Reporting Usability Defects: A Systematic Literature Review

Nor Shahida Mohamad Yusop, John Grundy, and Rajesh Vasa, *Member, IEEE*

Abstract—Usability defects can be found either by formal usability evaluation methods or indirectly during system testing or usage. No matter how they are discovered, these defects must be tracked and reported. However, empirical studies indicate that usability defects are often not clearly and fully described. This study aims to identify the state of the art in reporting of usability defects in the software engineering and usability engineering literature. We conducted a systematic literature review of usability defect reporting drawing from both the usability and software engineering literature from January 2000 until March 2016. As a result, a total of 57 studies were identified, in which we classified the studies into three categories: *reporting usability defect information*, *analysing usability defect data* and *key challenges*. Out of these, 20 were software engineering studies and 37 were usability studies. The results of this systematic literature review show that usability defect reporting processes suffer from a number of limitations, including: mixed data, inconsistency of terms and values of usability defect data, and insufficient attributes to classify usability defects. We make a number of recommendations to improve usability defect reporting and management in software engineering.

Index Terms—Systematic review, test management, user interface, usability testing, 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. Usable software should not only have an attractive user interface, but it should be easy to understand, learn, operate and also control [1]. Usability defects are often reported by end users, developers and customers who have limited usability and human-computer interaction (HCI) knowledge. For the purposes of this study we define a usability defect as an *unintended behavior* by the product that is *noticed by the user* and has an *affect on the user experience*. In this study we have also treated a *usability defect report* as a specialized kind of software defect report, as this is how they are currently treated in most software development projects. However, there is some debate in the usability and software engineering communities as to whether these should be treated differently and indeed managed separately. In fact, in the software development industry usability defects have been said to receive less attention than other non-usability defects [2].

Previous studies have reported that many usability defect reports contain unclear descriptions [3], [4], [5], [6]. The lack of features in existing defect tracking systems for capturing usability defect attributes does not help, nor encourage reporters to submit a high quality usability defect report [3]. These in turn lead to reporters sometimes providing irrelevant, incorrect and incomplete evidence.

As tracking and managing usability defects in a separate database may in our view increase the complexity of defect management processes, customizing the defect report form to work best for usability defects would seem to be a more fruitful solution. Thus, one important aspect is to understand the current *state of the art* of research in usability defect reporting. In prior works, usability defects in software development and usability engineering tend to be described using different approaches, therefore, this study considers both the software engineering and usability literature. We aim to identify similarities, commonalities and differences in the way usability defects are reported by these different communities.

In this paper, we report on how we carried out a systematic literature review to identify state of the art in usability defect reporting. We found a total of 57 studies to analyse. To facilitate the review, we mapped the studies into three categories: 1) reporting usability defect information—which is related to research on reporting the defects; 2) analysing usability defect data—which is related to researching the use of defect data; and 3) key challenges—which refer to issues that arise in usability defect reporting and management. Each category is further classified according to research areas and topics.

Our study aims to provide a comprehensive review on the reporting of usability defects to date and the usefulness of different usability defect attributes for defect management activities. We identify some key areas for future research to improve the state of the art in usability defect reporting. Through this study, researchers will find a review of current practices, key open issues and limitations, and important areas for future research with respect to reporting usability defects.

2 Background

In modern software development, usability defects can be reported through traditional defect tracking systems, user

- *N.S.M. Yusop is with Faculty of Computer and Mathematical Science, Universiti Teknologi MARA, Malaysia, and with Faculty of Science, Engineering and Technology, Swinburne University of Technology, PO Box 218, Hawthorn, Victoria 3122, Australia. E-mail: nor_shahida@tmsk.uitm.edu.my.*
- *J.C. Grundy and R. Vasa are with the Faculty of Science, Engineering and Built Environment, Deakin University, AUD 23880, Australia. E-mail: {j.grundy, rajesh.vasa}@deakin.edu.au.*

Manuscript received 29 May 2015; revised 29 Oct. 2016; accepted 27 Nov. 2016. Date of publication 8 Dec. 2016; date of current version 25 Sept. 2017. Recommended for acceptance by H. Sharp.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSE.2016.2638427

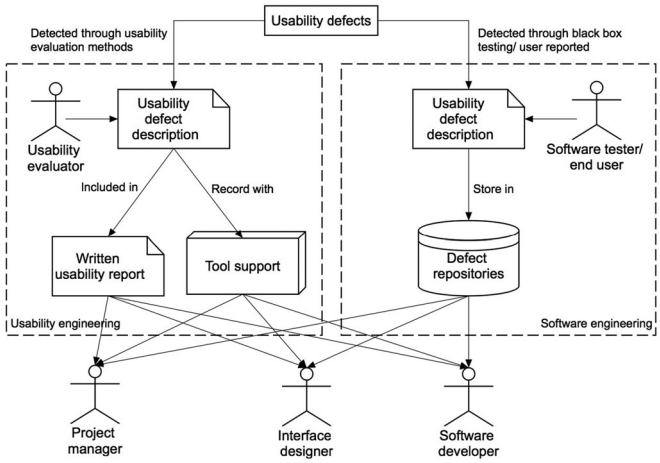

Fig. 1. An overview of usability defect discovery and reporting in the usability and software engineering fields

evaluations, Q&A discussion forums and social media, mailing lists, and other means. Previous research found, however, most of these prominent means of reporting usability defects have limited methods of tracking usability issues [2], [3], [7], [8], [9]. Structured discussions and threaded comments in mailing lists and forums, for instance, are less than ideal for tracking and managing defect information as the content and topics frequently change [10], [11]. Since informational content in defect reports is important for archival value, in this study, we focus on usability defects submitted with or without a predefined format. To this end, we limit our review on usability defect reports originating from defect tracking system and usability evaluation reports, as we depict in Fig. 1.

