Preliminary Evaluation of a Guided Usability Defect Report Form

Nor Shahida Mohamad Yusop
Faculty of Computer and Mathematical Sciences
Universiti Teknologi MARA
Shah Alam, Selangor, Malaysia
nor_shahida@tmsk.uitm.edu.my

John Grundy
Faculty of Information Technology
Monash University
Melbourne, Australia
john.grundy@monash.edu

Jean-Guy Schneider
School of Software and Electrical Engineering
Swinburne University of Technology
Melbourne, Australia
jschneider@swin.edu.au

Rajesh Vasa
Faculty of Science, Engineering, and Built Environment
Deakin University
Melbourne, Australia
rajesh.vasa@deakin.edu.au

***Abstract*— Open source software development projects typically use generic defect reporting forms, such as BugZilla and Jira, even for specialized kinds of defects such as usability issues. These issues typically tend to be reported as unstructured text, mix multiple defect attributes, lack precise cause, and overlook emotional impact on users. The poor quality of these defect reports impacts on the ability of developers to address them efficiently and effectively. In this paper, we describe a novel guided defect reporting method for capturing usability defects. We evaluate our approach using an expert judgment approach where experienced developers and researchers evaluate the clarity of usability defects reported via conventional defect reporting tools and our method. Our results show this is a promising approach at capturing more clear usability defect descriptions.**

Keywords—Bugzilla, expert judgment, jira, usability defect reporting

I. Introduction

In typical software development environments, usability defects are reported, tracked, and managed similar to other functional defects using centralized defect reporting tools such as Jira and Bugzilla. Usability issues are often diluted among non-usability defects and end up with a low severity rating [1]. There are different reasons for this spanning from processes, communications, role of reporters, all the way to the subjective nature of usability defects [1]–[3]. In this paper, we focus on improving the capture of usability defect information. Although finding usability defects is an important part of a usability testing process, communicating the defects effectively is critical for effective and efficient resolution. Research on end-user reporting indicates that even though users are fairly successful in finding usability issues, the quality of usability issue is low [4], [5].

In the context of open source software (OSS), multiple studies argue that existing defect reporting tools are inconvenient to report usability defects [6]–[8]. The use of generic and brief defect reports do not help OSS communities, which have varying levels of technical knowledge, to create informative usability defect descriptions, thus making it more challenging for them to gain an understanding of what kinds of information is useful to software developers in order to fix the usability issues [9]. Furthermore, unstructured text with multiple defect attributes results in low quality reports.

These limitations motivated us to design a new usability defect report form to specifically capture usability issues for Bugzilla. Based on our review of the literature and analysis of different requirements gathered from our survey of practitioners and open source software defect repositories mining [9]–[11], we designed a guided defect report form for capturing usability defects. We aimed to maximize the alignment between the information needed by OSS developers and the information that could be provided by users. Our evaluation focused on assessing the clarity of the usability defect descriptions generated using our proposed defect report form. This was done by conducting a study focusing on expert review of a selection of usability reports submitted via the form. In this paper, we describe our proposed guided usability defect reporting form design, and then describe our study followed by our findings.

II. Related Work

Software defect reporting has been studied for many years [12]–[15]. Most studies to date focus on defect reporting tools, processess, quality, and predominantly study functional defects [16]–[19]. Usability defect reporting, on the other hand, has received much less attetion in the software engineering community.

Most existing defect reporting tools provide generic defect reporting interfaces, which are often used in their default settings. For example, the defect reporting in Bugzilla form supports custom fields, but has limited flexibility and the expectation of using of plain textual defect description limits our ability to describe graphical issues, especially the dynamic aspect of usability issues that cannot be explained using a single screenshot [20]. The vague definition of a general *description* field in typical defect reporting forms is also one of the reasons that testers often document a mix of information in this field [14]. This is because the reporter may not have a sufficient appreciation of what is supposed to be written in the *description* field. They may write whatever comes to mind, and in many cases the information reported is not important, useful, or helpful enough for the software developer to address the defects [9], [12]. To overcome these limitations, a few studies have developed automated support for interaction capture, trace capture, and auto-complete defect reports [18], [21], [22]. However, in OSS environments, these kind of developer-centric defect reporting solutions are not an effective solution as many end-users have varying levels of usability knowledge and prefer a defect reporting system that is less technical and straight forward [6].

In usability engineering research, many studies have taken the initiative to improve the quality of usability defect report content [23], [24]. However, these initiatives are often documentation heavy and include comprehensive content with many supporting details, some of which are not valued by software developers. As an alternative to the heavy-documentation approach, Howarth et al. [25] have developed the form-based approach - Data Collection, Analysis, and Reporting Tool (DCART) for collecting usability data from lab-based usability evaluations. The DCART provides additional support for novice usability evaluators to appraise of what important information should be provided. Simões [8] designed a new defect report template by exploring the needs of designers in open source projects. Their contribution shows a positive solution for eliciting the information needed by OSS designers. However, the use of an unstructured, open text forms still produced incomplete, ambiguous, and irrelevant information. This was because not all questions were relevant for different types of problems, and such open-ended questions may produce non-informative descriptions.

Since usability engineering reports are typically very detailed and contain many usability jargon terms, specialist interpretation and specialized usability training and knowledge is required to effectively use the reports [26]–[28]. However, in OSS engineering communities typically consist largely of volunteers, with few usability and Human-Computer Interaction (HCI) experts. Furthermore, the large time and expense required for most formal usability engineering studies if often not prioritized in open source communities [29]. Key insights in usability defect practices within OSS development are presented in Table 1.

TABLE I. SUMMARY OF PREVIOUS STUDIES TRANSLATED INTO FEATURES FOR A PROPOSED USABILITY DEFECT REPORT FORM

Online Survey [9] and Systematic Literature Review [10]
• Problem *description, severity*, *context,* and *redesign description* were the four attributes most commonly used to describe usability defects.
• A desirable usability defect report form should be simple, have reasonable predefined values, and introduce usability keywords, options and descriptors.
• Reporters often provide *observed results, expected results,* and *steps to reproduce* when describing usability defects.
• Reporters ranked *assumed cause* as the most difficult information to provide, but this was considered to be the most helpful information by software developers.
• The top five most important attributes used by software developers is *assumed cause, screenshots, steps to reproduce, excepted results*, and *software context.*
• The most experienced problem with usability defect reporting encountered by software developers were *unclear assumed cause* and *insufficient information in steps to reproduce.*
Software defect repository mining [11]
• *Supplementary information* can improve defect resolution time
• The most widely reported attributes in usability defect reports are *actual output, expected output*, and *software context.*
• *Solution proposals* were commonly described in words.
• *Assumed cause* is rarely reported in usability defect reports.
• Impact on human emotion and task performance was common to explain usability defects.

