2017 Asia-Pacific Conference on Software Engineering (APSEC2017), 4-8 December 2017, Nanjing, Jiangsu, China, (c) IEEE 2017

Analyis of the Textual Content of Mined Open Source Usability Defect Reports

Nor Shahida Mohamad Yusop, Jean-Guy Schneider
School of Software and Electrical Engineering
Swinburne University of Technology
Melbourne, Australia
{nmohamadyusop, jschneider}@swin.edu.au

John Grundy, Rajesh Vasa
Faculty of Science, Engineering and Built Environment
Deakin University
Melbourne, Australia
{j.grundy, rajesh.vasa}@deakin.edu.au

Abstract— **Writing a good usability defect report can be a tedious task, especially in identifying what important information should be included, and capturing the attention of software developers to fix them. This paper is a continuity of our previous studies investigating software development practitioners' day-to-day practices when dealing with usability defects. In this study, we mined 377 developer-tagged usability defect reports from Mozilla Thunderbird, Firefox for Android and Eclipse Platform to confirm what software development practitioners claimed to provide when reporting usability defects. We looked for the presence of key defect attributes – steps to reproduce, impact, software context, expected output, actual output, assumed causes, solution proposal and supplementary information. In addition, we analyzed the trend of different types of usability defects, correlation between usability defects and defect severity, and failure qualifier. Our findings demonstrate a mismatch between what software development practitioners claimed to provide when reporting usability defects, and the information that actually appears in the defect reports. The results of our research have important implications for software defect reporting, especially in designing more effective mechanisms for reporting usability defects.**

Keywords-Bugzilla; software defect reporting; usability defects;

I. INTRODUCTION

Reporting usability defects can be a challenging task, especially in convincing software developers that the usability defect reported is indeed a real defect. Specifically, the subjective nature of usability defects that cause confusion for some people require stronger evidence to describe and report the problem. However, research to date in software defect reporting has not investigated the capture of different information based on defect types, such as usability defects. This lack of empirical data on information needs for different types of defect reporting impedes research on finding what information is best to describe a usability defect. While previous studies have identified steps to reproduce, actual output and expected output as an important information to fix software defects in general [1], [2], however, these studies do not consider what information should be reported, and how the information should be presented when it come to specific types of defects.

Our work fills this gap by focusing on usability defect reporting. In our previous studies [3], we have surveyed software development practitioners to identify what kind of information do they provided when reporting usability defects. In order to verify what the software development practitioners claimed in the surveys, we conducted an empirical investigation of 377 usability defects in Mozilla Thunderbird, Firefox for Android and Eclipse Platform projects. In addition to previous studies that investigate the characteristics of defects based on predefined defect data (i.e., product, component, version) and comments [4]–[8], we examined the textual content of usability and performance-related defects to find out if there is any difference in the way these two types of defects are described. We picked performance-related defects as a comparison benchmark because they are a non-functional type of defect that are commonly reported in open source projects and often studied by other researchers [9].

The rest of the paper is organized as follows. In Section 2, related work and motivational example is presented. In Section 3 we describe the methodology of our study. Section 4 follows with results and discussion in comparison with previous studies. Section 5 outlines threats to validity. The paper concludes with a summary, implications and future works in the conclusion Section 6.

II. RELATED WORK AND MOTIVATION

Although some recent work has analyzed defect repositories, very little has been done on mining and understanding usability defects. Most research has focused on improving the defect resolution of critical defects such as performance and security defects [7], [10], [11]. Other work focuses on identifying defect characteristics of software defects in general [5], [11]–[15]. To the best of our knowledge, Twidale et al. [16] provides a starting point to studying open source usability defects. In contrast to our work that studied usability defect report content, they examined the usability discussions in comments section and revealed four main challenges – (1) the difficulty to textually report usability defects, (2) the use of posted textual comments that is not suitable to discuss and keep track of design ideas, (3) the subjective nature of usability defects that leads to many disagreements, and (4) the ripple effects of fixing usability defects.

In this work we are interested in addressing the first challenge. The motivation for investigating usability defects is that these types of defects have been said to receive less

attention from software developers [17]. Furthermore, work in the usability and Human-Computer Interaction (HCI) literature has focused more on improving usability evaluation methods and given relatively little attention to the content of usability defect description [18], [19]. We postulate the completeness of a defect report often determines how quickly it will be fixed. According to [20]–[22], a well-written usability defect description should describe impact and severity, cause of the problem, observed user actions, solution to the problem, support with data and avoid wordiness and jargon. However, in the absence of formal usability evaluation in open source projects, aspects such as likely difficulties [23], impact [24], user's feelings, emotion and "struggling" with the interface [16], which is important to convince software developers about the validity of the problem are often missing in defect report.

To get a deeper insight into how important the early presence of certain information can speed up the defect correction process, consider usability defect 846414 in Mozilla Thunderbird, as shown in Fig. 1. The defect was opened on February 2013, but only responded by Developer A after a year noting that the defect is not of high priority and they have a lot of other things to work on. This defect report is very minimal, containing just summary of the problem extracted from a blog post and the reporter's expectation. Possibly, the unclear description of user difficulty and solution to the problem makes this issue slip to the bottom of the list of things to fix. Surprisingly, a patch to the issue was ready by Developer B on the same day Developer A expressed their idea to solve the issue. This suggests that including all relevant information when a defect is first reported is clearly important to help software developers prioritize the defect, fix the defect and speed up the defect resolution time. However, the current unstructured free-text defect report form may not help reporters to report such information at the initial report submission [16], [25], [26]. Studies have shown that important information, such as stack traces, error reports and test cases are common to be provided at a later time[13]. Furthermore, without specific prompts for this information, it can be challenging for non-technical users to be aware what kind of information should be provided.

Bug 846414: Hide the "Show All Tabs" button when there are less than 2 tabs

Part of a blog post that we are pushing to redo the TB UI.
Point 7: Only show "show all tabs" button when there are multiple tabs.
For more details see:
http://infinite-josiah.blogspot.com/2013/02/thunderbird-ui-concept.html
==========
It is pointless to show the button that lists all your tabs when you only have one open. In fact, it really is pointless to show them unless you have more tabs than your Window can hold.
This bug is to remove it/hide it when not being useful.