2.1 Usability Defect Reporting

Usability defect reports originate from two primary sources, as shown in Fig. 1. The first source derives data from usability evaluations. Examples of usability evaluations are usability testing, usability inquiry, heuristic evaluation, analytical modelling and simulation methods [12]. A usability evaluator in the usability-engineering field is referred to as the *expert evaluator* that has formal education, experience and knowledge of usability and/or HCI. During usability evaluation, raw usability data in the form of video and audio, sometimes supported with images and notes, are collected. Based on this raw data, the usability evaluator will then write usability defect descriptions to be included in the final usability evaluation reports, either in the form of written documents or recorded via a digital system. There are various formats to describe usability defects and their uses depending on the usability evaluator's preferences and suitability of the usability evaluation method.

The second source is usability defect reports generated from black box testing performed by software testers, or reported by software end-users. In the software engineering field, the software tester and end-user are referred to as *non-expert evaluators* with often no or limited formal education, training and experience in usability or HCI. They assess the usability of the software product indirectly while verifying the product functionality. In this case, the actual users are not involved because the main purpose is not to test the software usability. If a software tester discovers any frustrating or confusing task, they will report it as a *defect*. This defect report is usually submitted and managed digitally via defect repositories (e.g., Bugzilla, Jira and Trac) or defect tracking systems, so that the information can be shared with other stakeholders, such as a software developer, interface designer and project manager, and the defect correction actions tracked and managed.

Ideally these usability defect reports should provide abundant information for these different roles and responsibilities, from both managerial and technical aspects. From the managerial aspect, usability defect reports serve as source of information for managing team schedules and other resource allocation [13]. They also support tracking effort related to satisfying non-functional requirements on the project. For instance, if the project manager wants to get quick summaries of software developers who worked on usability defect correction, they just need to filter for usability relevant defect reports only. In this case, they may want to see the information in the form of listings, summaries, distribution reports (cross-tab or chart) or trend (time-based) reports [14]. In contrast, interface designers and software developers use usability defect information for the purpose of defect correction. In this respect, they are seeking information that can give them support for the defect correction process, such as event traces, proposed solutions [15] and steps to reproduce the defect.

2.2 Summary of Previous Reviews

We found three published review papers that discussed literature in the area of defect reporting [16], [17], [18]. Two papers cover software defect reporting in general and the other one focus on open source project usability defects. Only one paper used a systematic review approach while the others were based on traditional literature review methods. In contrast to our review in this paper, none of these specifically reviewed usability defect reporting, particularly in software engineering and the HCI fields.

Strate et al. [16] presented a traditional review of software defect reporting. This paper reviewed the commonalities of the research in defect reporting and categorized them into five areas. The goal of their survey was to reveal the state of the art in defect reporting and suggests some open issues for further research. The review contains a general breakdown of automatic defect fixing, automatic defect detection, metrics and prediction, quality of defect reports and triaging defect reports. In contrast, our study focuses on just one of these areas, which is the quality of usability defect reports.

Cavalcanti et al. [18] conducted a systematic mapping study on defect reports in software development. Their paper provides a comprehensive review on challenges in using defect repositories and opportunities to improve software development quality with the use of defect data. In addition, they also investigated the use of well-known tools and online services for defect repositories, such as Bugzilla, Jira, SourceForge, Trac and GitHub to understand how the identified challenges and opportunities have been considered in any of the tools. Similar to Strate et al., they focus on software defects in general. Their review covers only a small amount of the literature that we are interested in that is applicable to software defect repositories.

Despalatović [17] presented a review of the challenges and improvement of usability defects in Free/Libre/Open Source (FLOSS) software. This review was grouped into four general themes of usability aspects—user-centered design, HCI experts' motivation for participation, automatic usability testing tools and suggestion for usability improvement. The review has a different focus and scope from ours. It does not consider how usability defect information is communicated and reported. With its focus on usability defects and inclusion of defect reporting systems, it only covers a small subset of the research in which we are interested.

2.3 Related Literature

Besides the related reviews described above there is other literature related to usability defect and defect reporting that did not include in our systematic review. This was due to these not meeting our specific inclusion and exclusion criteria detailed in Section 3.3. Here we summarize a description of the major research contributions that were excluded from our review. Nevertheless, we use these works together with our systematic review findings in the Discussion section of this paper.

