Reporting Usability Defects – Do Reporters Report What Software Developers Need?

Nor Shahida Mohamad Yusop
School of Software and Electrical Engineering
Swinburne University of Technology
Melbourne, Australia
nmohamadyusop@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
Reporting usability defects can be a challenging task, especially in convincing the software developers that the reported defect actually requires attention. Stronger evidence in the form of specific details is often needed. However, research to date in software defect reporting has not investigated the value of capturing different information based on defect type. We surveyed practitioners in both open source communities and industrial software organizations about their usability defect reporting practices to better understand information needs to address usability defect reporting issues. Our analysis of 147 responses show that reporters often provide *observed result, expected result* and *steps to reproduce* when describing usability defects, similar to the way other types of defects are reported. However, reporters rarely provide usability-related information. In fact, reporters ranked *cause of the problem* is the most difficult information to provide followed by *usability principle, video recoding, UI event trace* and *title*. Conversely, software developers consider *cause of the problem* as the most helpful information for them to fix usability defects. Our statistical analysis reveals a substantial gap between what reporters provide and what software developers need when fixing usability defects. We propose some remedies to resolve this gap.

CCS Concepts
• Software and its engineering → Software creation and management → Software post-development issues → Maintaining software

Keywords
Defect repository; software quality; usability defect reporting.

1. INTRODUCTION
Software usability is one of the prominent software quality characteristics that determines acceptance of a software product in today's competitive market. Usability defects are an *unintended behavior* by the product that is *noticed by the user* and has an *affect on user experience*. According to Nielsen [1], good software should be easy to learn, efficient to use, allow rapid recovery from errors and be easy to remember. Typically, software companies manage and track defects using a central defect-tracking system where all defects are treated similarly in terms of information capture. Wilson and Coyne [2] argue that we should consider a different system for usability defects – but, do not offer specific guidance. They show that usability defects tend to get less priority compared with functional defects [2]. The most common reason for not properly addressing usability defects is either the software developers do not understand the reported problems or they do not consider the problem identified as valid. This is illustrated by this comment from an Ubuntu user:

> *'…Ubuntu developers either don't understand my usability reports or tag them as low priority bugs, which gets triaged for many released. Once I have submitted a bug report on usability issue that caused information loss, which is serious. In certain PDF files, I can't search for accented characters. This affects not only, say, evince search, it also affects tracker searches, for example. The main (non duplicate) bug for this was reported 2 years ago by lherrmann and, right now, it's tagged as confirmed/unknown, triaged/low*."

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 may cause confusion or problems for some people [3]. Let us consider a small clickable area of website hyperlinks in a touch screen device. For those who have small fingers, it is not a problem to click the hyperlinks, but for those with large fingers, clicking them might be. In this case, additional information is often needed to describe this finger touch problem.

However, research to date in software defect reporting has not investigated capturing of different information based on defect type, specifically usability defects. For example, [4]–[6] have surveyed software developers and users to identify useful information for software developer to fix defects. They identified information such as steps to reproduce, observed results and expected results are important on defect fixing. Nevertheless, they do not consider what information should be reported, and how the information should be presented when it come to other type of defect reports. Other work focuses on software repository mining to understand the structure and content of defect reports [3], [7]–[9], quality of defect reports [10], and use of defect data to develop prediction models [11]–[13] – however these too do not go into any specific issues related to usability defects.

Our research fills in this gap by focusing on usability defects, in particular among non-usability practitioners. We surveyed software development practitioners to investigate *"What*

EASE 2016, June 1-3 2016, Limerick, Ireland.

DOI: http://dx.doi.org/10.1145/12345.67890

information do reporters use to describe usability defects? and "What information do developers consider useful for fixing usability defects?". Such investigation set out to provide comprehensive view on the day-to-day practices when dealing with usability defects and pinpointing challenges. Through this study, researchers can find characteristics, open issues, and understand the nature of describing usability defects, which can be valuable for improving defect reporting processes and tools. Software development practitioners, in turn will also find technical references for reporting specific types of defects. Our study builds upon previous work that investigated defects in general [5], [6], and we add questions on many of the possible factors influencing content of usability defect reports. Both open source and industrial participants participated voluntarily in the survey.

In the next section our research methodology is described, and then we report our survey results in detail. We discuss the key findings from this survey, implications for usability defect reporting, and key threats to validity. We conclude with a summary of key findings and future directions for research.

2. RELATED WORK

Table 1 presents five studies that were characterized according to defect type, research method, population/ dataset, sample data, defect attributes studied and findings. As shown in Table 1, we found three studies that surveying software developers and reporters in order to investigate the most helpful information when reporting and fixing software defects. Zimmerman et al. [5] found the most helpful information for fixing software defects in open source projects are steps to reproduce, stack traces, test cases, screenshots, observed behavior and expected behavior. Laukkanen et al [6] replicated Zimmerman et al. survey in six industrial software organizations. They confirmed that steps to reproduce and observed behavior are the most important aspects of defect reports. However, they discovered that many defect reports lack this technical information. In contrast to [5] [6], Følstad [14] investigated usability defects from the perspective of usability practitioners. Their survey focused on how solution proposals are presented. The most prevalent materials used for presenting solution proposals are textual descriptions, annotated screenshots, UI digital mockups and oral presentations. Our survey, on the other hand, focuses on non-usability practitioners with varying knowledge on HCI (software tester, developers, project managers, consultant and etc), and we investigated all attributes of software defect instead of focusing on solution proposal only. Therefore, their survey only covers a small usability defect attributes that we are interested in

