do not know what to describe for each textual defect attribute in the defect form and how to explain them efficiently, especially for non-technical users.

While it is quite easy to collect information related to non-usability defects, it is usually much more difficult to foresee information needs for usability defects. From our observations, most of the defect reports do not follow the defect template and do not have a sufficient level of detail to explain the subjective nature of usability defects. For example, the report for defect #718960 in Firefox for Android reports:

> *"The only way I found to show the "Add to Home Screen" functionality is to long-tap on an awesome bar list item. I'm not sure this is very discoverable. Moreover, with the main screen being about: home, I initially thought this would add to the site to about: home, which would have been nice feature"*.

This report content suggests that the reporter understands the problem context, and knows that something is not matching his expectation, but they did not justify the impact and user difficulties that they have experienced as a consequence of this problem. This defect was rated as a low-level severity and took 1090 days to be resolved. The very long resolution time is likely due to the fact that software developers do not understand how to correct the user interface in order to improve the user experiences – especially when they thought that the user interface is already well designed. This is evidenced in the comments section, where software developers discussed redesigning the feature to be more visible. Perhaps as a consequence of this, more guided-reporting defect forms could be used to collect textual data, especially in describing the issues and user's expectations. For example, we could develop a catalog of usability defects categories, user difficulties, feelings, and impact to support reporters with relevant usability-related vocabulary when describing usability issues. Also, in order to capture user expectations, actual routine task and suggestions to improve usability, a defect form should specifically prompt users for this kind of information. This will particularly benefit reporters with limited usability knowledge to submit higher quality defect reports, and help software developers to understand and evaluate usability defects as being as crucial as functional defects.

C. Community Involvement

We define two research questions to investigate the open-source community involvement in responding to usability and non-usability defects: *"RQ5: How fast open source communities respond to usability defects in comparison to performance defects?"* and *"RQ6: Do different defect types and projects affect the number of comments?"* We measured the community involvement using the following two metrics.

- First comment response time (*FCRT*) in days – the time it took the first comment to be added to a defect report [18], *CommentResponseTime = DateOpened – FirstCommentDate.*
- Total comments (*TC*) – the total number of comments on a defect report. This data is automatically extracted from the defect CSV file.

Table VII shows the total number of comments on a defect report, and the time it took for defect reports to

receive a first comment. ***In response to RQ5, we found usability defects are responded slower than non-usability defects***. A Kruskal-Wallis H test showed that there was a statistically significant difference in *FRCT* between the different types of defects, $\chi 2$ (1) = 132.14, p= 0.000, and projects, $\chi 2$ (2) = 63.528, p=0.000. The most significant response time difference between usability and non-usability defects is observed in the Mozilla Thunderbird, in which usability defects took about twenty times longer to get a first comment. We also found open source communities are more responsive to non-usability defects than usability defects, as the median response time, in days is 0, and in most projects (Mozilla Thunderbird and Firefox for Android) the mean response time is less than a week.

TABLE VII
COMMENT RESPONSE TIME AND NUMBER OF COMMENTS OF USABILITY VS NON-USABILITY DEFECTS

Project	FCRT (days)				TC			
	Usability		*Non-usability*		*Usability*		*Non-usability*	
	Mean	***Median***	***Mean***	***Median***	***Mean***	***Median***	***Mean***	***Median***
Mozilla Thunderbird	100.70	0.00	4.99	0.00	38.14	23.50	31.60	16.00
Firefox for Android	20.84	1.00	6.76	0.00	26.88	19.00	24.90	14.00
Eclipse Platform	69.82	14.00	21.03	0.00	9.68	6.00	10.5	7.00

Among the three projects, community response to usability defects is the slowest in the Mozilla Thunderbird, while non-usability defects are the slowest in the Eclipse Platform. Besides, Spearman's correlation between the number of comments and defect resolution indicates a significant positive relationship between the number of comments and resolution time ($p=0.00 < 0.05$).