Since the focus of our review is on usability defects, there is literature that we excluded because it had a specific focus on some other specialized areas other than defect reporting. Generally, usability studies focused in these areas are concerned with aspects of usability that are not related to defect reporting.

Usability maintenance—Research in this area looks at post-deployment activities, challenges, opportunities and practices to bridge the gap between software support and maintenance teams [19], [20], [21]. The usability defects found during post–deployment phase, which are usually in the form of user requests and other error reporting, are reported through project forums, mailing lists and trackers.

Automated usability testing—This research area aims to develop automated testing methods for quickly identifying usability defects in various applications, such as web-based applications, mobile applications and haptic systems. Automated testing allows interfaces and user interaction with them to be captured automatically [22], [23], [24]. However, apart from the screen captures and user interaction records, no other usability defect information was reported by these studies.

Usability evaluation methods—Research in this area concentrates on different methods for identifying usability defects, such as think aloud, heuristic evaluation, walkthrough and user observation. There is a significant body of literature devoted to these methods describing tools and methodologies for applying the methods [25], [26], [27], [28], [29], [30], [31], [32], comparing different evaluation methods [33], [34], [35], discussing usability criteria, measurements and metrics [36], [37], [38], [39], and sharing best practices, issues and challenges in industrial [40], [41], [42], [43]. However, we excluded these papers because of their lack of explanation on how the usability defects are actually reported.

Evaluator effect—Research in this area studies the human aspects of performing usability evaluation. Most research we found evaluated the performance of expert and novice usability evaluators in identifying usability defects [44], [45], [46], discussed the effect of single and multiple user problems [34], [47], and investigated the factors involved in the evaluator effect [48]. Unfortunately, these do not address the defect reporting aspect, particularly around how the evaluator effect influences the level of details in a usability defect description and how the usability defects are actually described. Nevertheless, there are a few papers in this area that we included in our review as they do help to answer our research questions [4], [49], [50].

Characterization of usability defects—Research in this area uses data from defect reports to propose usability taxonomies [51], [52] and to classify and prioritize usability defects [53], [54], [55], [56]. This is related to our interest in usability defect reporting approaches but most prior work does not address the actual reporting of defects.

Finally, we excluded a range of research in software engineering fields that leverage defect reporting but do not focus on usability defects. This includes defect-reporting research to understand and improve software development aspects, such as: general assessment and improvement of existing defect tracking systems [14], [57], [58], [59], [60], [10]; use of data from defect repositories for automatic defect fixing, automatic defect detection, metrics and prediction of defect reports, quality of defect reports and triaging defect reports [16]; qualitative study of different defect report types other than usability [61], [62], [63], [64]; and usability practices in open source projects [17], [65], [66]. However, as they fit the broader literature, we use these in our Discussion section in mapping findings to recommendations where appropriate.

3 Research Method

In order to conduct our systematic literature review, we used the guidelines of Kitchenham et al. [67] and Petersen et al. [68]. Our review process consisted of three stages. In the first stage, we defined a set of research questions and prepared a review protocol. This review protocol assisted supervision of the researchers conducting the review and guided the researchers in the data collection. Next, we conducted the searches and selected relevant papers based on an agreed quality assessment. The selected papers were read thoroughly and data as in Tables 4 and 5 was extracted using a data extraction form. Finally, we analysed and synthesized the results for reporting.

3.1 Research Questions

The overarching aim of this systematic literature review was to understand *"To what extent is usability defect reporting considered in existing usability and software engineering research?"*

In usability engineering, usability defects are often found through usability evaluation methods. These usability defects are normally described in the written evaluation reports. On the other hand, usability defects that are found during system testing or reported by end users are reported in defect repositories, such as Bugzilla, Google Chromium and JIRA. These usability defects have the same underlying root cause but were found in a different testing stage and were reported by a different mechanism. This motivated us to review both the usability and software engineering literature to understand how usability defects are communicated in practice. Therefore, the above high-level research question was further divided into the following sub questions.

TABLE 1
Summary of PICOC

Population	Usability defects
Intervention	Defect reporting
Comparison	None
Outcome	Not concentrated on results
Context	Usability engineering and software engineering

These research questions are structured based on PICOC criteria suggested by Kitchenham et al. [69] as in Table 1:

1. How are usability defects communicated in the usability and software engineering literature?
   a. What are the mechanisms used to report and track usability defects?
   b. What are the defect information and formats used for reporting usability defects?
   c. Are there any guidelines available to assist the reporting process?
2. Is there any evidence that usability defects have been studied from the use of data in defect reports?
3. What are the identified challenges of usability defect reporting in usability and software engineering fields?

The first question searched the usability, HCI and software engineering literature to identify research works that focus specifically on usability defect reporting. These were then analysed and classified into topics of studies as suggested by McInerney [70]. The second question identifies studies that analysed data from usability defect report or defect repositories. While the third research question aims to reveal any challenge in reporting usability defects from the perspective of usability and software engineering.