III. GUIDED USABILITY DEFECT REPORTING FORM DESIGN

Motivated by the fact that software developers and reporters prefer to use web based defect reporting tools over bundle applications [6], our usability defect reporting form was designed to be like existing open source defect reporting form structures. This was to reduce learning curve and permit easier integration. We used Bugzilla defect report form layout as a point of reference, as it is a well-known open source defect reporting tool.

A. Form Design Criteria

Our strategy for designing the usability defect report forms consisted of the following criteria, based on the findings of the three studies that we conducted earlier [9]–[11]. Based on the key findings listed in Table 1, we designed our new usability defect report forms with the following criteria:

Orthogonal defect descriptions - defect description attributes only capture one value. Rather than unstructured text descriptions, different attributes are separated into their own entry with guidance on expected values rather than being bundled together.

Explaining a defect attribute with multiple instances - Single attributes in existing defect report forms are inadequate for reporting usability defects. For example, while defect report forms in the Mozilla project explicitly have three separate attributes to capture (i) steps to reproduce, (ii) actual results, and (iii) expected results, often the information that really is needed was not explained clearly or not even reported at all. While some reporters may think that their reports are well explained, they may provide inadequate information.

We propose several usability defect attribute instances to assist reporters in supplying the most useful information in their usability defect reports. Even breaking up an attribute into multiple instances does not always provide a better user experience and produce high quality defect information, but this kind of form design provides specific hints and examples about the information the user should enter. These attribute instances depend on the context of information to be described. For example, to explain the *impact* of the problem, we introduced three instances – *How does this problem affect your task? Explain your challenges*, and *How annoying is this problem to you?* Table 2 lists our guided reporting form usability defect attributes and their corresponding instances and values.

Guided wizard defect report form – using this approach ensures defect attributes relevant to the reporter's information needs are presented. From a technical user's standpoint, plain forms might be sufficient to report software defects, as they know what to write and just what information is necessary to report. This is not the case for less technical users. For an optimum usability defect reporting process, we considered a guided wizard form solution to guide novice reporters and less technical users through the reporting process, to hint at what is expected from them at each step, and to present relevant options. Since we introduced more usability attribute instances to collect important information, a plain form is not the best way to collect this kind of data. Key design decisions included: we break up the attributes into smaller sections presented one at a time;

early data entered influences later data and we used hide/show logic at the attribute and page level; we set predefined values for many attributes to keep the content clear and concise; we provide specific hints about the information the reporter must enter; we use a question-based approach so that users have a better idea on what should be written in the textual form; and due to the limited technical knowledge of many reporters, we used different attributes for understanding the validity of the problem.

Objective assessment of user difficulty – to better measure the impairment of tasks, we capture the users' feelings and difficulty caused by the usability defect. An inspection of existing Bugzilla defect reports revealed their limitation for eliciting information about user difficulty, seeming to miss a coherent expression of users' feelings and struggled to accomplish certain tasks. Moreover, the subjective nature of usability defects made some people think that the issue they identified was invalid. Reeder and Maxion [30] defined the user difficulty effect as "when the ability to achieve a goal is impaired". We used two user difficulty dimensions from [30], and the subjectivity in determining user difficulty was measured by using a scale rating with predefined values.

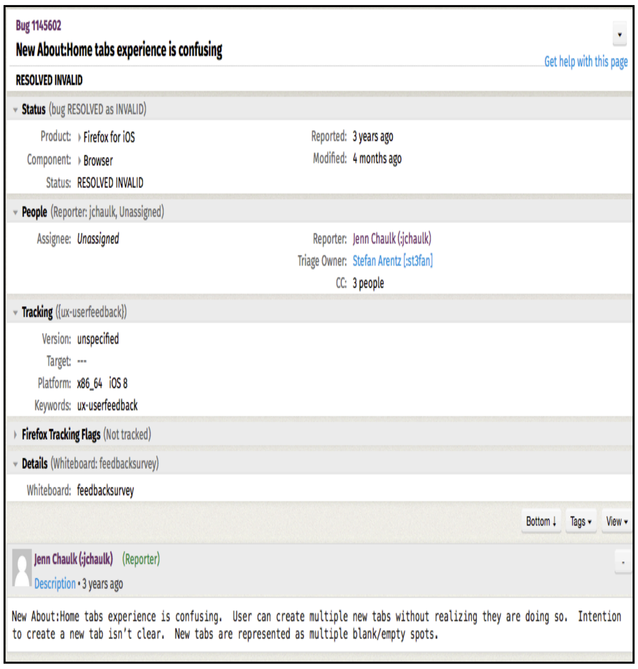
B. *Usage Example*

We briefly demonstrate our guided usability defect report form prototype. We reproduced the Firefox for iOS#1145602 issue shown in Figure 1 on our iPhone and wrote the detailed defect descriptions based on our reproduction steps. Refer section IV for information on the source of the defect.

To minimize the cognitive load imposed by our proposed attributes and instances, we presented the attributes in a tab-form style to ease navigation, group related content and avoid long-scroll page. In total, there are five tabs that represent the main attributes of the existing Bugzilla defect report form. Table 2 shows brief definitions of the proposed usability defect attributes and lists their values. *Pop up callouts* were also used for each free-form text attribute to provide hints on what type of information should be supplied within each text input field. The callouts pop up when a cursor is moved to each given text box. The remainder of this section provides an overview of the attrributes and instances.

1) ***Reporter Identification Tab****:* Similar to the Bugzilla defect report form, upon submission of a new defect report, a reporter has to select their role. This depends on whether the defect reporter finds the defects while using an OSS product, or when they contribute to OSS development. The selection of this role will determine the relevant defect attributes that will be prompted in the floowing screens. Figure 2 illustrates a role selection.

2) ***Software Information Tab****:* Information about the open source software being used will vary between software developer and user. When a software developer reports a usability defect, the information about product, version, and component is compulsory. However, component information is made optional for end users since component information is not automatically available to users. For less technical users who are not involved in product development, selecting appropriate component is sometimes difficult as they might not know which components best fits the location of the issues. Refer to Figure 3.