Figure 1. Example of a ONE-COLUMN figure caption.

III. METHODOLOGY

A. Research Questions

In order to investigate how usability defects are currently described in representative open source communities and to reveal common usability defect characteristics, we formulated the following six research questions:

- RQ1: What information is commonly provided in open source usability defect reports?
- RQ2: How if at all, is a proposed solution to the usability problem described?
- RQ3: Are usability defects described differently from performance-related defects?
- RQ4: What are the dominant types of usability defects (e.g., interface and interaction) in open source projects?
- RQ5: What are the impacts of usability defects and what types of usability defects have a severe impact?
- RQ6: On what basis, do usability defect reporters justify that the user difficulty that they experience is an issue?

B. Projects and Report Extraction

To answer these research questions, we performed an extraction and analysis of usability and performance-related defects gathered from the Bugzilla defect repository of the Mozilla Thunderbird, Firefox for Android and Eclipse Platform projects. Our choice of these projects was based on the following factors:

- These projects represent a variety of different uses and environments;
- These projects have significant graphical user interfaces (GUI) that use windows, icons and menus and user tasks can be manipulated by a mouse, keyboard or touch screen;
- These projects use standard defect-reporting templates provided by the Bugzilla defect repository, hence maintaining consistency when comparing the type of information presented when reporting defects;
- These projects make use of keywords to label and classify defect type reducing selection bias; and
- These projects are among the most successful user-facing applications that engage various levels of user participation with different levels of knowledge and technical experience.

Across the three projects, 23,373 defect reports are available to download in CVS format. However, we only studied 377 FIXED defect reports tagged with predefined Bugzilla usability keywords as listed in Table 2. The reason we chose to use developer-tagged usability defect reports is to reduce selection bias, as the software developers already completed the resolution process and have reached agreement on the actual types of defects reported and corrected. We extracted sample defect reports for each project in four steps: 1) filter defect reports that were resolved as FIXED; 2) specify columns/ attributes that we

wish to appear in the defect list. In this study, we add *Keywords, Opened, Reporter, Number of Comments* and *Last Resolved* attributes; 3) extract and save the data in CSV format; 4) filter the usability and performance defect reports. Since the usability defects downloaded for all projects only constitute a small percentage of all reported defects we chose to analyze all of them in this work.

C. Analyzing defect Report Content

Our analysis of usability defect report content only focused on the initial reporting of a defect, not investigating the subsequent discussion about the problem and its possible solution in the comments sections. We used the defect report *title, description* and *attachment* fields as our main source of investigation. We defined eight metrics based on Capra's guidelines as described in [27] to assess the presence of certain information when describing usability defects. Since defect reports presented in open source defect repositories are in unstructured plain text, we are unable to automatically assess the presence of these metrics. Even though the Bugzilla defect report template can be customized to label some of these metrics, such as "Steps to reproduce", "Expected Output" and "Actual Output", many reporters do not explicitly describe these metrics.

We read all 377 usability defect reports and manually identify whether the criteria listed in Table 1 were present in the defect report. The presence of *steps to reproduce (STR), impact (IMP), software context (SC), expected output (EO)*, *actual output (AO), assume cause (AC), solution proposal (SP),* and *supplementary information (SI)* set as 1 implies that the "information exist", and 0 implies that the "information does not exist". Since *IMP, AC* and *SP* do not have separate fields, we measured the presence of these information based on the following criteria:

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

The procedure to analyze the defect reports consisted of going through each report twice. The first reading focused on understanding the context of usability problems and identifying the main interface or/ interaction problems described by the reporter. The second reading was to highlight the keywords and snippets of the defect description describing the problem types, impact, and failure qualifier based on the previous classification. We used the card-sorting technique to group impact information into several groups that have similar ground of user difficulty, while problems types and failure qualifier were reorganized according to Usability Problem Taxonomy (UPT) and Orthogonal Defect Classification (ODC), respectively.

D. Data Analysis

We report our analysis results with descriptive statistics. Chi-square test of independence was used to find the significance relationship between: (1) the types of projects and defects, and the presence of seven usability defect attributes, and (2) the presence of impact information and defect severity.

TABLE I. CRITERIA USE TO RATE LEVEL OF DETAILS OF USABILITY DEFECT DESCRIPTION BASED ON CAPRA'S GUIDELINE [20]

Quality criteria	Related Capra's Guidelines
1. Is the defect report describe details steps to reproduce the defect? [11] (*Steps to reproduce*)	Describe observed user actions
2. Does the defect report indicate the effect of the problems on the user? (*Impact*)	Describe the impact and severity of the problem
3. Does the defect report describe the problematic part of the user interface? (*Software context*)	Describe the impact and severity of the problem
4. Does the defect report contain details of expected output and actual output? [11] (*Actual and expected output*)	Describe observed user actions
5. Does the defect report contains criteria used to justify the usability problem identified is true? (*Assumed cause*)	Describe the cause of the problem
6. Does the defect report contain design ideas? (*Solution proposal)*	Describe a solution to the problem
7. Does the defect report contain support information as evidence to the problem? (*Supplementary information*)	Support your findings with data

TABLE II. USABILITY DEFECTS STUDIED

Project	Total	Other resolution	Resolved/ Verified						
			Fixed	***Duplicate***	***Incomplete***	***Invalid***	***Wontfix***	***Worksforme***	***Expired***
Mozilla Thunderbird	384	185	**88**	64	4	9	16	17	1
Firefox for Android	292	62	**101**	59	3	11	36	20	0
Eclipse Platform	530	78	**188**	46	-	68	103	47	-
Total	1206	325	**377**	169	7	88	155	84	1

Other resolution – New, unconfirmed, assigned, and reopened
Usability-related – 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, ux-visual-hierarchy