3.2 Search Strategy

3.2.1 Data Sources

Five electronic database resources were primarily used to search usability defect reporting. These include: IEEE Explore, ACM Digital Library, ScienceDirect, Scopus and Google Scholar. These electronic databases selection were based on the recommendations in [12] and [71]. To facilitate the search process, an advanced search option was used that allowed multiple keyword searches. Title and abstract data field were primarily used to retrieve relevant journal and conference proceeding papers. In this research, we only reviewed papers published from the year 2000 onwards. Although we identified a few studies prior to 2000 they were extended in other studies, which were included in our review.

3.2.2 Search Strings

To ensure a thorough search in both usability and software engineering literature, a set of search strings was created for each research question. The search strings were formulated based on:

- Major terms from the research questions.
- Relevant terms extracted from relevant papers, journals and books.
- Synonyms, alternative terms and related concepts of research questions.
- Boolean AND and OR to link all the terms.

TABLE 2
Inclusion and Exclusion Criteria

Inclusion criteria	Exclusion criteria
• Studies that focus on usability defect description/ format/ report/ guideline • Studies that focus on analysing/ using usability defect information • Studies about a tool or mechanism to report usability defects • Empirical studies on usability defects	• Exclude if the paper is on systematic literature review or systematic mapping • Studies not in English • Short papers, posters, introduction to special issues, tutorials and mini-tracks • Defect reporting studies not focussed on usability defects • Usability defects studies not focussed on defect reporting

Three different search strings were derived and executed on different electronic databases. As the literature search progressed, search terms were refined, discarded and added. Any changes to the search strings were rerun on the selected electronic databases to ensure all relevant papers were retrieved. These strings are listed in Table 10 of Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TSE.2016.2638427.

3.3 Study Selection

The primary search resulted in 609 studies. This set was then filtered based on title and abstract analysis, which reduced the total to 191. The significant difference from the first and second filtration was partly due to duplication and relevant to context of study. For instance, the search on the term "usability defect" often returned studies that belonged to medical, engineering or telecommunication topics, which were out of our research context.

We then conducted a secondary search using a *reference chaining* technique. *Reference chaining* is commonly used in other systematic literature reviews [72], [73], [74] as supportive search approach to find relevant studies not found during the primary search. This resulted in 52 new studies being included after the second filtration process.

These total of 243 studies were then analysed by reading the full paper text. At this stage, inclusion and exclusion criteria as in Table 2 were applied to evaluate papers. Since this study surveyed a blend of software engineering, HCI and usability defect reporting literature, a narrow inclusion criterion was used. We defined our inclusion criteria to be specific to each research question [75], while the exclusion criteria are common to all research questions. Reasons to include a paper are: 1) that it belongs to the area of defect reporting in general and usability defect reporting in particular, and 2) that the defect reports originated from defect tracking system and usability evaluation reports, and not from other means of reporting usability defects. As discussed in the introduction and background sections, while a range of means of detecting usability defects exist, we were interested in how they are described, reported and tracked by software development teams, and hence papers that focus on these aspects. This review only considered papers published from January 2000 to March 2016. Finally, 57 studies were included in this

TABLE 3
Quality Assessment Questions

Common questions	
1. Are the aims of the research clearly articulated?	
2. Have other authors cited the study?	
3. Does the study report credible finding with supported data / evidence?	
Usability engineering	Software engineering
4. Is the study's focus on usability defect description?	4. Is the study's focus on defect reporting in software development?
5. Was there an in-depth description of communicating usability defects?	5. Does the paper study usability defects?

review. See Appendix A, available in the online supplemental material, for the list of included studies.

3.3.1 Quality Assessment of Selected Studies

All selected papers were assessed for their quality. Each of these papers was also classified as either a software engineering or a usability study.

Papers were evaluated using two sets of checklists that were formulated to measure the research credibility and validity. Each question was rated as 1 implies "Completely describe", 0.5 implies "Exists but does not completely describe" and 0 implies "Does not exist". The total quality score for each paper was computed by summing up all the scores. This ranged between 0 (very poor) and 5 (very good). The checklist used is shown in Table 3. Based on this quality assessment, we identified all of the selected papers as being of high enough quality to include.

3.4 Data Extraction

We created a data extraction form to extract detailed contents for each study. There are two categories of data extracted for each paper. First, common data such as bibliographic references, type of study, aim, research methodology and data analysis. Second, the specific data that answered each research question. Tables 4 and 5 show the data that was extracted for both categories.

All extracted data was put into shared spreadsheets that were reviewed by the second and third authors. The first author was responsible for reading and extracting the data. In