In terms of the number of comments received for each type of defect (RQ6), the Kruskal-Wallis H test showed that there is no significant difference between *TC* and defect types, $\chi 2$ (1) = 0.185, $p = 0.667 > 0.05$. However, the significant difference of the Kruskal-Wallis H test was observed between TC and project types, $\chi 2$ (1) = 430.944, $p=0.000 < 0.05$. As shown in Table VII ***in most projects (Thunderbird and Firefox for Android) non-usability defects received few comments.*** However, we found that the Thunderbird project attracts more comments than other projects. The significant difference of total comments received among the three projects suggested that the contributor community of the Thunderbird project is the most active, followed by Firefox for Android and Eclipse Platform. One possible explanation why Eclipse Platform has less comment may be due to the contribution of the technical users that provide more informative information and less discussion.

Based on the previous research [18], and our findings, we acknowledged that open source communities are very responsive to high-risk defects (e.g., performance and security) defects than usability defects. Even though in our study usability defects received more comments than non-usability defects, the difference was not significant. However, we found that regardless of different defect types and projects, the defect resolution time will increase when there are more comments. We postulate that when more comments are submitted during the discussions of defects, more questions, suggestions and debates will take place. Even though this process will delay the time to fix usability defects, the solution outcomes of these discussions could be more practical, ideal and less risky. In most cases, a defect report contains more than five comments.

In order to manage the active discussion, we recommend exploring community-centric improvements to defect repositories for discussion management. The existing linear sequence of comments may be suitable for a defect with relatively few comments and straightforward discussion, but not for complex usability defect discussions. For instance, to monitor how many users experienced a particular usability defect, we could use a one-click response (such as "Like" and thumb button) with a nested comment section to describe the corresponding reproducibility steps. For usability defects, the frequency of users that experienced the problems can provide evidence that the identified problem was true. Besides, developers could also systematically track specific information rather than repeatedly having to scrolls through the whole comments section.

IV. CONCLUSIONS

In this paper, we have compared the descriptions of usability and non-usability defects and investigated how the open-source communities treat these two types of defects. We analyzed the 377 and 1864 of usability and non-usability defect reports respectively from Mozilla Thunderbird, Firefox for Android and Eclipse Platform project. In this paper, we presented insights into the essence of report content, length of the defect resolution process, and community engagement on different types of defects in these projects.

In all three projects, we observed statistically significant differences in the means and medians of the resolution time for usability and non-usability defects. We found usability defects are resolved slower than non-usability defects in all three projects. We also observed that usability defects reported for open source projects usually consist of *output details*, and *software context*, while non-usability defects are preferably explained using *supplementary information*. Besides, both usability and non-usability defects are significantly influenced by a textual description of the *impact, expected output* and *actual output*. However, the positive Spearman's correlation between them and resolution time shows the presence of such information does not help to increase the resolution process. Our findings also show usability defects can be fixed faster when reporters provide more evidence, such as snapshots, video, recovery steps and external links. Results also suggest that community response to non-usability defects is much faster than usability defects. However, there is no significant difference in terms of number of comments received for both types of defects. We also found that most usability defects are treated as low severity and considered as enhancements.

The implication of this research is twofold. Firstly, these research findings extend the knowledge of software defect repositories mining specifically on usability defects, that to our knowledge has not been studied in detail. The valuable information of unstructured textual defect description presented in this paper also highlights the opportunities for other researchers to explore defect prediction models using the textual content of the reports, rather than typically used categorical data. Secondly, our findings may help practitioners, especially software testers, to report relevant usability defect information for software developers. Subsequently, this could facilitate software developers and management to prioritize defect fix task accordingly.

In the future, we plan to extend our text analysis approach using an automatic text analysis tool to improve the efficiency of text preprocessing methods. For example, we could conduct Natural Language Processing (NLP) techniques and bag-of-words representation to categorize the different categories of usability defects. This information would complement defect reports and may help software developers to understand the subjective nature of usability defects better.

Since we have obtained promising findings in this usability defects study, we also plan to improve the existing way of reporting software defects. Perhaps, a new defect report template could be designed to reflect different defect types and project-specific needs. In this way, we could enable users to submit clear defect information according to available information and technical knowledge limitations.