IV. Results and Discussion

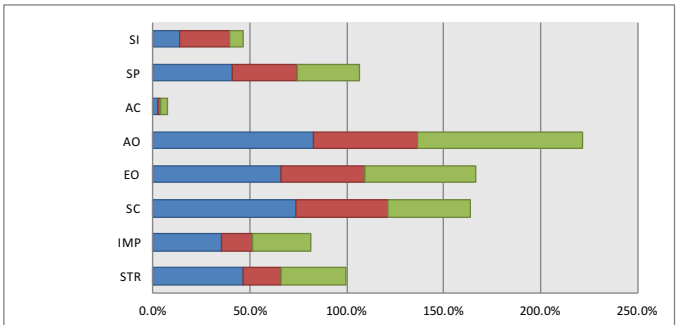
A. Usability Defect Report Contents

In this research we are interested in investigating the presence of the following key information in usability defect report descriptions: (1) steps to reproduce the problem; (2) user difficulty that could effect human task and emotional reaction; (3) context, or settings, in which the problem occurs; (4) actual and observed results; (5) assumption of or known cause of the problem; (6) solution proposed to solve the problem; and (7) supplementary information. We answer the following research questions: *RQ1: What information is commonly provided in open source usability defect reports?* when the defect is initially reported. In addition, we investigate *RQ2: How if at all, is a proposed solution to the usability problem described?*

Fig. 2 shows that the most widely reported attributes in usability defect reports open source projects studied are *AO, EO* and *SC*. Attributes rarely presented in usability defect description are *AC* and *SI*. The *AC* could only be found in less than 5% of defect reports. While *STR* is considered as the most important attributes in defect reports [1], [2], our study found that only between 19% - 46% of defect reports contain this information, depending on project.

Similar to previous findings investigating open source defect reporting in general [13], we also found that the presence of information for usability defects varies between projects. Referring to Table 3, the p-value for the Chi-square tests for each attribute except *AC*, which is less than 0.05, suggesting that there is indeed a relationship between projects and the presence of certain attributes for usability defects. As shown in Fig. 2, it is apparent that certain attributes are common in some projects. From this data, we can draw several observations. First, the *AO* is found in more than 80% of Mozilla Thunderbird and Eclipse Platform. Surprisingly, only about 50% of Firefox for Android usability defects contain actual outcome even though the defect reporting tool specifically prompts its users for this information. Second, while Mozilla Thunderbird and Eclipse Platform reporters mainly used textual description to describe usability defects, Firefox for Android more often provided supplementary information. As we can see from Table 4, screenshots are the most favorable supplementary materials used in Firefox for Android. Third, across the three projects, Firefox for Android usability defects contained very little information on *STR* (19.%), *IMP* (15.8%), and *EO* (43%). Fourth, given the nature of the open source communities, which the users have varying level of technical knowledge, defect reports in all projects contained virtually no *AC*. This attribute, however, is not necessarily appropriate for all types of usability defects. For example, a usability defect that is not related to technical deficiency is quite difficult to be explained even by technical users. The limited knowledge and experience of open source communities around usability and HCI aspects is one of the barriers for reporters to technically relate the usability issues and the violated usability principles and HCI guidelines. In the context of understanding the root cause of usability defects, perhaps *AC* could be explained in the form of rationale as a ground for a particular usability issue.

For answering RQ2, we observed about one third of the defect descriptions contain *SP*. A close inspection of usability defect data as in Table 6 shows that about 30% to 40% of solution proposals were described in words. The other means of describing solution proposals were in a form of ASCII art, graphical elements or code, and UI-digital mockups, that only account less than 2% of defects in all three projects.



	STR	IMP	SC	EO	AO	AC	SP	SI
Mozilla Thunderbird	46.6%	35.2%	73.9%	65.9%	83.0%	2.3%	40.9%	13.6%
Firefox for Android	19.8%	15.8%	47.5%	43.6%	53.5%	2.0%	33.7%	25.7%
Eclipse Platform	33.0%	30.3%	42.6%	56.9%	85.6%	3.2%	31.9%	6.9%

Figure 2. Distribution of usability defects in which each attribute was presented, by project

TABLE III. Chi-Square Tests Results to Examine the Influence of Projects and the Presence of Defect Attributes

Defect attributes	ρ value	χ^2 (df=2, n=377)
Steps to reproduce (STR)	**0.000**	16.67
Impact (IMP)	**0.006**	10.16
Software context (SC)	**0.000**	24.96
Expected output (EO)	**0.003**	11.51
Actual output (AO)	**0.000**	32.78
Assumed cause (AC)	0.934	0.137
Solution proposal (SP)	**0.024**	7.45
Supplementary information (SI)	**0.000**	26.68

TABLE IV. Types of Supplementary Information Provided in Usability Defect Reports

Supplementary information	Project		
	Mozilla Thunderbird (n=88)	***Firefox for Android (n=101)***	***Eclipse Platform (n=188)***
Annotated screenshots	1.1%	0.0%	0.0%
Error report	1.1%	0.0%	0.5%
External URL link	**6.8%**	**10.0%**	1.1%
Problem occurrence	0.0%	1.0%	1.6%
Screenshots	0.0%	**13.9%**	1.1%
Stack traces	0.0%	0.0%	1.1%
Task workaround	2.3%	0.0%	0.0%
Usability guidelines	0.0%	0.0%	0.5%
Violated UX-consistency	2.3%	0.0%	0.0%

TABLE V. CHI-SQUARE TEST RESULT TO EXAMINE THE INFLUENCE OF DEFECT TYPES (PERFORMANCE-RELATED AND USABILITY DEFECTS) AND THE PRESENCE OF DEFECT ATTRIBUTES

Attributes	Mozilla Thunderbird			Firefox for Android			Eclipse Platform		
	ρ	χ^2 (df=1, n=121)	Φ	ρ	χ^2 (df=1, n=177)	Φ	ρ	χ^2 (df=1, n=800)	Φ
Steps to reproduce	0.06	3.69	0.18	0.13	2.30	0.11	**0.01**	**7.54**	**0.10**
Impact	**0.03**	**4.54**	**0.19**	0.18	1.84	0.10	**0.00**	**65.65**	**0.29**
Software context	**0.00**	**12.45**	**0.32**	**0.03**	**4.57**	**0.16**	**0.00**	**3.52**	**0.07**
Expected output	**0.01**	**6.85**	**0.23**	**0.00**	**12.25**	**0.26**	**0.00**	**58.21**	**0.27**
Actual output	0.21	1.58	0.11	0.21	1.59	0.10	0.75	0.10	0.01
Assumed causes	0.38	0.76	0.08	**0.00**	**14.26**	**0.28**	**0.00**	**32.95**	**0.20**
Solution proposal	0.88	0.02	0.01	0.77	0.09	0.02	0.05	3.97	0.07
Supplementary information	0.83	0.05	0.02	**0.00**	**8.06**	**0.21**	**0.00**	**11.59**	**0.12**