TABLE 4
Specific Data Items Extracted from All Papers

Search focus	Data item	Description
Quality of defect descriptions	Assessment method	Method used to conduct assessment (i.e. experiment, survey, case study)
	Participant	Participants involved in assessment (i.e. student, usability expert, software developer)
	Assessment criteria	Criteria used to assess the quality of defect description (i.e. clarity, impact, cause)
	Approach	Description of the assessment process
	Outcome	Results of assessment
	Recommendation	Suggestions for future work in related assessment results
Usability defect description content and format	Format	Format used to report usability defects
	Approach	Description of the format usage and characteristics of the format
	Content	List of attributes or information used in the usability description
	Limitation	Limitations of the format
	Benefits	Benefits of the format
	Evidence	The empirical evidence regarding the benefits of using a specified format to improve the quality of defect description
Reporting mechanism	Medium	Medium used to report and track usability defect reports (i.e. form-based reporting and end-user reporting)
	Characteristics	Description of the reporting mechanism
	Benefits	Benefits of the reporting mechanism
	Limitations	Limitations of the reporting mechanism
	Evidence	The empirical evidence regarding the benefits of the reporting mechanism to improve reporting process
Reporting guideline	Aim	The purpose of the guideline
	Guideline	Description of the guideline
	Benefits	Benefits of the guideline
	Limitations	Limitations of the guideline
	Evidence	The empirical evidence regarding the benefits of using the guideline to improve the quality of usability defect reports
	Recommendation	Suggestions for future work
Analyzing defect information	Focus	The purpose of the study
	Attributes used	The data used for analysis purposes
	Approach	Description of the study that related to the research question
	Benefits	The benefits of the study
	Limitations	Limitation of the study
	Evidence	The empirical evidence regarding the benefits of the study
	Recommendation	Suggestions for future work related to analysis results
Issue in defect reporting	Challenges	The problems in defect repositories related to usability defects
	Recommendation	Suggestions for future work related to improving usability defect reporting

TABLE 5
Common Data Items Extracted from All Papers

Data items	Description
Identifier	Unique identification number
Bibliographic	Title, author, year
Type of article	Journal/ conference/ book chapter/ technical report/ theses
Study aim	The aims, goals or objectives of the primary study
Research methodology	Case study, survey, experiment, interview, observation, questionnaire, lesson learned
Data analysis	Qualitative, quantitative or mixed
Study findings	Results and conclusions from the primary studies

order to validate the extraction validity, the second and third authors independently rated a random sample of papers according to the inclusion and exclusion criteria. All discrepancies on the data extracted were discussed among authors with the aim of reaching a consensus. The reliability of the findings of this review was accomplished by considering only the quality score of relevant studies that are greater than 2.5 (50 percent of the percentage score) [74]. We did not measure inter-rater reliability since our review aimed for generalizability of the findings, in particular, to clearly describe how conclusions have been derived from the data instead of comparing agreements of the same codes or themes [73], [76].

4 Classification Scheme

A classification scheme was developed to organize the retrieved studies on usability defect reporting. As shown in Fig. 2, the classification scheme was structured to map onto our research questions. We categorized the studies using the process defined by Petersen et al. We started the classification process by analysing the title, keywords, abstracts and conclusions. We then compiled the keywords and phrases to build a high-level set of categories for classifying the papers. Finally, we grouped the phrases, research objectives and research findings of the papers in each category into a coherent set of themes.

The classification scheme is composed of three main categories; 1) reporting usability defect information—which is related to research on reporting usability defects; 2) analysing usability defect data—which is related to research on the use of defect data; and 3) challenges—which refer to issues identified in usability defect reporting and management.

Table 6 summarizes the distribution of the studies per topic. Studies that addressed more than one topic were classified repeatedly in each topic. For example, Nørgaard et al. [77] investigated mechanisms of usability defect reporting and challenges for each mechanism, and their study is counted in both topics.

In the following section, we present our answers to the review research questions based on analysis of the included studies. Each study is identified as P*m*, where *m* represents the study's number (see Appendix A, available in the online supplemental material, for the list of studies used in this systematic review).

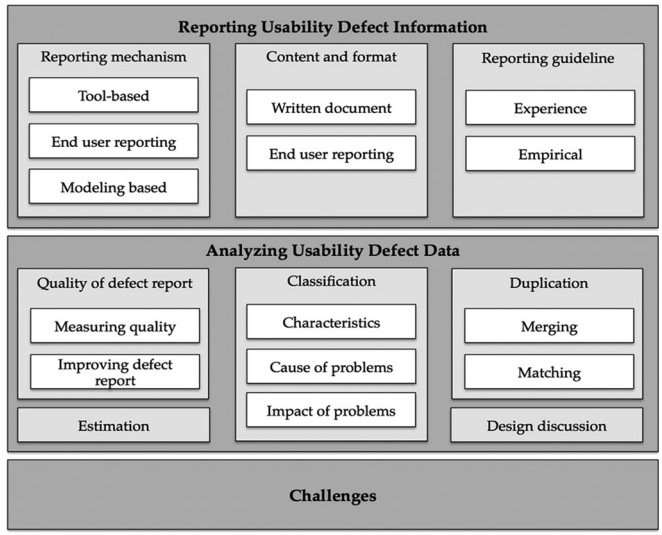

Fig. 2. Classification scheme.

5 Results

5.1 Research Question 1

How are usability defects communicated in the usability and software engineering literature?

a. *What are the mechanisms used to report and track usability defects?*
b. *What are the defect information and formats used for reporting usability defects?*
c. *Is there any guideline available to assist the reporting process?*

Much research in usability evaluation methods aims to improve evaluation techniques. A technique that is able to discover more defects is considered a good one. However, it is not sufficient to identify usability defects without communicating them effectively. In practical usability work, the findings from usability evaluation are useful to compare similar problems, prioritize the fixing task and recommend to developers what to fix. Understanding the characteristics and limitations of certain evaluation formats may impact the comprehensible level of usability defect description. The effectiveness of communicating usability defect information depends on the mechanism to report and track (RQ 1a), content and format (RQ 1b), and guidelines (RQ 1c). A total of 35 papers addressed this question and its sub-questions.