Bug 1145602
New About:Home tabs experience is confusing
Get help with this page
RESOLVED INVALID

Status (bug RESOLVED as INVALID)
Product: Firefox for iOS
Component: Browser
Status: RESOLVED INVALID
Reported: 3 years ago
Modified: 4 months ago

People (Reporter: jchaulk, Unassigned)
Assignee: *Unassigned*
Reporter: Jenn Chaulk (:jchaulk)
Triage Owner: Stefan Arentz [:st3fan]
CC: 3 people

Tracking ({ux-userfeedback})
Version: unspecified
Target: ---
Platform: x86_64 iOS 8
Keywords: ux-userfeedback

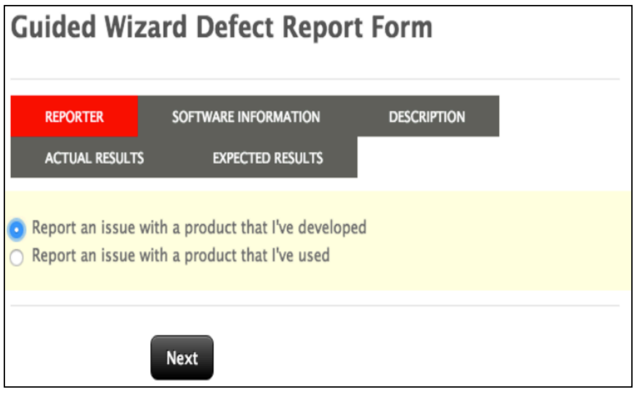
Firefox Tracking Flags (Not tracked)

Details (Whiteboard: feedbacksurvey)
Whiteboard: feedbacksurvey

Jenn Chaulk (:jchaulk) (Reporter)
Description • 3 years ago

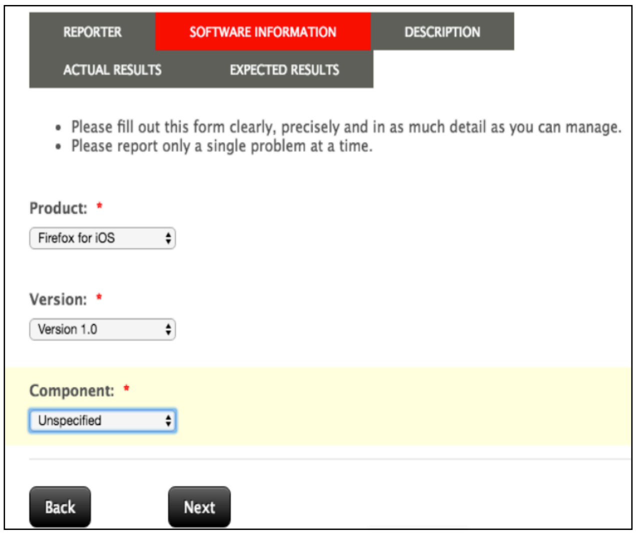
```
New About:Home tabs experience is confusing.  User can create multiple new tabs without realizing they are doing so.  Intention
to create a new tab isn't clear.  New tabs are represented as multiple blank/empty spots.
```

Fig. 1. Example of usability defect in Firefox gor iOS project


Guided Wizard Defect Report Form

[✓ Report an issue with a product that I've developed]
Report an issue with a product that I've used

Fig. 2. Type of reporter identification tab



- Please fill out this form clearly, precisely and in as much detail as you can manage.
- Please report only a single problem at a time.

Product: * Firefox for iOS

Version: * Version 1.0

Component: * Unspecified

Fig. 3. Software information tab

TABLE II. LIST OF ATTRIBUTES AND VALUES USED IN NEW USABILITY DEFECT FORMS

Attribute	Attribute instances	Attribute types	Input types	Values	Changes made to Bugzilla defect report form for current study
Software Information *	Product	Compulsory	Predefined data	The name of the application in which the issue exists.	Reused attribute
	Component	Optional	Predefined data	The sub-part of the application on which the issue exists.	Reused attribute
	Version	Compulsory	Predefined data	The version of application in which user discovered the usability issues. If user can reproduce the issue in more than one version, select the earliest.	Reused attribute
Description	Summary / title	Compulsory	Free form text	Headline summarizing the usability issues. The title should consist the description about (1) a software entity or an entity behavior, (2) a relevant quality attribute, (3) the problem, (4) the execution context, and (5) whether the report is a defect or feature request.	Reused attribute but a caption is provided to hint the appropriate ways to write meaningful defect report title.
	Usability defect type	Optional	Predefined data	Indicates the categories of usability issues. There are six primary categories: (1) Visualness, (2) Information presentation, (3) Audibleness, (4) Manipulation, (5) Task execution, and (6) Functionality. The selection of the main usability defect category will determine the associated usability defect subcategories that will be prompted. Upon selection of usability defect subcategories, examples of defects associated with the selected category are listed. If the experienced problem is not in the list, the defect reporter may choose "Other" option and describe the problem in detail.	New attribute. The values were based on the revised Open Source Usability Defect Taxonomy [31]. The values were organized hierarchically into six categories: *Visualness* (appearance, layout, object state), *Information presentation* (naming and labeling, non message feedback, error and notification message, on screen text, menu structure), *Audibleness* (voice and sound, audio cues, text and feedback in speech), *Manipulation* (keyboard press, mouse click, finger touch, voice control, scrolling mechanism, drag and drop, zooming), *Task execution* (action, reversibility, system task feedback), *Functionality*.
	Problem summary	Compulsory	Free form text	Concise description of the usability issues.	Reused attribute.
	Steps to reproduce	Compulsory	Free form text	Step by step instructions to reproduce the issues. It is recommended that the instructions to be explicitly given as a numbered sequence of instructions.	Reused attribute.
	Assumed cause	Optional	Free form text	Description of what possible cause of the problem.	New attribute.
	Failure qualifier		Predefined data	Indicates why users think the experiencing usability issues or dissatisfaction of the product is a real issue: (1) Wrong, (2) Missing, (3) Incongruent mental model, (4) Irrelevant, (5) Better way, and (6) Overlooked. Only one value can be selected at one time.	New attribute. This attribute was adapted from [33]. The definition of the values was refined to reflect the interpretation of failure qualifier based on the statement written in the defect report, not from the observation.
Actual results	User observation	Optional	Free form text	Description of what was wrong, why is it wrong, or, any error shown.	Reused attribute.
	Task difficulty	Optional	Free form text	The anticipated difficulties/ challenges the user encountered as a consequence of the problem. Additional information about how you did a workaround for the usability defect to continue using the software could be explained here.	New attribute.
	Annoyance level	Optional	Predefined data	A scale to rate the user's annoyance or mood when experiencing the usability issues (1-5 scale; 1= Not at all, 5 = Very much).	New attribute.
	Reproducibility	Optional	Predefined data	The ability to reproduce the issue and make it happen again in the same device, different device, or in other environment: Yes or No.	Revised attribute. The free text attribute was converted to Yes and No option.
	Support evidence	Optional	Attachment	Supplementary material such as image, audio, video, stack traces, crash log that can help to reproduce the issue.	Reused attribute.
Expected results	User expectation	Optional	Free form text	Describes what behavior is expected when the issue occurred, or what change is required to the way the software works or to improve some other aspects.	Reused attribute.
	Solution proposal	Optional	Free form text	Recommendation to remedy the usability issues, including the alternatives solutions and rationale for the recommendation.	New attribute
	Support evidence	Optional	Attachment	Supplementary material to support the idea of proposal such as Photoshop sketches, ASCII art, screenshots, or code patch.	Reused attribute

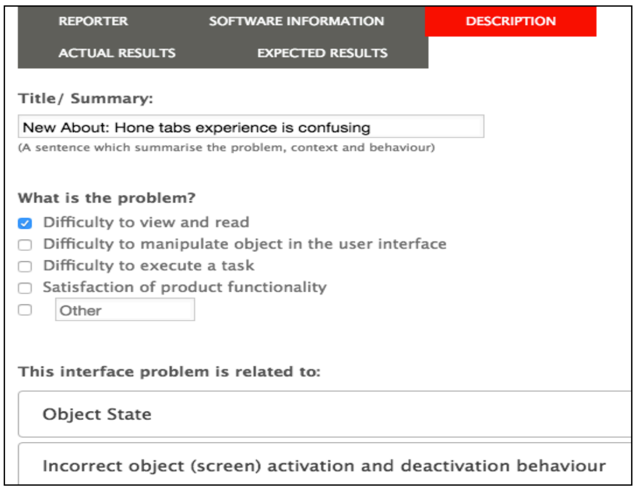
* Software information attribute instances can be more depending on the project. Other instances are such operating system, hardware platform, build identifier and etc.