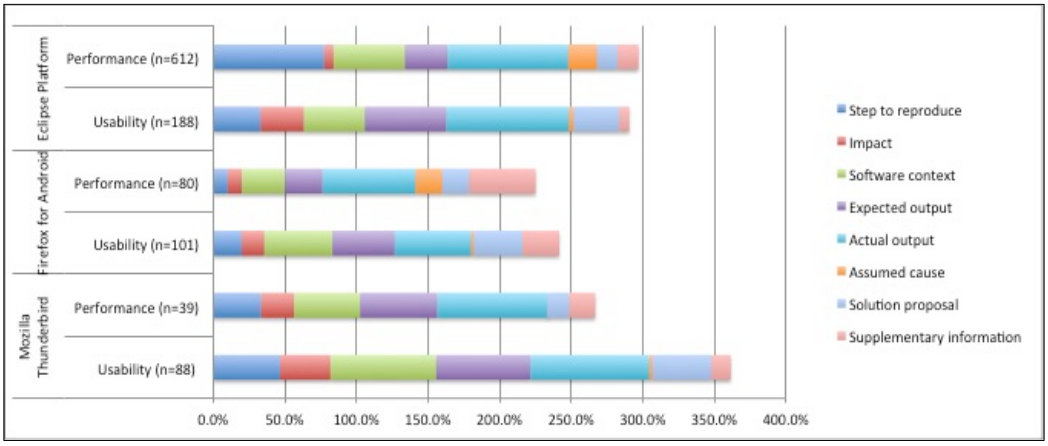


Figure 3. Distribution of defect attributes between usability and performance-related defects.

TABLE VI. THE WAY SOLUTION PROPOSAL IS PRESENTED

Solution proposals presentation	Project		
	Mozilla Thunderbird (n=88)	*Firefox for Android (n=101)*	*Eclipse Platform (n=188)*
Textual descriptions	39.8%	29.7%	30.3%
ASCII arts	1.1%	1.0%	0.0%
Graphical elements or code that could be immediately implemented	0.0%	2.0%	0.5%
UI digital mockups	0.0%	1.0%	0.5%

B. *Differences Between Usability and Performance Defect Report Content*

In this research we are also interested in investigating how the different types of defects are described using different information. In answering *RQ3: Are usability defects described differently from performance-related defects?* we chose performance-related defects as a benchmark to compare these differences. We considered the defect attributes STR, IMP, SC, EO, AO, AC, SP and SI as our dependent variables (nominal data) and defect types as the independent variable (nominal data). We read the defect description and rated the presence or absence of information as 0 = information is present, 1 = information is not present, respectively.

As shown in Table 5, at a significant level p=0.05, we found relationships between *software context* and *expected output*, and defect types in all the three projects. However, no relationships were found between *actual output* and *solution proposal*, and defect types in the three projects. At the same significance level, the relationship between *assumed causes* and *supplementary information* and defect types was only significant in Eclipse Platform and Firefox for Android projects, while relationship between *steps to reproduce* and defect types was only observed in Eclipse Platform.

Fig. 3 reveals several observations about these relationships. First, *actual output* is commonly reported both in usability and performance-related defects for all three projects. On average, more than 70% of usability and performance defects contain explanation about what happened or what they saw while using the software product. Second, in contrast to Mozilla Thunderbird and Firefox for Android, *steps to reproduce* is most often used to explain in performance-related rather than usability defects. Third, *assumed causes* is least reported for both usability and performance-related defects across the projects. However, we found Firefox for Android and Eclipse Platform performance-related defects contain more information on *assumed causes* than usability defects compared to Mozilla Thunderbird. Fourth, while *supplementary information* is mostly attached to performance-related defects, *solution proposals* is more common in usability defects. Interestingly, Firefox for Android defect reports contain more *supplementary information* as compared to Mozilla Thunderbird and Eclipse Platform. Fifth, less than 30% of Firefox for Android and Eclipse Platform performance-related defects described the *expected output* (26.3%, 29.9%, for Firefox for Android and Eclipse Platform, respectively). In summary, we found that usability and performance-related defects reported in open source projects usually contain output details and software contexts. Possibly, these findings were influenced by the generic Bugzilla tool

defect report form that has specific attributes for actual result, observed result and software context attribute.

C. Types of Usability Defects

In this section, we describe the distribution of different types of usability defect categories, including interface and interaction usability defects. We answer *RQ2: What are the dominant types of usability defects (e.g., interface or interaction) in open source projects?* In addition, we study the subcategories of interface and interaction usability defects, as suggested in [28], [29].

Fig. 4 summarizes the distribution of usability defects within different categories and subcategories. Based on UPT [28] and fault GUI model [29], we classified interface usability defects into GUI structure and aesthetics, information presentation and audibleness, while interaction usability defects are divided into manipulation, task execution and functionality.

From Fig. 4, we can draw the following findings and implications:

a) Both interface and interaction usability defects are common in all three open source projects. However, as can be seen in Fig. 4b GUI structure and aesthetics are the most dominant types of usability defects, 34.1% in Mozilla Thunderbird, 36.6% in Firefox for Android, and 35.1% in the Eclipse Platform. To this end, the object state problem is responsible for a significant proportion of the GUI structure and aesthetics categories in Mozilla Thunderbird and Eclipse Platform (Fig.4c). Object state issues account for 15.9% and 14.9% of the sample usability defects in Mozilla Thunderbird and Eclipse Platform, respectively. Firefox for Android, on the other hand, contains a much smaller percentage of object state problems (7.9%) but had the largest percentage of object appearance problems, accounting for 18.8%. One possible reason is due to the nature of the application, in which Firefox for Android is an application developed for mobile devices that have several limitations such as small screen, single window, touchscreen, and screen orientation support. For example, the small screen size may limit the content displayed and organization of information that substantially affects the overall look and feel of the application.