Key findings from our analysis include:

- Three key types of reporting mechanisms were identified in the usability and software engineering literature: *tool-based*, *end-user*, and *modelling-based*. Many tools use free format text to capture most usability defect information. Integration with the system under test and defect tracking systems can be limited.
- A large range (13) of usability defect reporting formats was found—containing a wide variety of data. *Structured web-based forms* are very common but reporting can be impacted by information overload, lack of checking of input, and bias due to form content. Conventional reports offer more unstructured but often richer data. Problem lists and redesign proposals are also common approaches.

TABLE 6
Summaries of Research Areas and Topics

Research areas	Description	Topic	Studies	Total
Reporting mechanism	Investigate how usability defects are collected and reported	• Tool-based reporting	P7, P10, P15, P17, P34, P41, P45, P49, P51, P55	13
		• End-user reporting	P18, P30	
		• Modeling-based support	P28	
Content and format	Investigate what attributes are used to describe usability defects	• Written document format	P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, P18, P24, P25, P29, P30, P32, P33, P35, P36, P37, P43, P47, P53	26
		• Learning-oriented format	P37	
Reporting guideline	Studies that assist reporter in reporting usability defects	• Experience	P2, P52	4
		• Empirical	P5, P37	
Defect report quality	Studies that analyzed defect reports for quality assessment	• Measuring usability defect reports quality	P1, P3, P4, P5, P7, P8, P35, P39, P44	11
		• Improving usability defect reports	P3, P7, P27, P35, P44, P48	
Classification	Studies that analyzed usability defect data to understand the characteristics of usability defects	• Usability defect characteristics	P22, P50	5
		• Cause of usability defects	P42	
		• Impact of software defects	P21, P46	
Duplication	Studies that use usability defect data for identifying similar usability problems	• Matching	P12, P13, P56	4
		• Merging	P36	
Estimation	Studies that use defect data to estimate defect discovery rate		P19	1
Design discussion	Studies that examined the structure and content of defect report discussion	• Addressing and resolving usability defects	P21	3
		• Structure and content of design discussion	P26	
		• Online forum	P31	
Challenges	Studies that discussed the issues of reporting usability defects	• Developer mindset	P1, P9, P37, P53, P54	22
		• Subjective bias	P1, P23, P240, P54	
		• Evaluator effect	P1, P5, P7, P8, P14, P20, P39, P53	
		• Defect discovery methods	P3, P20	
		• Complexity management	P26, P29, P40, P53	
		• Lack of appropriate channel for reporting usability defects	P6, P10, P20, P23, P38, P40, P57	
		• Lack of guideline for specific usability defect information	P29, P36, P54,	

- We found that *problem description*, *severity*, *context* and *redesign description* were the four attributes most commonly used to described usability defects. A wide variety of attributes are used but many infrequently in reports. *Problem description* is widely used but vague in its definition across the studies.
- A small number of studies (4) provide guidelines for usability defect reporting. Two were experienced-based and two empirically-based.

5.1.1 Reporting Mechanisms

Three key types of reporting mechanisms were identified in the usability and software engineering literature. To effectively capture usability defect data, each reporting mechanism uses a variety of input designs, such as auto-generated data, predefined data, free-text form, and online help. Auto-generated data such as *tester name*, *timestamp*, and *problematic user interface* can be automatically recorded when the test is run and the report is submitted. In contrast, predefined data contains a variety of categorical data that is dependent on the input from the reporter. During a defect report submission, the reporter will select some values, such as *severity*, *type of defect*, and *heuristics* used. Some of these values can be changed over the defect life cycle, such as *severity*. For a free-text form, the reporter is allowed to write any description—comments, feedback, complaints, feelings or disappointment, steps to reproduce, expected and actual result—regarding the problems. In summary, the description of three reporting mechanisms can be described as follows:

Tool-based reporting is the easiest way to record and generate data as compared to a paper-based approach [78]. Tool-based reporting allows data to be collected instantly, and recorded data can be measured quantitatively, analysed for trends and used to generate feedback for the quality improvement. A well-designed tool will assist users to provide sufficient data, thus, in turn reduce missing data issues. Several tools in the usability evaluation field were developed to assist in usability defect reporting. Some key examples are outlined below.

Data Collection, Analysis and Reporting Tool (DCART) [P7] uses auto-generated data and free-text form input design. The defect form was designed for collecting and organizing usability defect data in lab-based usability evaluation using a Usability Problem (UP) instance concept. Each occurrence of a UP found by multiple evaluators or multiple times by one evaluator is considered as the same UP. However, given multiple instances of UPs, evaluators must manually review and combine them to determine the main UP experienced by the users.

Merging duplicate problem descriptions [P41] helps evaluators to record usability problems into a database using different usability evaluation methods, to search the database for similar problems, exchange datasets, and to perform a meta-analysis of the datasets.