ACKNOWLEDGMENT

Support for the first author from the Fundamental Research Grant Scheme (FRGS) under Contracts FRGS/1/2018/ICT 01/UITM/02/1, Universiti Teknologi MARA (UiTM). We also would like to thank Prof Denny Meyer from Swinburne University of Technology Melbourne for her valuable advice on statistical matters used in this paper.

REFERENCES

[1] R. A. Majid, N. L. M. Noor, and W. A. W. Adnan, "An assessment tool for measuring human centered design adoption in software development process," in *Advances in Intelligent Systems and Computing*, 2018.

[2] N. H. Basri, W. A. W. Adnan, and H. Baharin, "E-participation service in Malaysian e-government website: the user experience evaluation," *Proc. 10th Int. Conf. E-Education, E-Business, E-Management E-Learning*, pp. 342–346, 2019.

[3] D. M. Nichols and M. B. Twidale, "Usability processes in open source projects," *Softw. Process Improv. Pract.*, vol. 11, no. 2, pp. 149–162, Mar. 2006.

[4] C. Wilson and K. P. Coyne, "The whiteboard: Tracking usability issues: to bug or not to bug?" *Interactions*, pp. 15–19, 2001.

[5] R. Stefan, A. Giris, and C. Yilmaz, "How to Provide Developers only with Relevant Information?" in *2016 7th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, 2016, pp. 1–6.

[6] N. S. M. Yusop, J. Grundy, and R. Vasa, "Reporting Usability Defects: A Systematic Literature Review," *IEEE Trans. Softw. Eng.*, vol. 43, no. 9, pp. 848–867, 2017.

[7] S. Zaman, B. Adams, and A. E. Hassan, "Security Versus Performance Bugs: A Case Study on Firefox," in *Proceedings of the 8th Working Conference on Mining Software Repositories*, 2011.

[8] V. Garousi, E. G. Ergezer, and K. Herkilo, "Usage, usefulness and quality of defect reports: an industrial case study," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, 2016.

[9] N. S. M. Yusop, J.-G. Schneider, J. Grundy, and R. Vasa, "Analysis of the Textual Content of Mined Open Source Usability Defect Reports," in *24th Asia-Pasific Software Engineering Conference (APSEC)*, 2017.

[10] J. Uddin, R. Ghazali, M. M. Deris, and R. Naseem, "A survey on bug prioritization," *Artif. Intell. Rev.*, vol. 47, no. April, 2016.

[11] M. G. Capra, "Usability Problem Description and the Evaluator Effect in Usability Testing," 2006.

[12] N. S. M. Yusop, J. Grundy, and R. Vasa, "Reporting Usability Defects – Do Reporters Report What Software Developers Need?" in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, 2016.

[13] E. I. Laukkanen and M. V. Mantyla, "Survey Reproduction of Defect Reporting in Industrial Software Development," in *International Symposium on Empirical Software Engineering and Measurement*, 2011, pp. 197–206.

[14] N. S. M. Yusop, J. Grundy, and R. Vasa, "Reporting Usability Defects: Limitations of Open Source Defect Repositories and Suggestions for Improvement," in *Proceedings of the 24th Australasian Software Engineering Conference*, 2015, pp. 38–43.

[15] U. Raja, "All complaints are not created equal: text analysis of open source software defect reports," *Empir. Softw. Eng.*, vol. 18, no. 1, pp. 117–138, Jan. 2012.

[16] T. D. Sasso, A. Mocci, and M. Lanza, "What Makes a Satisficing Bug Report?" in *IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 2016.

[17] M. R. Karim, A. Ihara, X. Yang, E. Choi, H. Iida, and K. Matsumoto, "Improving the High-Impact Bug Reports: A Case Study of Apache Projects," 2016.

[18] P. Bhattacharya, L. Ulanova, I. Neamtiu, and S. C. Koduru, "An empirical analysis of bug reports and bug fixing in open source Android apps," in *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, 2013, pp. 133–143.