Other studies in defect reporting used defect reports mined from open source defect repositories. Davies and Roper [7] examined 1600 defect reports from Eclipse, Firefox, Apache and Facebook API to understand what information users provide in defect reports. They found that the most included information in defect reports is observed behavior followed by expected behavior and steps to reproduce. Similarly, Android-based apps defect reports often contains steps to reproduce and explanation of the difference between expected and the observed behavior [15]. In fact, Bhattacharya et al. reported a good quality of defect report is one that has long textual description of the problem. In the context of existing related work, our study replicated and extended the survey conducted by [5][6][14]. Our study makes new contributions in the following aspects:

Table 1. Related work on software defect reporting and our work

Criteria	**Zimmermann et al.** [5]	**Laukkanen et al.** [6]	**Følstad et al.** [14]	**Davies & Roper** [7]	**Bhattacharya et al.** [15]	**Our Work**
Defect type	General	General	Usability	General	General & Security	Usability
Research Method	Survey	Survey	Survey	Repositories mining	Repositories mining	Survey
Survey population/ Dataset	Software developers and reporters from Apache, Eclipse and Mozilla	Software developers from industrial	Usability practitioners	Eclipse, Firefox, Apache HTTP, Facebook API	Open source Android-based apps	Software developers and reporters from both open source and industrial
Respondents/ sample size	466	74	155	1600	53 940	147
Defect attributes studied	Product, component, version, severity, hardware, operating system, summary, build information, observed result, expected result, STR, stack traces, screenshots, code examples, error reports, test cases	Title, component, configuration, error reports, hardware, expected result observed result, operating data, part of the application, product, contact information, screenshots, severity, software context, stack traces, STR, test cases, test scripts, user input	Solution proposal	Observed result, expected result, steps to reproduce, error reports, stack traces, screenshots, code examples, test cases, build information, application code	Description length, steps to reproduce, observed result, expected result, additional information (such as user input and version)	Title, cause of the problem, software context, solution proposal, observed result, expected result, STR, severity, product, component, version, operating system, hardware

* STR = Steps to reproduce

- Our survey population includes both open source and industrial projects, while Zimmermann et al. [5] focused on open source projects, Laukkanen et al. [6] used only industrial software developers, and Følstad et al. [14] surveyed usability practitioners.
- Zimmermann et al. [5] and Laukkanen et al. [6] studied coarse-grained levels of defect attributes. That is, they only identified which attributes were considered useful. But in this study, we wanted to drill down into fine-grained level of usability defects by asking specific

questions on what information is included and how the information is presented for each selected attributes.

- We focus on usability defect reporting, while Zimmermann et al. [5] and Laukkanen et al. [6] studied software defect reporting in general.
- We added *video, audio, cause of the problem, UI event trace, proposed solution* and *usability principle* to the list of defect information. This additional information was based on the literature [14][16][17]. We removed attributes that do not directly map to usability defects.

3. STUDY DESIGN

3.1 Selection of Participants

We use a survey of practitioners to collect their current practices, challenges and perspectives. Our participants were selected from software development practitioners with varying experience levels and roles (including developers, testers, and managers). The respondents were recruited from both open source and industrial communities. For open source respondents, we advertised the survey through the community forum, such as Eclipse Community forums. While industrial respondents were invited through Facebook, LinkedIn, Software Testing Club[1] and researchers' industrial contacts. Participation was voluntary and participants were allowed to discontinue participation at any time during the research activity. The consent to participate in the survey was implied by the return of the anonymous questionnaire. However, a precise response rate cannot be determined, as the total number of the participants who received the invitation is unknown.

3.2 Questionnaire Design

The survey had 50 questions, split into seven sections. Around 14% of the questions on investigating usability defect attributes were derived from [5], [6]. While questions on the influential factors of defect reporting practices, like knowledge, experience, tools and methods were based on [18]. Other questions were formulated based on literature review. The questionnaires consisted of two versions: one for usability defects fixer (developers) and one for reporters. The sections are:

1) *Background information*: We collected general information about the respondents including gender, age ranges, employment information, and role in dealing with usability defects.
2) *Training/ certification in Human-Computer Interaction*: We asked both reporters and developers if they attended any Human-Computer Interaction and/ or usability training and how useful the training/certification was.
3) *Discovering usability defects*: We asked the reporters about their experience in software testing and method they used to discover usability defects. The respondents were also asked if they agreed (on a Likert scale) that the amount of information available for reporting usability defects varies according to how defects are discovered.
4) *Reporting usability defects*: We asked what information reporters usually provide, evidence they used to support their claim, and how usability defects are presented. This section also asked reporters to rank top five most difficult attributes to provide.
5) *Fixing usability defects*: We asked what information do software developers usually use when fixing usability defects and ranked the top five most importance attibutes. The software developers were also asked to indicate the problematic attributes that they have experienced and their opinions on the quality of defect report produced by different types of reporters.
6) *Defect reporting and automation tool*: We asked both reporters and software developers on their experience of using defect reporting/automation tools. Questions focused on tools used and their effectiveness to capture and manage usability defects. Other questions aimed to get opinions on the influence of experience, and knowledge in designing new defect reporting form.
7) *Knowledge and experience in usability defect reporting*: We asked both reporters and software developers about their view of experience and knowledge in usability defect reporting. The questions asked about the influence of level of experience, and whether different type of knowledge (usability/ software engineering, domain and technical) can affect the level of detail of defect reports. For space reasons, the results of this section will be published in future works.