Web tool [P10] uses tooltips, predefined data and free-form text input and was designed to record usability defects found during heuristic evaluation only. By having the tooltips and examples of usability problems, the evaluators get help on attributes and better guidance to assign severity and heuristics used to find problems. However, a non-integrated reporting tool with the software under test may trouble some users in switching between these two systems and users may bias certain values. In contrast, *Usability Reporting Manager* [P17] uses predefined data and free-text form input. Using a web interface, reporters can enter, manage and export data in into defect tracking system connected to a source code repository.

Usability Problem Inspector (UPI) [P15] uses auto-generated data, predefined data and free-text form input incorporating usability action framework (UAF) content. UPI has two modes; task-based and free-based exploration. Using the task based-approach, evaluators are presented with a series of questions from the UAF structure. When a problem is identified, evaluator is presented with a defect report form and the inspection path is automatically recorded. However, for free-exploration mode, no task information is recorded. *DESTINE* [P34] uses predefined data input. The tool is limited to evaluate ergonomic quality of websites and it can support two types of user profile; expert and designer.

In the software engineering field, defect tracking systems are commonly used to record and track software defects, including usability defects. Our review only found four tools that explicitly assist in usability defect reporting.

GUI monitoring and automated replaying [P45] uses a generic non-intrusive GUI usage monitoring mechanism that can be integrated into existing applications. The monitoring of usage can produce actual usage traces that can be included in the defect reports and used as an input for replaying purposes. The traces are triggered by user interactions like mouse clicks or key presses.

GUI editor tool support [P49] was developed on the Eclipse platform to support exploratory graphical user interface testing. The tool uses Eclipse logging to record uncaught exceptions during execution of a test and a cheat sheet viewer to for evaluators to describe the observed failures. The test results are available in form of results file and can be automatically exported into the defect repositories.

Timeline tool [P51] was developed to visualize monitored interaction traces and application events preceding failures. Using the tool, software developers may analyse the traces to derive steps to reproduce by manually replaying the monitored user interactions.

FUSION [P55] was developed to produce more reproducible defect reports than traditional defect tracking systems. Using the event-driven paradigm of Android application, the tool aids the reporter in constructing the steps needed to reproduce a defect by making auto-completion suggestions based on the potential GUI actions, such as click (tap), long click (touch), type and swipe.

End-user reporting tools collect information in much simpler forms to address users' frustration and complaints and the users report defects as part of their day-to-day activities. We identified two approaches of designing end-user reporting.

One-bit-feedback [P18] uses auto-generated data and a free-text form input. It is a background process that monitors certain system characteristics and packs them into an incident report whenever the user clicks on the screen button or punches the hardware button. The reports are stored locally on the user's system. Usability defect data is collected using auto-generated data and user is given the opportunity to provide comments and feedback through a free-text form. Using this approach, defect incident is automatically recorded and reduces data entry.

Two-mouse-click [P30] uses auto-generated data, predefined data and free-text form input design. The prototype was developed to allow report submission with minimal user click and supplements user comments with objective program state information. The program only collects information relating to the user's interaction. No sensitive information is sent.

Modelling-based reporting provides a standard description with more structured data. The reporter uses a modelling language with defined notation to represent information. For example, ErgoPNets [P28] uses a formalism that combines Petri Nets and ergonomic criteria to describe ergonomic problems and their recommendations. The method used icons, graphical representation and text to describe problem. In this way, the usability defect descriptions can be unified into a model.

5.1.2 Defect Information Content and Format

Thirteen usability defect description formats were identified from the selected studies. Eleven out of the 13 formats are presented in written documents, while the other two are learning-oriented formats. These formats are associated with a list of attributes for communication and report keeping. Altogether, 33 attributes are identified across 13 formats by a total of 26 studies.

As shown in Table 12 (See Appendix B, available in the online supplemental material), we classify these attributes into eight groups based on the defect description content objective, and we summarize all the formats and attributes in Table 7. Note that the attributes checked for each format does not mean that all these attributes are present in the format at any one time. Rather, it is a compilation of several studies that mention the use of certain attributes for a particular format.

The most reported format used to report usability evaluation findings were *web-based form* and *report*. The use of variety input design techniques such as auto-generated