*3) **Description Tab**:* Rather than an unstructured text form as in the default Bugzilla setup, in our wizard, usability defect inforamtion is captured in multiple attribute instances as shown in Figure 4. Five attribute instances are used to summarise usability issues – title, usability defect type, problem summary, steps to reproduce, and failure qualifier. The usability defect type and failure qualifier attributes were added to the original Bugzilla defect report to help software developers understand the nature of the problem and accept the issue as valid. Both attributes are often found in usability evaluations, however, in the OSS development, the absence of formal usability evaluation make it impossible for software developers to understand how and why users claim certain difficulties as usability issue. The values of usability defect type and failure qualifier were adopted from our revised open source usability defect taxonomy (OSUDC) [31] and Orthogonal Defect Classification [32] scheme. However, we revised the values of failure qualifier to reflect the OSS situation.

- Wrong – when the reporter notices that something has gone wrong while performing a task or some elements on the user interface are violating usability principles and standards.
- Missing – when the reporter fails to find something in the user interface that he/she expected to be presented, or the results of performing certain task did not meet his/ her expectations.
- Irrelevant – when the user interface contains information objects, steps to accomplish task or functionality that do not contribute to system services and are unnecessary.
- Better way – when the reporter suggests that something in the user interface could have been done differently, or suggests a different way of doing a certain task.
- Overlooked – when the reporter overlooks an entity in the user interface, or does not know how to perform a certain task.
- Incongruent mental model – when the user interface is unclear because it does not match the reporter's mental model, previous experiences, or they notice inconsistencies with other similar applications.

*4) **Actual Results Tab**:* Describes what currently happens when the usability defect is present, as shown in Figure 6. There are five attribute instances – user observations, task difficulty, annoyance level, reproducibility and support evidence. The task difficulty and annoyance level are two new attributes to emphasize the impact the issue has on users. We used plain input text to give flexibility for reporters to explain their challenges while overcome the issue, and rating scale to objectively measure user frustration.

*5) **Expected Results Tab**:* Describes what the user believes should happen if the defect were fixed, as shown in Figure 6. There are three attribute instances – user expectation, solution proposal and supporting evidence. Solution proposal is an optional attribute to allow reporters to share ideas how each issue should be solved, how it actually works from a user's perspective, or a proposed technical solution. If reporters have any sketches or ASCII art, they could upload them.



Title/ Summary:

New About: Hone tabs experience is confusing

(A sentence which summarise the problem, context and behaviour)

What is the problem?

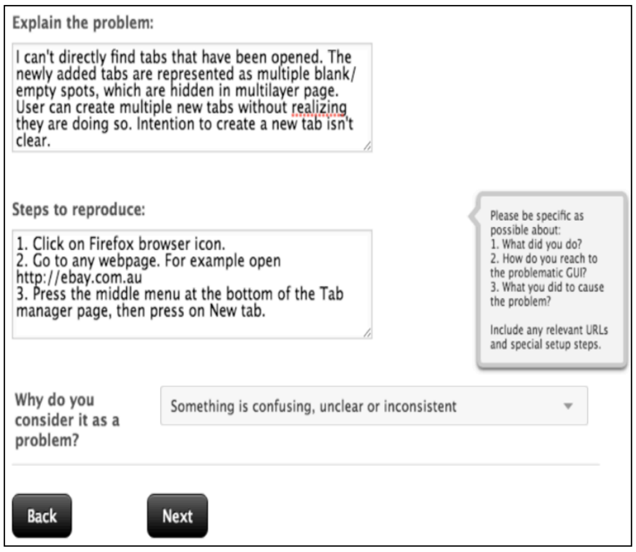
- [✓] Difficulty to view and read
- [ ] Difficulty to manipulate object in the user interface
- [ ] Difficulty to execute a task
- [ ] Satisfaction of product functionality
- [ ] Other

This interface problem is related to:

Object State

Incorrect object (screen) activation and deactivation behaviour

(a) Description about usability defect type



Explain the problem:

I can't directly find tabs that have been opened. The newly added tabs are represented as multiple blank/ empty spots, which are hidden in multilayer page. User can create multiple new tabs without realizing they are doing so. Intention to create a new tab isn't clear.

Steps to reproduce:

1. Click on Firefox browser icon.
2. Go to any webpage. For example open http://ebay.com.au
3. Press the middle menu at the bottom of the Tab manager page, then press on New tab.

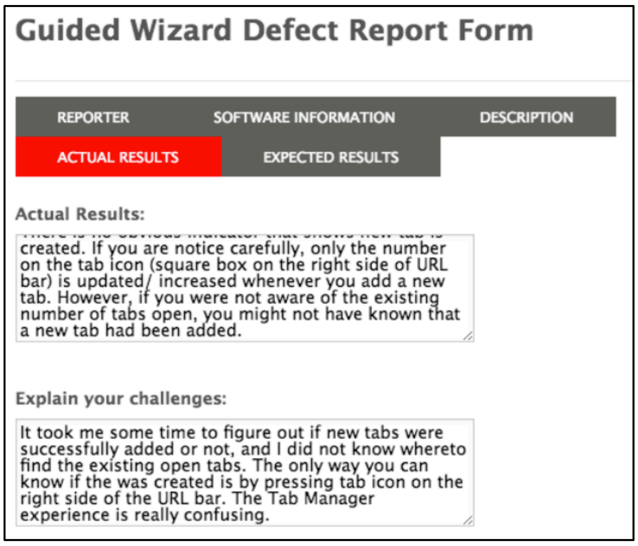
Please be specific as possible about:
1. What did you do?
2. How do you reach to the problematic GUI?
3. What you did to cause the problem?

Include any relevant URLs and special setup steps.