We used an online survey using the Opinio survey tool. The survey was piloted with Swinburne Software Innovation Lab (SSIL) software engineers and fifteen software developers recruited during a developer conference (DDD Melbourne 2014). Based on the verbal comments and the pattern of responses received, the survey instruments were refined.

The survey is at http://bit.ly/UsabilityDefectsReportingSurvey. This survey study was approved on behalf of Swinburne's Human research Ethics Committee (SUHREC) by a delegated SUHREC subcommittee (SHESC2) (Approval number: SHR Project 2014/231).

3.3 Data Analysis

In this study, both quantitative and qualitative data were collected. For quantitative data, we used descriptive statistics. While qualitative data was analyzed using exploratory analysis [19]. We began by reading the respondents' comments, looking for key words, trends and themes. Next, the results of the analyses were used as supportive evidence for the quantitative results. Finally, we generated hypotheses for further study. The responses to the qualitative questions are discussed only briefly in this paper.

4. FINDINGS

The data was collected during June – November 2015. A total of 294 respondents attempted the survey. However, only 147 responses were included in this analysis. The remaining 50% responses were excluded for no response beyond the first parts of the questionnaire. One possible explanation of the high percentage of invalid responses is due to the *out of scope* problems, where the respondents are not in the target population. For example, the software development practitioners who do not have experience in dealing with usability defects would be not interested or they may find the questions are not relevant and return blank questionnaires.

4.1 Respondents Background

The majority of the respondents were (65.3%) male, with 34.0% female participants, and 0.7% of participants who did not indicate their gender. About 85% of the respondents were between 25 - 44 years of age. As shown in Table 2, majority of the respondents are software developers (40.9%) followed by software testers (14.8%)

[1] http://www.softwaretestingclub.com/forum

and project managers (10.0%). In terms of year of experiences, 63.8% of respondents had one to five years of work experience in their current position, while 25.3% had more than five years.

Table 3 shows respondents' knowledge on human-computer interaction (HCI). The vast majority of respondents do not receive any training usability-related training. However, for those who had acquired the related training, 84% believed the training was useful for to understand and report usability defects (cf. Table 4).

Table 2. Distribution of respondents across professional position and year of experience

Professional Position	Less than 1 year	Between 1 and 3 years	Between 3 and 5 years	More than 5 years
Software developer	4.1%	21.8%	7.5%	7.5%
Software tester	2.0%	5.4%	3.3%	4.1%
Quality assurance engineer	0%	0%	2.0%	0.7%
Customer consultant/ support	0%	0.7%	1.4%	0.7%
System engineer	0%	0.7%	0%	0.7%
Test manager	0.7%	0%	0.7%	0.7%
Project manager	0.7%	3.3%	3.3%	2.7%
Usability engineer	0%	0%	0.7%	0.7%
User interface designer	0.7%	0.7%	0.7%	0.7%
Other	2.7%	6.8%	4.8%	6.8%

Table 3. Distribution of respondents' knowledge on HCI

Role in dealing with usability defects	HCI related training		Total
	Yes	No	
Reporting usability defects	17	65	55.8%
Fixing usability defects	8	57	44.2%

Table 4. Responses on "usefulness" of usability-related training for reporting usability defects

Very useful	44.0%
Somewhat useful	40.0%
Neither useful or not useful	4.0%
Not very useful	12.0%
Complete waste of time	0%

4.2 Dealing with Usability Defects

Most respondents had experience reporting usability defects (55.8%), while 44.2% has experience fixing them. The majority of them have more than 5 years of experience in software testing (34.1%) (cf. Table 5). Table 6 summarizes the defect discovery methods used by the respondents to discover usability defects. Note that respondents were able to select more than one option. More than 60% of respondents indicated that they found usability defects when performing *exploratory testing*, *functional testing* and while *using the product*. A smaller proportion of respondents indicated that they discovered usability defects through *alpha/beta testing* (26.8%). From the free-text explanation, some respondents explicitly mentioned other methods including *focus groups, GUI testing, performance testing, heuristics evaluation* and *automated testing*.

Table 5. Years of experience in software testing

No experience	25.6%
Less than 1 year	6.1%
Between 1 and 3 years	22.0%
Between 3 and 5 years	12.2%
More than 5 years	34.1%

Table 6. Defect discovery methods

Exploratory testing	62.2%
Functional testing	63.4%
Usability testing	58.5%
Beta/ alpha testing	26.8%
Complaints/ reports from customers	53.7%
Using the product	62.2%
Other	7.3%

4.3 What do Reporters Usually Provide in Defect Report?

Question 13 asked respondents to indicate a frequency of supplying attributes listed in Table 7 when reporting usability defects. Based on "Often" and "Always" rating, the reporters mostly provide *title/ summary (81.8%), observed result (79.5%)* and *expected result (76.8%). steps to reproduce (74.4%)*, *software context (70.7%)* and *software information (70.7%)*. In order to understand the content of each selected attributes, for those who selected option other than "Never", we further asked questions about the elements and supportive materials that they used. For these subsequent questions, multiple answers were allowed and respondents could describe other details in a free text section (Question 14 – Question 24). Table 9 summarizes the responses.

Title/ summary is a headline summarizing the problem. When crafting a title, majority of the reporters explained the situation that was happening at the time the problem occurred (70.7%). Some of them (58.8%) prefer to copy and paste an error message. Only less than half of the reporters provided product details (such as build, version and operating system) (47.6%) and clarified on the quality issues that were affected (43.9%). As expected, most of the reporters who are able to provide quality issue are those who have software testing experience of more than five years.