b) For the information presentation and audibleness category, all three projects only contain a small percentage of defects, accounting for less than 7% of the usability defects reported. Among the information presentation problems, error and notification message and menu structure are the highest problems reported for Eclipse Platform and Mozilla Thunderbird, respectively. One possible reason for the large fraction of error message problems may be that the Eclipse Platform is targeting technical users, and therefore the use of syntax error codes or abbreviations in error message such as *"an error of type 2 has occurred"* should not be a problem. However, as everyone can contribute to open source projects, inexplicit and technical messages may sometimes not be understandable by users with limited "technical-development" knowledge. Perhaps, no matter what kind of projects and intended users, error messages should include explicit, polite, precise, constructive advice written in human-readable language so that all users can understand it. Surprisingly, non-message feedback and data error is virtually not exist in Firefox for Android, while these problems occur in less than 4% of defects in Mozilla Thunderbird and Eclipse platform.

c) As shown in Fig.4b, issues related to functionality is the dominant interaction problem and our study results indicate that Firefox for Android has the highest amount of missing functionality, accounting for 14.9% of defects. As Mozilla Thunderbird is a more stable project than Firefox for Android and Eclipse Platform, technical deficiency is very low in comparison.

d) For task execution subcategories, incorrect task action has the largest proportion in Mozilla Thunderbird (17%), followed by Eclipse Platform (13.3%). In all three projects, less than 2% of usability defects had reversibility problems.

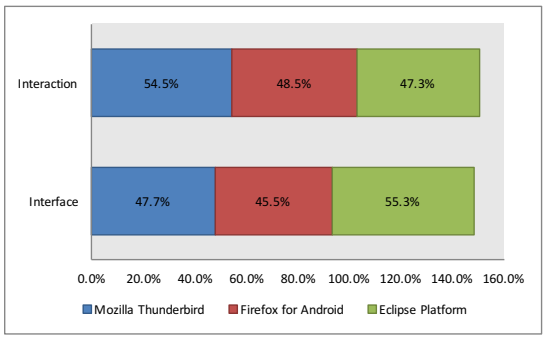
e) As shown in Fig.4f, keyboard press accounts the largest manipulation problems in Mozilla Thunderbird (9.1%) and Eclipse Platform (6.4%). Across the three projects voice control does not exist. This finding suggests that the voice control category is not pertinent to the current usability problem taxonomy and could be eliminated. Unsurprisingly given the nature of the application, Firefox for Android usability defects contained virtually no mouse click and drag and drop problems, and very few keyboard problems (1.0%). As Firefox for Android is an application developed for a mobile device, finger touch is the relevant. Also, likely due to the fact that direct manipulation using mobile gesture is a very sensitive interaction, scrolling is more common in Firefox for Android compared to Mozilla Thunderbird and Eclipse Platform.

D. The Impact of Usability Defects

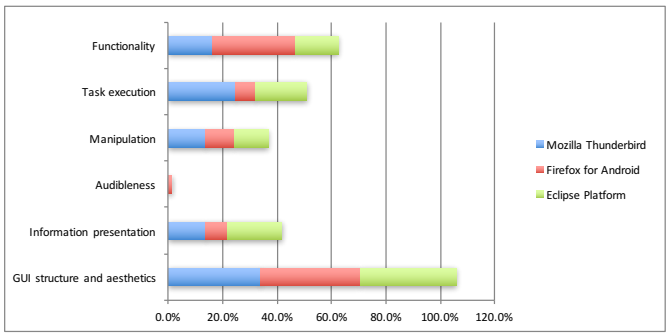
In this section, we describe the distribution of effect and the correlation between effects and defect severity. We answer *RQ3: What are the effects of usability defects and what types of usability defects have severe impact?*

In our analysis, we identified two fundamental types of impact: the (1) impact on *human emotion,* to which the defect reporter described their feelings when they were experiencing difficulties using the software due to the usability defect, and the (2) impact on *task performance,* that prevented the reporter from completing task execution. Overall, when describing a particular user difficulty, reporters tended to support a single impact description at a time (rather than discuss both impact on emotion and task performance) and reporters provided little evidence for their impact claims.

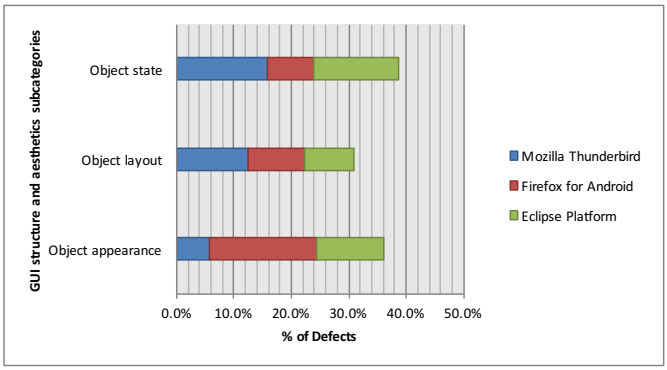
Fig. 5 summarizes the distribution of different impacts with the corresponding defect severity. It shows that accessibility is the dominant task performance impact in Mozilla Thunderbird, accounting for 11.4% of sampled defects.



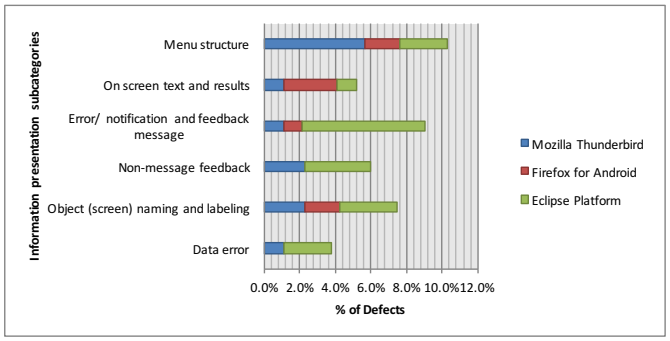
a) Interface vs interaction usability defects



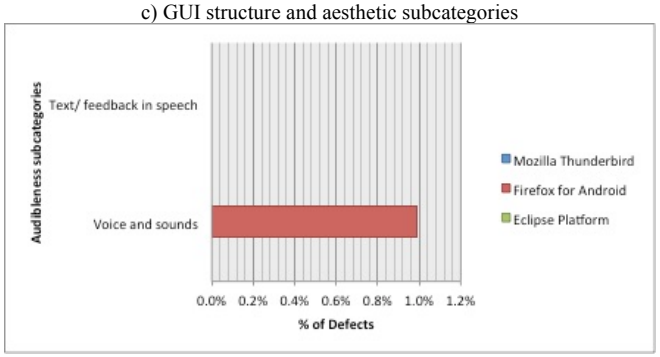
b) Interface and interaction defects subcategories