Why do you consider it as a problem? Something is confusing, unclear or inconsistent

(b) Detailed description about the problem, steps to reproduce and failure qualifier

Fig. 4. Description information tab



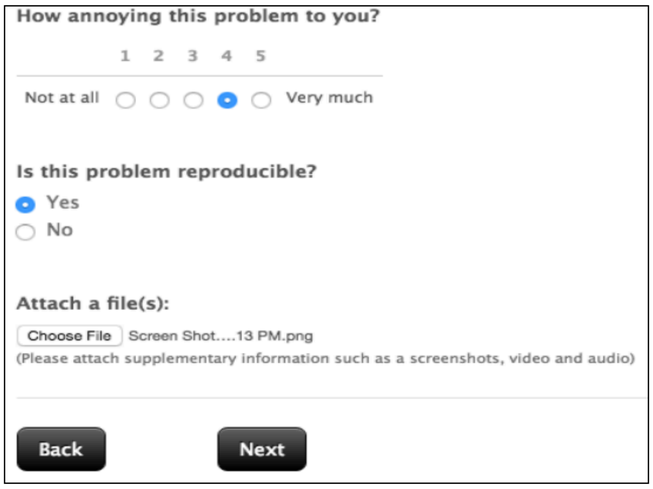
Actual Results:

created. If you are notice carefully, only the number on the tab icon (square box on the right side of URL bar) is updated/ increased whenever you add a new tab. However, if you were not aware of the existing number of tabs open, you might not have known that a new tab had been added.

Explain your challenges:

It took me some time to figure out if new tabs were successfully added or not, and I did not know whereto find the existing open tabs. The only way you can know if the was created is by pressing tab icon on the right side of the URL bar. The Tab Manager experience is really confusing.

(a) Description about the actual results


How annoying this problem to you?

	1	2	3	4	5	
Not at all	○	○	○	●	○	Very much

Is this problem reproducible?

[✓ Yes] No

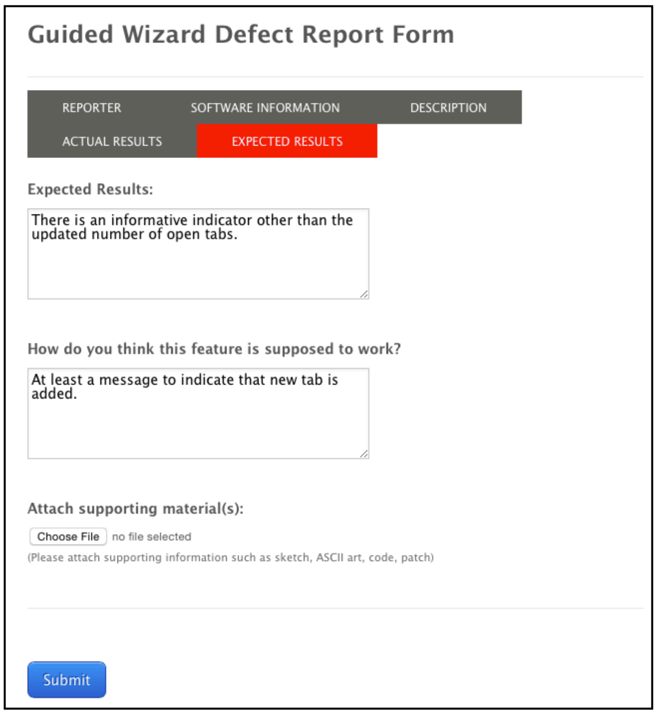
Attach a file(s):

Choose File Screen Shot....13 PM.png

(Please attach supplementary information such as a screenshots, video and audio)

(b) Information about annoyance level, reproducibility and supplementary materials

Fig. 5. Actual results tab


Guided Wizard Defect Report Form

Expected Results:

There is an informative indicator other than the updated number of open tabs.

How do you think this feature is supposed to work?

At least a message to indicate that new tab is added.

Attach supporting material(s):

Choose File no file selected

(Please attach supporting information such as sketch, ASCII art, code, patch)

Fig. 6. Expected results tab

IV. PRELIMINARY EVALUATION

Even though the proposed usability defect report form was intended for less technical users, in this preliminary evaluation we only focused on expert users. The rationale of assessing via expert judgment during our preliminary evaluation was to verify the clarity of technical content in particular, to assess the readability and understandability of the proposed attributes, whereas these aspects may have been more difficult and less reliable to verify with less technical users. The outcome of the preliminary evaluation only focuses on the aspect of *clarity* of the usability defect report information, rather than ease of use of using the proposed forms or the effectiveness of the proposed attributes to improve defect resolution time. The following subsections describe participant selection, problem selection, and protocol in conducting our preliminary evaluation.

A. Participant Selection

In our study, three experienced software development experts evaluated the information presented in the Bugzilla defect report form and our proposed form. The evaluators had significant levels of experience in both industrial development and academic research environments. Two evaluators had substantial experience with Bugzilla, and one had limited exposure to the Bugzilla defect reporting tool. The participation is on voluntary basis.

B. Problems Selection

To evaluate our proposed open source usability defect report forms, we decided to use the Firefox for iOS project. Firefox for iOS is a mobile web browser from Mozilla for the iPhone, iPad, and iPod touch. We selected ten usability defects from the Firefox for iOS project for the evaluation case study. These case study defect reports were then used in our evaluation. The ten usability defects were chosen in the following way:

- The usability problems were selected randomly from the 861 *New* defects (as of 21st March 2017). The decision to use defects with *New* status guaranteed that the defects had not been examined by the software developers, and we have the possibility of reproducing the defects using our own Firefox for iOS app and reporting them using our proposed form.
- The defects were tagged with Bugzilla usability keywords - 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-userfeedback, ux-visual-hierarchy. The rationale for using these developer-tagged keywords was made to reduce selection bias, as the software developers already assessed the validity of the defects and accepted the need for fixing them.
- The defects are reproducible using our own iOS mobile device. This allowed us to rewrite the usability defect descriptions using our defect report form and not bias them based on the original descriptions that had been submitted.

C. Development of Case Studies

We chose five usability defect reports from Firefox for iOS projects. We considered two approaches to select the report to reproduce: sampling randomly, or sampling only reports with GUI-related usability defects. We chose the latter, since our goal is to reproduce the issue and rewrite the usability defect descriptions. We found GUI-related usability defects are more objective and much easier to reproduce in our iPhone. We read the defect report, understanding the problem context, and reproduced the problem on our own until we found the reported problem. Then we used our proposed usability defect report forms to write usability defect descriptions for the defects. The defect report evaluation case studies used can be found in Chapter 7 of [31].

Although both original and proposed defect report forms contain specific contextual information (i.e., status, people, tracking, software information), the defect descriptions given to evaluators contained minimal information. We only provided contextual information about reproduction steps, actual and expected results, and explanation of the usability defects. We limited the amount of detail provided to evaluators to ensure the evaluators were not biased with a specific defect report format. The final copy of defect reports that were presented to the evaluators was modified to prevent the identification of specific formats.