Cause of the problem describes what a user is doing when he/ she discovers the problem or criteria used to justify the problem. Our findings indicated that cause of the problem is normally justified based on the reporter's knowledge (61.0%). About half of reporters (51.2%) were able to explain a design fault, such as how the interaction architecture may contribute to the problem. Nearly quarter (26.8%) of the reporters pointed out violated usability heuristics. Other information, as described in the free text, specified the consequence to the user (3.7%). To support this justification, the screenshot with accompanying text is the most widely used material other than annotated screenshot and textual description (Table 8). Other materials used by reporters included UI event trace (36.6%) and videos with captions (17.1%).

Software context addresses the location of the problem in the interface. As expected, majority of the reporters were mentioned

the name of the defective interface, such as screen title (76.8%) and problematic user interface object (70.7%). Some respondents (58.5%) describe user's task to indicate the context of usage. In terms of specifying the components affected, only 41.5% of reporters were able to supply the information. This information was often conveyed in annotated screenshots (82.9%) followed by textual description (64.6%) – cf. Table 8.

Proposed solution describes a recommendation to remedy the usability defects. 56.1% (cf. Table 8) of reporters addressed the proposal solutions as a way to improve the desired software behavior, rather than to correct a defective software behavior (46.3%). Reporters mentioned that these recommendations were originated from their knowledge (45.1%). To support these recommendations, only a few reporters provided alternate solutions (37.8%), advantages and disadvantages for alternative solutions (28.0%), and usability design principles (26.8%). Additionally, nearly half of respondents (48.8%) prefer to use textual form to explain the recommendations. Some of them supported these recommendations with the annotated screenshot (39%), and simple sketches (28%). While, 25% of reporters prefer to demonstrate their recommendations through oral presentations.

Observed results describes the actual results that differ from the reporter's expectation or violating specification. As indicated in Table 8, the observed results were usually described based on the effect on a user's performance (65.9%). In this case, the justification on what was wrong and why it is wrong (53.7%) were supplied. Others explained the user's behavior by outlining the issues (50%). To support the claimed issues, reporters preferred to attach annotated screenshots (64.6%) followed by error messages (57.3%).

Expected results describes what reporters expect the software should response. The majority of the reporters used their knowledge and experience to interpret the intended results (72%). Some reporters mentioned usability requirements (59.8%) and usability design guidelines (40.2%) that were used to justify their expectations.

Steps to reproduce outlines the instructions suggested by reporters so that software developers can reproduce the discovered usability defects. The majority of reporters write a textual description on the user's navigation flow through the system (72%). About one-third recorded the actual steps that they followed and attached a link to the video (30.5%). Other supplementary information, as specified in the free text, included logs and even event traces (3.7%).

Severity indicates the level of effect the usability defects had on the user. In order to rate the severity level, most reporters considered the impact of the issue (74.4%). While others examined the frequency of issues occurred during usage (61%) and business impact (48.8%).

In Question 27, we asked respondents to rank top five attributes in order of difficulty from 1 to 5 where 1 is the most difficult and 5 is least difficult. Among the attributes considered to be most difficult to provide, respondents ranked *cause of the problem, usability principle, video recording, UI event trace* and *title.*

Table 7. Defect attributes used to report usability defects – title, observed and expected result are the most provided attributes by reporters

Items	Never	Rarely	Sometimes	Often	Always
Title/ Summary	1.2%	2.4%	14.6%	25.6%	56.2%
Cause of the problem	3.7%	6.1%	23.2%	23.2%	43.8%
Software context	1.2%	3.7%	24.4%	24.4%	46.3%
Proposed solution	9.8%	12.2%	32.9%	26.8%	18.3%
Observed result	1.2%	3.7%	15.9%	20.7%	58.5%
Expected result	1.2%	7.4%	14.6%	26.8%	50.0%
Steps to reproduce	2.4%	6.1%	17.1%	31.7%	42.7%
Severity	1.2%	18.3%	20.7%	22.0%	37.8%
Software information (product, component, version)	0%	9.8%	19.5%	29.3%	41.4%
Test environment (Operating system, hardware, browser)	0%	9.8%	26.8%	29.3%	34.1%

Table 8. Materials used to support usability defect description – the most prevalence material is screenshots with annotations

Material	Cause of the problem	Software context	Proposed solution	Observed result
UI event trace	36.6%	-	-	-
Screenshot and accompanying text	75.6%	1.2%	-	-
Screenshot with annotations	62.2%	82.9%	39.0%	64.6%
Textual description	61.0%	64.6%	48.8%	-
Video with captions	17.1%	18.3%	-	-
Fix patch	-	-	12.2%	-
Digital mockups	-	-	11.0%	-
Simple sketches	-	-	28.0%	-
ASCII art	-	-	-	-
Graphical elements or code	-	-	-	-
Error messages	-	-	-	57.3%
Oral presentation	-	-	25.6%	-
Other – audio video	2.4%	1.2%	3.7%	1.2%