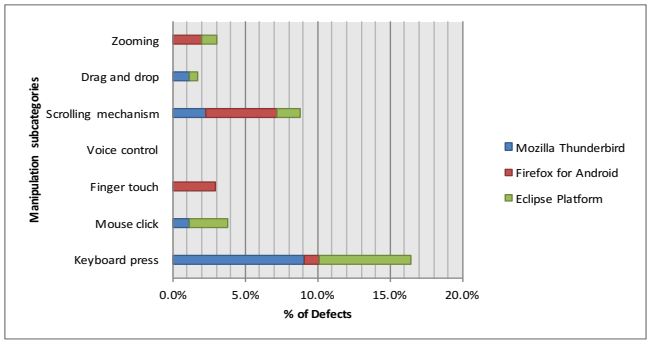
c) GUI structure and aesthetic subcategories



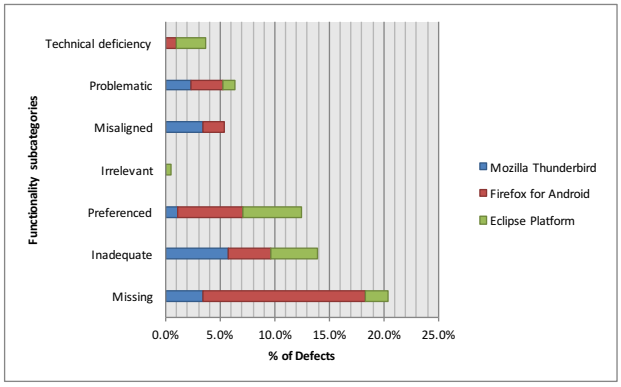
d) Information presentation subcatgories



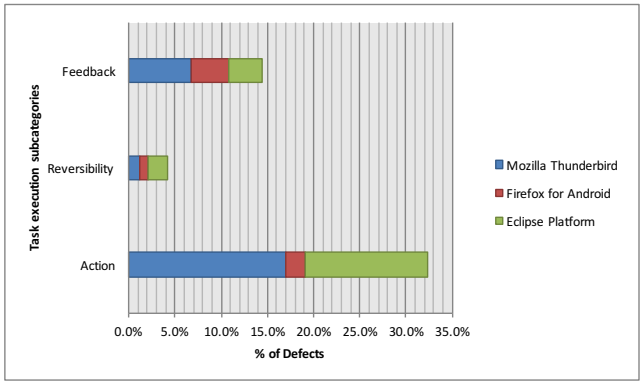
e) Audibleness subcategories



f) Manipulation subcategories

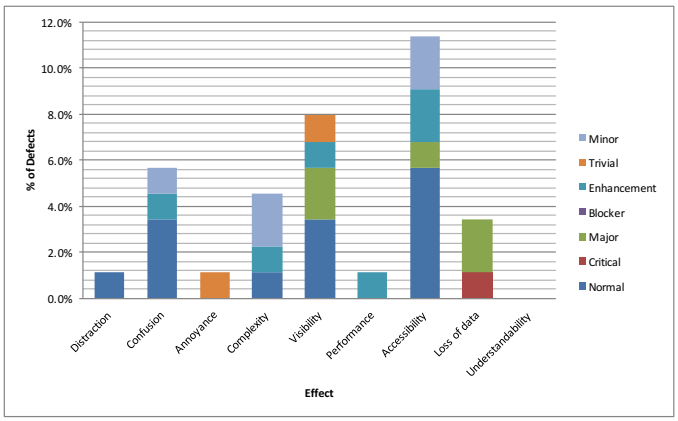


g) Functionality subcategories

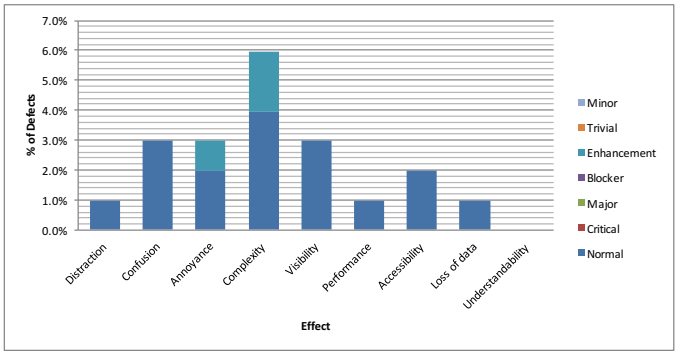


h) Task execution subcatgories

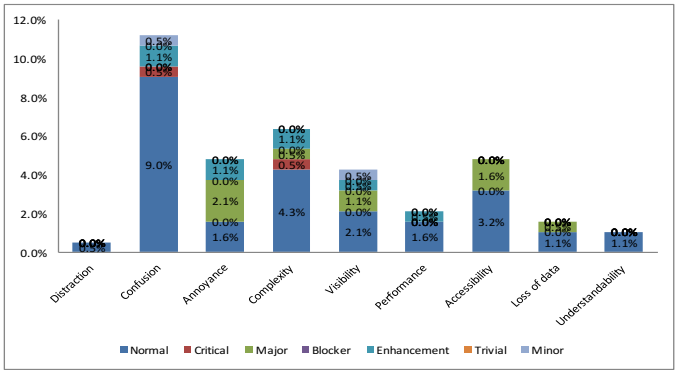
Figure 4. Distribution of usability defects with categories and subcategories.



a) Mozilla Thunderbird



b) Firefox for Android



c) Eclipse Platform

Figure 5. Distribution of usability defects impact and defect severity.