D. Evaluation Criteria

The four aspects we used for the evaluation were adopted from [33], [34]:

- ***Informative*** – According to Capra's guidelines [35], informative usability defect descriptions should describe the solution to the problem, the cause of the problem, and the usability issue involved in the problem. Describing this information has been suggested as important to better understand and fix the problem [35], [36]. This information should be supported with screen snapshots, pictures, video and audio, usability design principles and/ or previous research.
- ***Accuracy*** – Accuracy is measured in terms of how closely the problem can be reproduced by the evaluators. Good defect descriptions should consist of a clear set of instructions that other readers can use to reproduce the defect on their own;
- ***Claim and rationale*** **-** In the absence of usability specialists to observe and verify usability defects in open source projects, justification about why it was a problem [33] is critical for software developers to understand the nature of the problem. The claim about the problem should justify the failure qualifier criteria that violates user expectations, including missing, incongruent mental model, irrelevant, wrong, better way, and overlooked. When arguing for a particular claim, support for rationale and evidence is valuable in confirming the validity of the problems.
- ***Impact*** – The defect description should contain something valuable that highlights the priority of defects that need to be fixed. For this purpose, defect reports should describe the impact of the problem on business goals (i.e., costs, time loss), user task, and human emotion [4]. Impact on the user's task explains about interruptions of task performance, unnecessary steps to work around the problems, or the user struggling with task completion, while human emotion places emphasis on confusion, frustration, annoyance, and uncertainty [30].

For each of these aspects, the evaluators were given eleven questions as listed in Table 3. Each evaluation aspect was given a score of 1 if the evaluators thought that the usability defect report "completely described", 0.5 if the usability defect report "partially described", and 0 if the usability defect report "do not describe" the evaluation aspects. The total quality score was calculated by summing up the scores, that ranged between 0 (low quality) and 11 (high quality).

E. Protocol

Each evaluator was given the following material: (1) Five original usability defect reports of Firefox for iOS product; and (2) Five usability defect reports of Firefox for iOS developed with our new reporting tool. The evaluators were required to read the ten defect reports and evaluate the contextual information of each report on four aspects. For each aspect, we evaluated whether the report provided or failed to provide a description containing the aspect under consideration. Problems were evaluated in random order and it was not made known as to which reporting tools and format was used to record the usability defects the evaluators reviewed.

F. Analysis

Due to the limited number of evaluators, we used descriptive statistics to describe evaluation results. In addition, we measured level of agreement among evaluators using Fleiss Kappa Inter-rater reliability (IRR)[1].

V. Results and Discussion

The expected outcome of this preliminary evaluation was that case study usability defect reports contain much more information than the original usability defect reports, receiving higher scores for the eleven evaluation criteria. Table 4 shows the evaluation scores for the eleven evaluation criteria range from 0 to 1. The outer right column is the overall inter-rater reliability for each usability defect report. As shown in Table 4 it can be seen that the case studies usability defect descriptions were evaluated with higher scores than the original usability defect descriptions for most of the evaluation criteria. Also, the higher agreements on case studies usability defect descriptions suggest that the presence of certain information in the report is clearer and more understandable to the readers.

TABLE III. Evaluaton Criteria Asked in the Expert Evaluation

Aspect	Questions
Informative	Q1. Does the defect report offer proposals for solutions? For example, the descriptions provide alternatives and tradeoffs, and supplied rationale for the recommendations [33]. Q2. Does the defect report describe the cause of the problem, including a justification of what posed a problem, including system components that are affected or involved? Q3. Does the defect report describe the main usability issue involved in the problem? For example, a description about what is wrong with the interaction architecture, interface and user task design.
Accuracy	Q4. Has the defect report explained in detail step by step how to reproduce the problem, including user's navigation flow through the system? Q5. Are you able to reproduce the problem on your own device and environment? Q6. Were the actual results you observed similar to the one in the defect description?
Claim and rationale	Q7. Does the defect report offer a justification for why the reporter thinks that it was a problem?
Impact	Q8. Does the defect report explicitly mention what poses a problem to the user? Q9. Does the defect report describe the impact of the problem on business effect, impact on the user's task, and importance of the task? Q10. Does the defect report describe reporters' emotion, feeling, or reactions with regards to the issues? Q11. Does the defect report mention how often the problem occurred or if other users experienced the same problem?

[1] http://www.real-statistics.com/reliability/fleiss-kappa/

TABLE I. PERCENT AGREEMENT ACROSS THREE EVALUATORS EVALUATED THE PRESENCE OF USABILITY DEFECT INFORMATION

Report		Evaluators	Evaluation Criteria											IRR
			Informative			***Accuracy***			***Claim and rationale***	***Impact***				
			Q1	***Q2***	***Q3***	***Q4***	***Q5***	***Q6***	***Q7***	***Q8***	***Q9***	***Q10***	***Q11***	
Original	**1**	E1	0.5	1	1	0.5	0	0	0	0.5	0	0.5	0	0.79
		E2	0.5	1	1	0.5	1	0.5	0	1	0.5	0.5	0	
		E3	0.5	1	1	1	1	0.5	1	1	0	1	0	
		% Agreement	1.00	1.00	1.00	0.67	0.67	0.67	0.67	0.67	0.67	0.67	1.00	
	4	E1	0.5	0	1	0.5	0	0	0	0.5	0	0	0.5	0.55
		E2	0.5	0.5	1	0.5	0	0	0.5	1	0.5	0	0	
		E3	1	1	1	1	1	1	1	1	0	0	1	
		% Agreement	0.67	0.00	1.00	0.67	0.67	0.67	0.00	0.67	0.67	1.00	0.00	
	6	E1	0	0	0	1	0	0	0	0	0	0	0	0.48
		E2	0.5	0.5	0.5	1	0	0.5	0.5	0.5	0.5	0.5	0	
		E3	1	1	1	1	0	0	1	0.5	0	0	1	
		% Agreement	0.00	0.00	0.00	1.00	1.00	0.67	0.00	0.67	0.67	0.67	0.67	
	8	E1	0.5	0.5	0.5	0.5	0.5	1	0	0	0	0	0	0.61
		E2	0.5	0.5	0.5	1	0.5	0.5	0.5	0.5	0	0	0	
		E3	0.5	1	1	1	1	1	1	1	0	0.5	1	
		% Agreement	1.00	0.67	0.67	0.67	0.67	0.67	0.00	0.00	1.00	0.67	0.67	
	10	E1	0	0	0.5	0.5	1	1	0.5	0.5	0	0	0	0.55
		E2	1	0.5	1	0.5	0.5	0.5	0.5	1	0.5	0	0	
		E3	1	1	1	1	1	1	1	1	1	1	1	
		% Agreement	0.67	0.00	0.67	0.67	0.67	0.67	0.67	0.67	0.00	0.67	0.67	
Case Study	**2**	E1	0.5	1	**1**	**1**	1	1	1	**1**	0.5	1	0	**0.88**
		E2	0.5	0.5	**1**	**1**	1	1	1	**1**	1	1	0	
		E3	1	1	**1**	**1**	1	1	1	**1**	0.5	1	0.5	
		% Agreement	0.67	0.67	**1.00**	**1.00**	1.00	1.00	1.00	**1.00**	0.67	1.00	0.67	
	3	E1	1	0.5	**1**	**0.5**	1	1	0.5	**1**	0	0.5	0	0.73
		E2	1	0.5	**1**	**1**	1	1	1	**1**	1	1	0	
		E3	1	1	**1**	**0.5**	0	0	0	**1**	0	1	0	
		% Agreement	1.00	0.67	**1.00**	**0.67**	0.67	0.67	0.00	**1.00**	0.67	0.67	1.00	
	5	E1	1	1	**1**	**1**	1	1	1	**1**	0	0.5	1	0.73
		E2	1	0.5	**1**	**1**	1	1	0.5	**1**	0.5	1	0.5	
		E3	1	1	**1**	**1**	0	0	1	**1**	0	0	1	
		% Agreement	1.00	0.67	**1.00**	**1.00**	0.67	0.67	0.67	**1.00**	0.67	0.00	0.67	
	7	E1	0	1	**1**	**1**	0	0	0.5	**1**	0	1	1	0.70
		E2	1	0.5	**1**	**1**	1	1	1	**1**	0.5	0	0.5	
		E3	1	1	**1**	**1**	1	1	1	**1**	1	1	1	
		% Agreement	0.67	0.67	**1.00**	**1.00**	0.67	0.67	0.67	**1.00**	0.00	0.67	0.67	
	9	E1	1	0.5	**1**	**1**	1	1	1	**1**	1	1	1	**0.88**
		E2	1	0.5	**1**	**1**	1	1	1	**1**	1	1	0.5	
		E3	1	1	**0**	**1**	1	1	1	**1**	0.5	1	1	
		% Agreement	1.00	0.67	**0.67**	**1.00**	1.00	1.00	1.00	**1.00**	0.67	1.00	0.67	