Table 9. Details explanation included for each attribute

Items	%
Title/ Summary	
Explanation on the situation that was happening at the time the problem occurred	70.7%
Build or version of the software or operating system on which the problem occurred	47.6%
An error message that come up	58.5%
Quality attributes affected	43.9%
Other – Not mentioned	4.9%
Cause of the problem	
Heuristics that are violated	26.8%
Design fault	51.2%
My knowledge of performing and understanding a task or object interface	61.0%
Other – consequence to the user	3.7%
Software context	
Location of the problem in the interface, such as screen title	76.8%
The problematic user interface object, such as button, menu and dialogue box	70.7%
User's task	58.5%
System components that are affected	41.5%
Other – Not mentioned	2.4%
Proposed solution	
Alternate solutions	37.8%
Advantages and disadvantages for alternatives solutions	28.0%
Usability design principles and/ or previous research	26.8%
My knowledge to interpret how design is supposed to work	45.1%
The expected behavior (defects)	46.3%
The behavior desired (enhancements)	56.1%
Other – as suggested by project manager	2.4%
Observed result	
The effect on the user's performance	65.9%
The user's behavior following the issue	50.0%
What was wrong and why is it wrong	53.7%
Other – user experience	1.2%
Expected result	
Usability requirements	59.8%
Knowledge/ experience to interpret the expected result	72.0%
Usability design guideline	40.2%
Other – Not mentioned	3.7%
Steps to reproduce	
User's navigation flow through the system	72.0%
Record the steps	30.5%
Other - logs	3.7%
Severity	
Impact of the issue	74.4%
Business effects, such as costs and time loss	48.8%
Frequency of the issue occurs during usage	61.0%
Other – Not mentioned	2.4%

4.4 How are Usability Defects Reported?

As shown in Table 10, nearly half of our respondents used written reports (50%), verbal meetings (53.7%) and defect reporting tools (53.7%) as a medium for their defect reporting (Q25). Only a few respondents used edited video for reporting purposes. For those who have used a defect-reporting tool, we asked respondents to mention their tool (Q36). The most commonly used defect reporting tools reported by our respondents were *JIRA, Bugzilla* and *Redmine*. *Mantis, HP Quality Center, Trello, IBM Rational Team Concert, HP Application Lifecycle Management* and *Visual Studio TFS* were listed multiple times. For JIRA, Bugzilla and Redmine users - 90% of them agreed to some extent that the tool offers sufficient flexibility to capture and manage usability defects (Q37), but free-text feedback revealed considerable negative satisfaction (Q38). The following are representative: *"Most of the defect reporting tool do not have exhaustive options for usability defects"* and *"JIRA more customized by client but no specific customizations done for usability*".

Some respondents nominated specific recommendations for usability defect reporting tool improvements (Q50). For example, they argued that video evidence can reduce the amount of time to reproduce and describe the issues, especially when working with offshore development teams. One respondent also suggested a questionnaire feature.

4.5 What do Developers Actually Need?

As shown in Table 11, question Q28, Q29 and Q30 asked software developers to rate a frequency of using textual information and supplementary information respectively when fixing usability defects. Based on the "Often" and "Always" rating, *cause of the problem* (83.1%,), *steps to reproduce* (81.6%), *software context* (78.5%), *expected result* (78.5%) and *observed result (73.8%)* were the most widely used textual information. Three least used textual information were: *title/ summary (50.8%)*, *hardware (55.4%)* and *severity (56.9%)*. About half the developers seldom used title/summary. One possible explanation can be that title/summary contains limited information for understanding the likely difficulty faced by users. Also, Meanwhile, severity may only be useful for prioritizing defects to be fixed but it does not provide input to solve the problem. Supplementary information which was most frequently used for fixing usability defects were: *screenshots (83.1%)*, *UI event trace (56.9%)* and *patch (41.5%)* (Table 11). Even graphical elements such as *ASCII art (9.3%)* and *digital mockups (23.1%)* provided rich source of proposed fix, but software developers rarely used them.

For the importance of information (Question 31), *cause of the problem* clearly stands out. This is followed by *screenshot, steps to reproduce, expected result, software context* and *proposed solution*. Similar to the findings from [5][6], in order to fix usability defects, mandatory fields such as hardware, product, component and severity were not of much value. This does not mean the information is not useful; rather they might be used in a different context for different purposes.

In Question 32, we asked software developers to select problematic attributes supplied by reporters. Note that software developers were able to select more than one option. As shown in Table 12, among the problems experienced, *unclear cause of the problem* (76.9%) and *insufficient information in steps to reproduce* (72.3%) was the most commonly encountered. Other common problems include *unclear software context* (46.2%) and *screenshots* (46.2%). Apart from lacking of technical information, the software developers also received *vague comment* (52.3%), *unstructured text* (55.4%) and *duplicate defect reports* (50.8%). While the low occurrence of spam and viruses confirms [5].

Table 10. Medium to report usability defects – respondents mostly used defect-reporting tool and discuss through verbal meeting

Medium of reporting	Never	Rarely	Sometimes	Often	Always	Not answer
Traditional written report	13.4%	11.0%	12.2%	22.0%	28.0%	13.4%
Verbally in a meeting with designers/ developers	3.7%	6.1%	23.2%	42.7%	11.0%	13.4%
Edited videos	31.7%	31.7%	13.4%	8.5%	1.2%	13.4%
Entry in defect reporting tool	7.3%	6.1%	19.5%	15.9%	37.8%	13.4%

Table 11. Frequency of attributes used to fix usability defects – the most useful attribute is cause of the problem