TABLE VII. DISTRIBUTION OF FAILURE QUALIFIER ACROSS PROJECTS

Failure Qualifier	**Project**		
	Mozilla Thunderbird (n=88)	***Firefox for Android (n=101)***	***Eclipse Platform (n=188)***
Wrong	23.9%	9.9%	19.1%
Missing	3.4%	1.0%	1.1%
Irrelevant	5.7%	5.9%	3.2%
Better way	25.0%	10.9%	22.9%
Overlooked	3.4%	1.9%	1.1%
Incongruent mental model	27.3%	20.8%	15.4%
None	11.4%	49.5%	37.2%

In terms of impact on human emotion, confusion is the most addressed in all three projects: 5.6% in Mozilla Thunderbird, 3.0% in Firefox for Android, and 11.1% in Eclipse Platform. However, these percentages are still considered low, suggesting that the open source usability defect descriptions do not contain sufficient information to describe how the interface-interaction defects cause user difficulty that confuse and mislead users. Perhaps as a consequence of this, usability defect reporting tools should specifically prompt its users for this information in a separate field so that users clearly describe this information.

As we can see, a large fraction of usability defects are rated as "normal" severity: 14.7% in Mozilla Thunderbird, 17.0% in Firefox for Android, and 24.5% in Eclipse Platform. Only 6.8% of usability defects in Mozilla Thunderbird and Eclipse Platform were reported as being a severe problem that causes system failure, data loss and impairs the product function: major (5.7% in Mozilla Thunderbird, and 5.8% in Eclipse Platform), and critical (1.1% in Mozilla Thunderbird, and 1.0% in Eclipse Platform). While the usability defect severity distribution in Mozilla Thunderbird and Eclipse Platform varies between enhancement, trivial, minor, major, critical and blocker, Firefox for Android usability defects were only rated as normal and enhancement. One possible reason for this situation is may be because normal severity was set as the default option for selection of severity when reporting a defect using the project Bugzilla defect repositories [30].

In order to understand the influence of defect report impact on severity, we carried out Chi-Square tests for independence to examine whether the defect severity is dependent on the presence of impact information. We found $\chi^2(5) = 9.076$, $p=0.169 > 0.05$, suggesting that there is no relationship between the reported defect severity and the presence of impact information in the usability defect report.

E. Usability Defect Failure Qualifier

In a formal usability evaluation, failure qualifier can usually be easily identified by observing how users experience a particular usability problem during testing. However, in the context of open source projects, where usability defects are normally reported as a result of "black box" usage without a proper usability evaluation method, it is quite difficult to identify the qualifier of the problems. This section answer *RQ4: On what basis, do usability defect reporters justify that the user difficulty that they experience is an issue?* This information is helpful for software developers to understand the nature of the problem. In our study, we used qualifier attributes of the ODC [31] and revised some of the original definitions to suit the context of our analysis.

As shown in Table 7, the most common failure qualifiers across the three projects are incongruent mental model (27.3% in Mozilla Thunderbird, 20.8% in Firefox for Android, and 15.4% in Eclipse Platform), better way (25.0% in Mozilla Thunderbird, 10.9% in Firefox for

Android, and 22.9% in Eclipse Platform) and wrong (23.9% in Mozilla Thunderbird, 9.9% in Firefox for Android, and 19.1% in Eclipse Platform). With regards to incongruent mental model, it was common for reporters to compare two different things that have similar characteristics in order to justify a desire for consistency in design. It was also common for reporters to use their previous experiences to illustrate some point of view on certain issues:

> *With the menu changes, and the removal of the numbered perspective items, we need another way of switching between perspectives. We should add a perspective switcher**, similar to the editor and view switchers*** ... [Consistency]
> *With recent E4 builds I often see unjustified flickering in the workbench window. **I haven't seen this before**.* [Previous experience]

Only a small fraction of usability defect reports contain missing (3.4% in Mozilla Thunderbird, 1.0% in Firefox for Android, and 1.1% in Eclipse Platform), overlooked (3.4% in Mozilla Thunderbird, 1.9% in Firefox for Android, and 1.1% in Eclipse Platform) and irrelevant (5.7% in Mozilla Thunderbird, 5.9% in Firefox for Android, and 3.2% in Eclipse Platform) to support the claim of the usability issues. Overall, we observed that these failure qualifiers are primarily used to support a claim and defend solution proposal ideas.

V. USING THE TEMPLATE

Construct validity concerns the appropriateness of the studied metrics. For example, for our assessment of the existence certain information in usability defect reports, we have chosen important usability attributes from human computer interaction studies [20]. One possible threat might be the reliability of the attributes to be significant for comparing performance defects with usability defects. We minimized this threat by using common defect attributes such as *steps to reproduce, actual* and *expected output*, and introducing general attributes like *supplementary information*.

Internal validity For each defect in the sample, the first author manually read the textual description. This process required reading the bundle of text, interpreting the problems, classifying the information into any of the defined attributes and assigning a score if the information is presented. Since this qualitative analysis leads to the subjective evaluation, incorrect interpretation, classification and scoring may potentially affect our results. To overcome this threat the other authors crosschecked a selection of classifications. We also built a checklist that consists of guidelines, so that researchers are consistent when classifying and giving score for the information presented in the defect reports.

External validity concerns the generality of our findings. Our study is limited to open source projects and restricted to the Bugzilla defect repository. To reduce selection bias and for a comparison purpose, we selected two projects from the same defect repository while the third project is from a different defect repository. We examined three open source projects, with projects spanning diverse product functionality and environment; this small sample cannot be regarded as representative. However, we did not consider the defect reports from closed defect repositories, which might affect our results. We plan to replicate the study with a large number of datasets in future to test the outcome of this study.

VI. CONCLUSION AND FUTURE WORK

We manually examined 377 usability defect reports from Mozilla Thunderbird, Firefox for Android and Eclipse Platform. We presented insights into the current practice of describing usability defects by open source communities. In all three projects, *actual output, expected output* and *software context* are the most widely reported attributes for usability defects, while *assumed cause* and *supplementary information* are the least. It was also surprising that *steps to reproduce* is less reported in Mozilla Thunderbird, Firefox for Android and Eclipse Platform even software practitioners in our previous survey claimed that they always provide this information while reporting usability defects. These findings raise the question of how the absence of such important information will affect time to fix? In comparison to performance-related defects, *solution proposal* are more common for usability defects. Additionally, we studied usability defects from three different dimensions (usability defect categories, user difficulty and failure qualifier).

These research findings have important implications for further research and practice of open source software development alike. From a research perspective, this study adds to the body of knowledge focused at improving defect reporting approaches by considering the need of different defects types that may need to be further verified and extended in subsequent research. For practitioners, especially software testers, it provides a general guide to the important defect attributes that should be reported for certain types of defects in order to improve defect resolution time. Additionally, our findings can facilitate software developers and management to prioritize defect fix tasks.