The highest score (score =1) of evaluation aspects for case study was observed in *Q3 – issue description*, *Q4 – steps to reproduce*, and *Q8 – user difficulty*. The three evaluators were in a perfect agreement (IRR = 100%) on four out of five defect reports, indicating that these reports more clearly described the issue and steps to reproduce. In the original defect reports, even though the three evaluators could identify the presence of this information, their evaluation of the clarity of this information varies. This result suggests that by introducing multiple attributes instance, giving hints and example for free form text input could improve the clarity of the usability defect descriptions.

For the *user difficulty* attribute, even though this attribute was not specifically requested on the original defect report, this information could be identified by some of the evaluators with varying scores. Our approach to introduce specific attributes to collect information on task difficulties has shown a promising assistance to help reporters convey the impact of the issues in a meaningful way. From our preliminary evaluation, the three evaluators had 100% agreement that the five case study defect reports have well described user difficulties and challenges. However, other evaluation criteria related to impact, *Q10 – human emotions,* has shown contrary results, even though the new proposed usability defect form has specifically requested this information. The varying scores of Q10 for all the five case studies usability defect descriptions reflect that user frustration written in a textual description may cause different interpretations. Although Klein et al. [37] found that users are able to express their frustration positively, but not all users are skilled enough to communicate details of the frustrating issues to the developers [38]. In fact, commonly used web-based defect reporting tools, such as Bugzilla, do not provide an easy means to convey user frustration. This motivated us to use "annoyance level rating" scale in our proposed usability defect report form. However, the clarity of this attribute was not evaluated in our preliminary evaluation, since we aimed to minimize bias on certain field formats during the evaluation. Rather than presenting user frustration using a rating scale as suggested, we described user frustration as descriptive text.

We also introduced a new attribute *failure qualifier* to capture the reason why reporters think that the experienced usability issues or dissatisfaction of a certain product is a valid issue. Our preliminary evaluation shows that the five case study usability defect descriptions clearly explained their claim and rationale, where three evaluators were in 100% agreement to award a score of 1 to *Q7 – failure qualifier* on the five case studies. Since the original usability defect reports do not have a specific attribute for this information, it is bias to compare the clarity of this information in both original and case study usability defect descriptions. However, the ability of evaluators to identify this information in the original usability defect description suggests that some reporters were able to provide this information even though the description was vague. The introduction of predefined values (missing, wrong, incongruent mental model, irrelevant, better way, and overlooked) in our proposed usability defect report form is seen as a promising approach to assist reporters in providing a reason why the problem is reported.

In the current Bugzilla defect report template, the reproducibility of issue was recorded as Yes / No using free text form. However, we were in doubt of this information when mining open source usability defect reports [11] - whether the issue has been really reproduced or if it was entered in error due to a default value in this field. As such, we revised the *reproducibility* attribute value. We changed the free text input into predefined option Yes or No. If the Yes option is selected, the number of occurrence will be asked. In this way, we are able to minimize invalid data and provide more evidence to support the impact of the issue. However, our results were not promising. As shown in Table 4, *Q11 – number of occurrence* was rated variedly in both original and case study usability defect reports. We believe this insignificant result was influenced by the way we constructed the case study usability defect reports to avoid bias on a certain format. Further end-user studies will be required to evaluate the effectiveness of the proposed reproducibility attribute.

Our prior work shows that the cause of the problem is rarely found in most usability descriptions [9], and similar findings can be found in Table 4 (refer to *Q2 – assumed cause*). Although information about *assumed cause* could be identified in both original and case study usability defect descriptions, the three evaluators interpret the clarity of this information differently. Possibly, the use of plain text to describe possible cause is not suitable and the descriptive informative may cause subjective interpretations from technical readers. Including evidence-based information such as stack traces, UI event traces, and error logs are beneficial for software developers, but this information is not readily available for less technical users. An approach to resolving this is to rephrase or redefine attributes related to Q2 to better capture relevant information.

In terms of *solution proposal*, the introduction of this attribute did not have a significant impact on described recommendations to solve usability issues. Possibly, the descriptions of *solution proposal* and *expected result* are quite similar and it makes it difficult for evaluators to differentiate between them. The inability of evaluators to fully recognize the presence of solution proposal suggests that the information presented is not meaningful to be the basis of fixing the usability issues (*Q1 – solution proposal*). However, we believe that the introduction of specific attribute to request supplementary material, especially the supplement of visual components like ASCII art, Photoshop sketches is helpful to support the proposing idea.

Since no open-ended question was provided with the assessment questions, we could not obtain the evaluators' qualitative feedback on the overall form. However, the findings from this preliminary evaluation suggest several opportunities for improving the proposed usability defect form, such as: (1) minimize descriptive information to avoid misinterpretation of the problems, (2) find better ways to capture evidence-based information for less technical users, and (3) avoid redundant attributes – in our case, we removed the solution proposal attribute and revised the definition of the expected result attribute to accommodate solution proposal.

VI. Threats to Validity

External Validity - The choice of evaluators and number of evaluators affects the outcome of the evaluation. Previous studies have reported that evaluators' background can have a significant impact on the outcome of software testing [39], [40], and we have seen this potentially affecting the quality of the assessment defect reports as well. We plan to use other evaluation methods in future to retest the outcomes of this study. For example, to replicate the study conducted by Capra [34] with different settings.