Items	Never	Rarely	Sometimes	Often	Always
Textual information					
Title/ Summary	3.1%	13.8%	32.3%	15.4%	35.4%
Cause of the problem	3.1%	0%	13.8%	24.6%	58.5%
Software context	4.6%	4.6%	12.3%	27.7%	50.8%
Proposed solution	3.1%	10.8%	26.2%	46.1%	13.8%
Observed result	3.1%	1.5%	21.5%	32.3%	41.5%
Expected result	3.1%	4.6%	13.8%	33.8%	44.7%
Steps to reproduce	1.5%	3.1%	13.8%	27.7%	53.9%
Severity	3.1%	7.8%	32.3%	21.5%	35.4%
Product	7.8%	6.2%	20.0%	36.9%	29.2%
Component	1.5%	10.8%	26.2%	38.5%	23.1%
Version	0%	6.2%	29.2%	23.1%	41.5%
Hardware	0%	18.5%	26.2%	32.3%	23.1%
Operating system	0%	12.3%	29.2%	27.7%	30.8%
Supplementary information					
Usability principle/ violated heuristic	9.2%	15.4%	43.1%	20.0%	12.3%
Video recording	16.9%	15.4%	30.8%	24.6%	12.3%
Audio recording	21.5%	35.4%	26.2%	10.8%	6.2%
UI event trace	6.2%	0%	36.9%	35.4%	21.5%
Screenshots	1.5%	3.1%	12.3%	29.2%	53.9%
Fix patch	3.1%	13.8%	41.5%	21.5%	20.0%
Digital mockups	15.4%	24.6%	36.9%	10.8%	12.3%
ASCII art	30.8%	33.8%	26.2%	7.8%	1.5%

Table 12. Problems with usability defect reports – unclear cause of the problem and insufficient information in steps to reproduce were the most common encountered problems experienced by software developers

Problems	Attributes	Frequency	Problems	Attributes	Frequency
You were given unclear	Title/ summary	38.5%	There was insufficient information in	Steps to reproduce	72.3%
	Cause of the problem	76.9%		UI event trace	38.5%
	Software context	46.2%	The reporter used	Bad grammar	27.7%
	Usability principle/ heuristic violated	21.5%		Unstructured text/ format	55.4%
	Proposed solution	29.2%		Vague comment	52.3%
	Screenshots	46.2%		Too long text	32.3%
	Audio recording	20.0%		Non technical language	35.4%
	Video recording	16.9%		Usability jargon/ term	20.0%
You were given incorrect	Component	16.9%	Other	Duplicate	50.8%
	Observed result	38.5%		Spam	23.1%
	Expected result	44.6%		Viruses/ worms	15.4%
	Product	23.1%			
	Version	32.3%			
	Severity	23.1%			
	Hardware	20.0%			
	Operating system	27.7%			

5. DISCUSSION

5.1 Reporting Usability Defects

The responses from the survey indicated *title/summary, observed result, expected result, steps to reproduce, software context,* and *software information* are often supplied when reporting usability defects. However, we argue that these findings are biased in a way the defect form is designed. In Bugzilla, for example, by default the defect form contains title/ summary, software information (i.e. product, component and version), steps to reproduce, actual results, expected results and

attachment. Therefore, the use of this generic defect reporting form will produce the same content structure for all types of defects. To address this concern, in this survey we explicitly identify what detailed information reporters provide for each textual attribute and what other materials are attached to support usability defect description.

Even though most of the reporters can produce a relatively complete defect description, usability-related information is rarely included. For instance, only 26.8% of the reporters were able to augment the cause of the problem by relating the issue with violated usability heuristics. In terms of describing proposed solutions and expected results, barely a quarter of the reporters included usability design principles to justify their idea. This is possibly due to the fact that the majority of the reporters do not have sufficient knowledge on relevant usability principles. Moreover, the shortcoming of the existing generic defect report form does not assist users in reporting a clear usability defect description. Perhaps, a defect report form should be designed as a wizard-style guided-answering form that consists of necessary information. A good example is the question-based structure form proposed by Simões [20]. The form contains of six questions to report HCI issues. However, a lack of pre-defined usability attributes for input selection can be explored for future work.

Many respondents considered *cause of the problem, usability principles, video recording, UI event trace* and *title/ summary* are the most difficult items to provide. One possible reason for that may be reporters lack background to provide this information. Since UI event traces, for instance, are not readily available to end users, reporters may need to take extra steps. In this circumstance, reporters with deficient programming skills may only report problems in a graphical user interface (GUI) rather than a sequence of events that can be mapped into user tasks. This corresponds with Wang et al. [21]'s examination of open defect reports, which showed that most of the defect reports that contains technical information are often submitted by highly technical reporters. Moreover, lack of automated tools and limited types of supported recording and attachments in existing defect reporting tools make it challenging to include video and audio files [22].

As shown in Table 10, many reporters mentioned verbal discussion with developers as a common approach to communicate usability defects. In our opinion, as Andre et al. [23] addressed, verbal communication alone is not sufficient to resolve usability defects without a written description. This is because every software defect needs to record evidence and keep formal logs. In this case, the use of formal reporting tool, such as defect tracking systems were considered particularly useful to track and manage defects in a timely manner. However, as our survey results indicated, defect descriptions suffer from insufficient information for problem correction. One possibility is due to the text-centric approach used by the defect tracking system. The unstructured text may contain a mixed of data such as cause of the problem, proposed solution and impact. This results in unorganized data and ambiguities that make it difficult to understand the whole issues as compared to data stored in fielded form. Furthermore, it is important to consider the subjective nature of the usability defects that may not be easy to explain textually [24]. Therefore, additional information in the form of attachable files may be required to complement this deficiency. We have identified four categories of attachable files: (1) screenshots, (2) videos, (3) graphical elements such as ASCII art, and (4) UI event traces / error messages. In this survey, we found that reporters tend to use screenshots, rather than videos, UI event traces and error messages when highlighting the cause of the problem, software context and observed results. Whereas, to propose a design solution, reporters often used textual descriptions as compared to graphical elements (i.e, digital mockups, simple sketches and ASCII art). Possibly, the low rate of use of non-textual media is due to the fact that good drawing tools may not be readily available [25] and producing graphical representations using a toolkit may require extra skill and time to learn and use [3].