In future work, we would like to investigate the different categories usability defects belong to, and how categories vary across projects. Since our manual analysis may be biased on human interpretation, further research using text mining tools, such as Weka, could be explored to automatically classify certain words and phrases into different categories. While we have done some comparison with performance-related defect reports, we believe comparison with other types of fixed defect reports is needed. Perhaps, a similar study could be extended to investigate an in-depth relative comparison with other kinds of functional and non-functional defect reports, so that we could confirm if the presence of mismatch only happens with usability defect reports or other kind of reports as well.

ACKNOWLEDGMENT

Support for the first author from the Ministry of Higher Education Malaysia, Universiti Teknologi MARA (UiTM), the ARC Discovery Projects scheme project DP140102185, and from the Deakin Software and Technology Innovation Lab and Data61 for all authors, is gratefully acknowledged.

REFERENCES

[1] T. Zimmermann, R. Premraj, N. Bettenburg, C. Weiss, S. Just, and A. Schro, "What Makes a Good Bug Report ?," *IEEE Trans. Softw. Eng.*, vol. 36, no. 5, pp. 618–643, 2010.

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

[3] 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.

[4] L. Marks, Y. Zou, and A. E. Hassan, "Studying the fix-time for bugs in large open source projects," in *Proceedings of the 7th International Conference on Predictive Models in Software Engineering - Promise '11*, 2011, pp. 1–8.

[5] L. Tan, C. Liu, Z. Li, X. Wang, Y. Zhou, and C. Zhai, "Bug characteristics in open source software," *Empir. Softw. Eng.*, pp. 1–41, 2013.

[6] K. Moran, M. Linares-Vásquez, C. Bernal-Cárdenas, and D. Poshyvanyk, "Auto-completing Bug Reports for Android Applications," *Proc. 2015 10th Jt. Meet. Found. Softw. Eng.*, pp. 673–686, 2015.

[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] Z. M. Jiang, "Automated analysis of load testing results," in *ISSTA10 Proceedings of the 2010 International Symposium on Software Testing and Analysis*, 2010, pp. 143–146.

[10] A. Nistor, T. Jiang, and L. Tan, "Discovering, reporting, and fixing performance bugs," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, 2013, pp. 237–246.

[11] 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.

[12] S. Lal and A. Sureka, "Comparison of Seven Bug Report Types: A Case-Study of Google Chrome Browser Project," in *2012 19th Asia-Pacific Software Engineering Conference*, 2012, pp. 517–526.

[13] S. Davies and M. Roper, "What's in a bug report?," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '14*, 2014, pp. 1–10.

[14] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai, "Have Things Changed Now ? – An Empirical Study of Bug Characteristics in Modern Open Source Software," in *Proceedings of the 1st workshop on Architectural and system support for improving software dependability*, 2006.

[15] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, "Categorizing Bugs with Social Networks : A Case Study on Four Open Source Software Communities," in *Proceedings of the 2013 International Conference on Software Engineering*, 2013, pp. 1032–1041.

[16] M. B. Twidale and D. M. Nichols, "Exploring Usability Discussions in Open Source Development," in *Proceedings of the 38th Annual Hawaii Internatioal Conference on System Sciences*, 2005, pp. 1–10.

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

[18] T. S. Andre, H. Rex Hartson, S. M. Belz, and F. a. Mccreary, "The user action framework: a reliable foundation for usability engineering support tools," *Int. J. Hum. Comput. Stud.*, vol. 54, pp. 107–136, 2001.

[19] M. Theofanos and W. Quesenbery, "Towards the Design of Effective Formative Test Reports," *J. usability Stud.*, vol. 1, no. 1, pp. 27–45, 2005.

[20] M. G. Capra, "Usability Problem Description and the Evaluator Effect in Usability Testing,". PhD Thesis. Viginia Tech, Blacksburg, VA, 2006.

[21] K. Hornbaek and E. Frokjaer, "What Kinds of Usability-Problem Description are Useful to Developers?," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2006, vol. 50, no. 24, pp. 2523–2527.

[22] B. J. S. Dumas, B. R. Molich, and B. R. Jeffries, "Describing usability problems: Are we sending the right message?," *Interactions*, pp. 0–4, 2004.

[23] G. Cockton, A. Woolrych, L. Hall, and M. Hindmarch, "Changing analysts' tunes: The surprising impact of a new instrument for usability inspection method assessment," *Proc. HCI 2003 People Comput. XVII*, pp. 145–162, 2003.

[24] S. G. Vilbergsdottir, E. T. Hvannberg, and E. L. C. Law, "Assessing the reliability, validity and acceptance of a classification scheme of usability problems (CUP)," *J. Syst. Softw.*, vol. 87, pp. 18–37, 2014.

[25] 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 ASWEC 2015 24th Australasian Software Engineering Conference*, 2015, pp. 38–43.

[26] F. P. Simões, "Supporting End User Reporting of HCI Issues in Open Source Software,". PhD Thesis. Pontificia Universidade Catolica, Do Rio De Janeiro, 2013.

[27] J. Howarth, T. Smith-jackson, and R. Hartson, "Supporting novice usability practitioners with usability engineering tools," *Int. J. Human-Computer Stud.*, vol. 67, no. 6, pp. 533–549, 2009.

[28] S. L. Keenan, H. R. Hartson, Dennis G. Kafura, and R. S. Schulman, "The Usability Problem Taxonomy : A Framework for Classification and Analysis," *Empir. Softw. Eng.*, vol. 4, pp. 71–104, 1999.

[29] V. Lelli, A. Blouin, and B. Baudry, "Classifying and qualifying GUI defects," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation, ICST 2015 - Proceedings*, 2015.

[30] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, "Comparing Mining Algorithms for Predicting the Severity of a Reported Bug," in *2011 15th European Conference on Software Maintenance and Reengineering*, 2011, pp. 249–258.

[31] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, B. K. Ray, and D. S. Moebus, "Orthogonal Defect Classification - A Concept for In-Process Measurements," *IEEE Trans. Softw. Eng.*, vol. 18, no. 11, pp. 943–956, 1992.