Internal Validity - The selection of ten usability defect reports from only one system as an instrument for evaluation is a key threat to internal validity. Since we reused the original defect reports as a basis to construct new ones, this could potentially introduce bias when we self-reproduced the five case studies used in the evaluation. To minimize this bias, instead of just copying the original information and adding dummy information to the new form, we reproduced the defects on our own and wrote the usability defect description using our own interpretation. The original defect reports were used as the guidelines to reproduce the problems. Hence, we consider the instruments as supportive for evaluating the quality of information between original and case study defect reports.

Construct Validity - The appropriateness of the assessment metrics used to rate the quality of information threatens the construct validity. To mitigate this threat, we adopted well-accepted Capra's guidelines [35] as a ground of quality assessment metrics used for evaluation.

VII. Summary and Future Work

Our proposed usability defect report forms contain four main criteria based on the findings of literature reviews, online surveys, and software defect repositories mining: 1) each attribute captures only one value, 2) one attribute is explained with multiple instances, 3) attributes are prompted when relevant to the reporter's information needs, and 4) user difficulties are measured using objective scales. These criteria were used to create a prototype form, based on the current Bugzilla defect report layout. The clarity of usability defects descriptions constructed using our proposed forms was then evaluated using an expert judgment approach. The results from the preliminary evaluation are encouraging. The proposed new attributes show an improvement in the clarity of usability defect descriptions, and the uses of multiple attributes instance could possibly increase the accuracy and completeness of usability defect report content.

These findings have several implications. First, software defect reporting tools could be redesigned to consider information needs of different types of defects. For example, user frustration is very important to convey usability issues and this information can be overlooked if a plain text or generic defect report is used. Second, our results raise several design opportunities to explore about an ideal usability defect description content and format for less technical users, automatically extracted information, the choice of descriptive and evaluative attributes, and the amount of effort required to fill in the form. Future work will consider direct integration of our form into the issue tracker and expanding the scope of evaluation.

Acknowledgment

Support from Swinburne University of Technology, Deakin University, Ministry of Higher Education Malaysia and ARC Discovery Project DP140102185 are gratefully acknowledged.

References

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

[2] N. S. M. Yusop, J.-G. Schneider, J. Grundy, and R. Vasa, "What Influences Usability Defect Reporting ? – A Survey of Software Development Practitioners," in *23rd Asia-Pasific Software Engineering Conference (APSEC)*, 2016.

[3] K. Hornbæk, "Current practice in measuring usability: Challenges to usability studies and research," *Int. J. Hum. Comput. Stud.*, vol. 64, pp. 79–102, 2006.

[4] 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.

[5] D. M. Nichols and M. B. Twidale, "The Usability of Open Source Software : analysis and prospects," in *Open Source Software in Business: Issues and Perspectives*, Ravi Kumar., Hyderabad, India: The ICFAI University Press, 2006, pp. 167–188.

[6] 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.

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

[8] F. P. Simões, "Supporting End User Reporting of HCI Issues in Open Source Software," 2013.

[9] 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.

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

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

[12] 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.

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

[14] 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.

[15] J. D. Strate and P. a. Laplante, "A Literature Review of Research in Software Defect Reporting," *IEEE Trans. Reliab.*, vol. 62, no. 2, pp. 444–454, Jun. 2013.

[16] J. Li, S. Tor, R. Conradi, and K. J. M.W., "Enhancing Defect Tracking Systems to Facilitate Software Quality Improvement," *IEEE Softw.*, vol. 59–66, 2012.

[17] T. Zimmermann and S. Breu, "Improving Bug Tracking Systems," in *31st International Conference on Software Engineering - Companion Volume*, 2009, pp. 247–250.

[18] S. Herbold, J. Grabowski, S. Waack, and U. Bünting, "Improved Bug Reporting and Reproduction through Non-intrusive GUI Usage Monitoring and Automated Replaying," in *IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, 2011, pp. 232–241.

[19] B. Dit and A. Marcus, "Improving the Readability of Defect Reports," in *Proceedings of the International Workshop on Recommendation System for Software Engineering*, 2008, pp. 47–49.

[20] 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.

[21] T. Roehm, N. Gurbanova, B. Bruegge, C. Joubert, and W. Maalej, "Monitoring user interactions for supporting failure reproduction," in *21st International Conference on Program Comprehension (ICPC)*, 2013, pp. 73–82.

[22] K. Moran, M. Linares-Vásquez, C. Bernal-Cárdenas, and D. Poshyvanyk, "Auto-completing Bug Reports for Android Applications," in *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 673–686.

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

[24] N. Bevan, J. Carter, J. Earthy, T. Geis, and S. Harker, "New ISO standards for usability, usability reports and usability measures," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.

[25] J. Howarth, T. S.Andre, and R. Hartson, "A Structured Process for Transforming Usability Data into Usability Information," *J. usability Stud.*, vol. 3, no. 1, pp. 7–23, 2007.

[26] A. Bruun and J. Stage, "Barefoot usability evaluations," *Behav. Inf. Technol.*, vol. 33, no. 11, pp. 1148–1167, Feb. 2014.

[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] M. B. Skov and J. Stage, "Supporting problem identification in usability evaluations," in *Proceedings of OZCHI'05, the CHISIG Annual Conference on Human-Computer Interaction*, 2005, pp. 1–9.

[29] L. Zhao and F. P. Deek, "Improving Open Source Software Usability," in *Proceeedings of the Eleventh Americas Conference on Information Systems*, 2005.

[30] R. W. Reeder and R. A. Maxion, "User interface defect detection by hesitation analysis," in *Proceedings of the International Conference on Dependable Systems and Networks*, 2006, pp. 61–70.

[31] N. S. M. Yusop, "Improving Reporting of Usability Defects in Open Source Software Projects," 2017.

[32] 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.

[33] 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.

[34] M. G. Capra, "Comparing Usability Problem Identification and Description by Practitioners and Students," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 2007.

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

[36] 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.

[37] J. Klein, Y. Moon, and R. W. Picard, "This computer responds to user frustration: Theory, design, and results," *Interact. Comput.*, 2002.

[38] D. M. Nichols, D. Mckay, and M. B. Twidale, "Participatory Usability : supporting proactive users," in *Proceedings of the 4th Annual Conference of the ACM Special Interest Group on Computer-Human Interaction*, 2003, pp. 63–68.

[39] M. Hertzum and N. E. Jacobsen, "The Evaluator Effect: A Chilling Fact About Usability Evaluation Methods," *Int. J. Hum. Comput. Interact.*, vol. 15, pp. 183–204, 2003.

[40] T. Kanij, R. Merkel, and J. Grundy, "A Preliminary Study on Factors Affecting Software Testing Team Performance," in *International Symposium on Empirical Software Engineering and Measurement*, 2011, pp. 359–362.