5.2 Fixing Usability Defects

The most widely used textual information mentioned by software developers are *cause of the problem, steps to reproduce, software context*, *expected result* and *observed result*. Similar to previous studies [5], [6] that focused on general defects, *steps to reproduce, expected result* and *observed result* were also common to be used by developers when fixing usability defects. In particular, it indicates that *steps to reproduce*, *expected result* and *observed result* are fundamental pieces of information for understanding all types of defects. In contrast to previous studies [5], [6], we added *cause of the problem*, *usability principles* and *proposed solution* to the list of defect information to reflect relevant attributes for fixing usability defects. Out of these items, *cause of the problem* was selected as the most useful and important attribute for fixing usability defects. Hence, our research extends the knowledge from the previous research.

Since our study is solely focused on usability defects that are primarily dealing with graphical elements, *screenshot* was rated as the most widely used supplementary information other than video, ASCII art, digital mockups and patch. When looking at the low frequency used of video, ASCII art, digital mockups and patch, we can explain that this is probably due to the reporters rarely supplying such information to justify the problems and propose their suggestions. Therefore, in the absence of this information, it is not surprising that software developers rarely use this information when fixing usability defects. In [5][6], software context was not considered useful, but was rated the third of most widely used attributes in our research context. We think that difference in the results comes from the different definition we introduce for software context. In our study, we refer software context as the specific location of problem in the interface where the problem was observed, such as button, menu and dialogue box. In contrast, [5][6] defined software context as the operating environment where the problem occurred, such as web application.

A number of problems in the way usability defects are described were identified. *Unclear cause of the problem* and *insufficient information in steps to reproduce* was very strongly identified as problematic defect attributes. This is not surprising because, as noted above, the reporters are indeed difficult to explain the cause of the problem clearly. In fact, the inability of reporters to supply UI event traces in defect reports will limit the essential information regarding user behavior and task sequences with respect to the application's user interface. In the absence of this technical information, such as the time to complete certain task, number of erroneous action sequences and usage of certain functions may cause software developers to misinterpret the usability problems [26]. Another problem to consider is *"vagueness"*, a similar issue raised by Dumas et al. [27]. In our study, about half of the software developers claimed that they received vague comments. These either did not have precise problem descriptions or the solutions suggested were too

general. According to Dumas et al., when describing usability problems or giving fix suggestions, reporters should be specific as they can, and do not let software developers make self-judgment. For instance, instead of suggesting "use a color with better contrast to the background and increase the font clarity" one could precisely suggest color contrast theory - black text on a brown background, for example.

5.3 Mismatch Between What Reporters Provide and What Software developers Need

We compared the responses obtained from reporters and software developers to find out whether reporters provide sufficient information for software developer to fix usability defects. In Table 13, attributes are ranked based on the mean of response to the questions "*Do you use the following items when describing usability defects*" and "*Do you use the following items when fixing usability defects*" that range from 1 (Never) to Always (5). To discover the level of agreement between what reporters provide and what software developers need, we measure the absolute value of differences between reporters' and developers' mean. The lowest difference indicates a more agreements and vice versa. As shown in Table 14, reporters and developers are in agreements on severity, software context, and expected result. However, more disagreements were observed. The most notable ones are *title/ summary* and *cause of the problem*. While our study and [6] identified *title/summary* is the least problematic information, but *title/summary* is not really needed by software developers to fix usability defects, as it was ranked as the second lowest. On the contrary, the *cause of the problem* that is expected to be presented in usability defect descriptions is seldom provide by reporters. In summary, our experiments suggest that reporters do not provide information that is frequently used by software developers.

Table 13: Rank of attributes

Rank (based on mean)	Reporter	Developer
1	Title/ summary	Cause of the problem
2	Observed result	Steps to reproduce
3	Expected result	Software context
4	Software context	Expected result
5	Steps to reproduce	Observed result
6	Software information	Software information
7	Cause of the problem	Severity
8	Test environment	Test environment
9	Severity	Title/ summary
10	Proposed solution	Proposed solution

Table 14: Agreement level between what reporters provide and what software developers need

Item	Mean ($\bar{x}$)		Differences of mean
	Reporter	Developer	
Severity	3.77	3.78	0.01
Software context	4.11	4.15	0.04
Expected result	4.17	4.12	0.05
Test environment	3.88	3.68	0.20
Software information	4.02	3.82	0.20
Steps to reproduce	4.06	4.29	0.23
Observed result	4.32	4.08	0.24
Proposed solution	3.32	3.57	0.25
Cause of the problem	3.98	4.35	0.37
Title/ summary	4.33	3.66	0.67

5.4 Threats to Validity

Internal validity. The main threat to this study is a misunderstanding of the survey context by the respondents. Our goal is to focus on usability defect reporting instead of general software defect reporting. Respondents may have answered the questionnaire based on their general defect reporting knowledge and experience. We addressed this threat by (a) giving three different types of usability defect examples at the beginning of the survey, (b) highlighting the ***usability defects*** (bold and italic) keyword for every question, so the respondents were always aware of the survey context.

External validity. One possible external threat to the validity of the survey outcome is the representativeness of the respondents. While the respondents were recruited from a range of software practitioners, there is the possibility that the software developers and testers responding have not used formal defect reporting processes and tools. Therefore, there is possibility of response bias towards providing answer and feedback. Due to the sample size of the survey, the generalizability of the results is limited. We mitigated this by incorporating open source and closed source projects and different kinds of software systems.

Construct validity. One concern is regarding incorrect measures, i.e. not precisely measuring respondents' practices in reporting usability defects. To mitigate this concern, we reused previous surveys and added questions from both usability and software engineering fields. Another possible threat is that our respondent recommendation does not entirely reflect the true reality of defect reporting practice. Since our survey is anonymous, some responses we received stated that they have never used defect reporting tools. In fact, some comments are not meaningful.

6. CONCLUSION AND FUTURE WORK

We conducted a survey among open source software communities and industrial practitioners to understand the most valuable information in reporting and fixing usability defects. We extended previous studies [5],[6] that focused on software defects in general. We added *cause of the problem, software context* and *proposed solution* in context of usability-related defect information. Our study extends the previous findings on software defect reporting. We discovered that developers need additional defect information when fixing usability defects. We found that *cause of the problem* is the most useful, but seldom supplied by reporters. Our results confirm that *observed results*, *expected results* and *steps to reproduce* are also substantially important for software developers.

According to reporters, they usually provided *title/ summary, steps to reproduce, observed result and expected result.* While usability-related information: *cause of the problem, video recording, UI event trace* and *usability principle* are the most difficult information to provide. When we compare the responses from software developers and reporters, we found the information most expected by the software developers was the least provided by reporters. The most significant one are *title/ summary* and *cause of the problem.* Our statistical analysis shows a mismatch between what reporters reported and software developers claim that they need to fix usability defects.

Our results showed that *unclear cause of the problem* and *insufficient information in steps to reproduce* were most commonly experienced by software developers. This reaffirms with evidence an anecdotal expectation that *cause of the problem* is difficult to provide. Other problems include vague comments, unstructured text and duplication of reported

usability defects. We plan to further investigate the potential usefulness of custom defect reporting forms for different kinds of usability defects. We want to determine better ways of reporting such defects, particularly to identify factors that influence a defect-type-specific form. A better understanding of usability defect report characteristics is necessary as this may identify redundant information as well.

7. ACKNOWLEDGEMENT

Support for the first author from the Ministry of Higher Education Malaysia, Universiti Teknologi MARA (UiTM), and from the Swinburne Software Innovation Lab and the National ICT Australia for all authors, is gratefully acknowledged.

8. REFERENCES

[1] J. Nielsen, *Usability Engineering*, vol. 44. 1993, p. 362.

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

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

[4] N. Bettenburg, C. Weiß, S. Just, and A. Schröter, "Quality of Bug Reports in Eclipse," in *Proceedings of the 2007 OOPSLA workshop on eclipse technology eXchange*, 2007, pp. 21–25.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] S. Breu and J. Sillito, "Information Needs in Bug Reports : Improving Cooperation Between Developers and Users," in *The 2010 ACM Conference on Computer Supported Cooperative Work*, 2010, pp. 301–310.

[10] P. Bhattacharya and I. Neamtiu, "Bug-fix time prediction models: Can we do better?," in *Proceedings - International Conference on Software Engineering*, 2011, no. 3, pp. 207–210.

[11] L. D. Panjer, "Predicting Eclipse Bug Lifetimes," in *Fourth International Workshop on Mining Software Repositories (MSR'07)*, 2007.

[12] P. Hooimeijer and W. Weimer, "Modeling Bug Report Quality," in *Proceedings of the 22nd IEEE/ACM international conference on Automated software engineering*, 2007, pp. 34–43.

[13] P. Anbalagan, M. Vouk, C. Science, and N. Carolina, "An Empirical Study of Security Problem Reports in Linux Distributions," in *Third International Symposiumm on Empirical Software Engineering and Measurement*, 2009, pp. 481–484.

[14] A. Følstad, P. O. Box, E. L. Law, K. Hornbæk, and S. Copenhagen, "Analysis in Practical Usability Evaluation : A Survey Study," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012, pp. 2127–2136.

[15] 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.

[16] K. Hornbæk and E. Frokjær, "Comparing usability problems and redesign proposals as input to practical systems development," in *CHI 2005: Technology, Safety, Community: Conference Proceedings - Conference on Human Factors in Computing Systems*, 2005, pp. 391–400.

[17] K. Hornbæk and E. Frøkjær, "Comparison of techniques for matching of usability problem descriptions," *Interact. Comput.*, vol. 20, no. 6, pp. 505–514, Dec. 2008.

[18] J. Itkonen and C. Lassenius, "The Role of the Tester ' s Knowledge in Exploratory Software Testing," *IEEE Trans. Softw. Eng.*, vol. 39, no. 5, pp. 707–724, 2013.

[19] G. Guest, K. MacQueen, and E. Namey, *Introduction to applied thematic analysis*. London, UK: Sage, 2012, pp. 3–20.

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

[21] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information," in *Proceedings of the 30th international conference on Software engineering*, 2008, pp. 461–470.

[22] N. Shahida, 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.

[23] 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.

[24] 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.

[25] G. Çetin, D. Verzulli, and S. Frings, "An Analysis of Involvement of HCI Experts in Distributed Software Development: Practical Issues," *Online Communities Soc. Comput.*, vol. 4564, pp. 32–40, 2007.

[26] Y. Tao, "Grammatical analysis of user interface events for task identification," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8517 LNCS, pp. 197–205, 2014.

[27] 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.