TABLE 7
Summary of Usability Defect Attributes Used in 13 Formats

Attributes	Contents	Total studies	Written document											Learning oriented	
			Problem list	Redesign proposal	Report	Web-based form	Forum	Diary	Multimedia	Human-centered	Screen dump	Digital object	Others	Self-experience	Redesign workshop
Description	ID	6	✓			✓			✓				✓		
	Summary	7	✓		✓	✓									
	Problem Description	23	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		
	Product description	2			✓	✓									
	Actual result	4			✓	✓						✓			
	Expected result	2				✓									
	Type of defect	2				✓						✓			
Impact	Likely difficulties	4		✓	✓	✓									
	Severity	15	✓		✓	✓	✓	✓	✓				✓		✓
	Frequency	2			✓								✓		
	Confidence	1											✓		
	Reproducibility	2			✓	✓									
Location	Context	15	✓		✓	✓				✓		✓	✓		
Discovery resources and methods	Evaluator	2				✓									
	Test user	5			✓	✓						✓			
	Task	7			✓	✓								✓	
	Goal of the task	4			✓	✓						✓			
	Evaluation method	8			✓	✓						✓	✓		
Assumed cause	Possible cause	4		✓	✓	✓									
	Trigger	7				✓							✓		
	Failure qualifier	2	✓										✓		
Solution proposal	Redesign description	12	✓	✓	✓	✓			✓		✓	✓			✓
	Redesign argument	4		✓		✓									
Supplementary information	Attachment	8		✓	✓	✓			✓		✓				✓
	User's response and feelings	1								✓					
	Positive findings	2			✓	✓									
	Business goals	1			✓										
	Recovery steps	1				✓									
	Problem elimination	1			✓										
	Usability specification	1				✓									
	Conclusion	1			✓										
Timestamp	Time on task	3			✓	✓	✓	✓							
	Created date and time	4	✓			✓						✓			
Total studies			5	4	9	9	1	1	2	1	1	1	2	1	1

data [P7, P15, P17, P18, P30], pre-defined data selection [P10, P15, 17, P30], free-form text [P6, P7, P15, P17, P18, P30, P47] and question-based [P6, P47] can produce more structured and consistent defect information. Common attributes collected in the web-based form are *problem description*, *location* of the problem user interface, specific *task* where the problem was observed, what *triggered* the problems, and *severity* rating. There are several key features of the web-based reporting that reduce manual effort. Examples include features for reminding users about key information to report [P7], online help and tooltips for quick reference [P10], support for data transformation into different formats [P17], and automatically recording system-generated information [P15]. However, users are exposed to onerous data entry due to the cognitive load and biased use of default data.

In contrast to web-based forms, conventional *reports* contain unstructured content and a large amount of information. The reports generally provide a detail description of usability evaluation methods that have been conducted so that the content of the report will not only be able to justify the situation of the problems encountered but to also present a good argument to management for requesting resources allocation [P9]. Other attributes commonly reported in conventional reports are *problem description*, *severity rating*, and *attachment*.

The *Problem list*, on the other hand provides lightweight documentation. Even though the content is briefer and lacks context, it is useful to support ongoing discussions [P53], and helps to prioritize tasks during the problems merging process [P37]. In this way, complex problems could be described as multi-faceted without going into a detailed report [P13]. This format usually requires a *problem description* and *severity* rating.

The *redesign proposal* is focused on problem solution. It provides concrete recommendations and arguments. The recommendations are usually supplied with drawings or code fragment [P53]. While software developers may prefer redesign proposals, it is difficult to write useful recommendations for major changes, especially problems that involve business and technical constraints [P29].

The other nine formats are less commonly reported—these include forum and diary [P6], multimedia [P6, P37], human centered story [P37], screen dump [P37], digital objects [P32], self-experience [P37], and redesign workshop [P37]. Even though these formats do not use a variety of attributes, *multimedia* and *redesign workshop* format, for example, provide

TABLE 8
Guidelines for Writing a Usability Defect Report

Studies	Guidelines
Dumas et al. [P2]	Emphasize positive Express your annoyance tactfully Avoid usability jargon Be as specific as you can
Capra [P5]	Describe a solution to the problem Be clear and precise while avoiding wordiness and jargon Describe the cause of the problem Support your findings with data Help the reader sympathize with the user's problem Describe the impact and severity of the problem Describe observed user actions Be professional and scientific in your description Consider politics and diplomacy Describe your methodology and background
Norgaard et al. [P37]	Provide evidences for the observed problems Describes the underlying assumptions that must be agreed upon before the claim can be accepted Provide argument's purpose or position—the difficulty arises from the problems Provide insightful remarks and conclusions about the system being evaluated Use log data or statistical data from a user test as backing for a usability problem Show videos of users struggling with an application or even letting the developers experience the problem themselves
Avnon et al. [P52]	Log one defect for each problem Clearly document each defect Include the visuals—observed problems and the intended design Include violated usability interface guideline Include prototype to visualized complex interactions Prioritize the problems

persuasive and well balanced defect descriptions [P37]. Two studies were categorized as "other" format as they do not clearly state the format used but did mention attributes used to extract usability defect data [P33, P36].

In terms of content, we found that *problem description*, *severity*, *context* and *redesign description* were the four attributes most commonly used to described usability defects across formats. Attributes that are rarely present in usability defect descriptions are *product description*, *expected result*, *type of defect*, *frequency*, *confidence*, *reproducibility*, *evaluator*, *failure qualifier*, *user's response and feelings*, *positive findings*, *business goals*, *recovery steps*, *problem elimination*, *usability specification* and *conclusion*. These attributes are often available in report and web-based forms.

Twenty-three studies primarily used *problem description* to report usability defects across ten formats, except multimedia, self-experience and redesign workshop. However, it is uncertain whether the *problem description* is mutually exclusive—that is the attribute has only one value. This is because *problem description* has a very vague definition in which the reporter has a probability of mixing it with other attributes such as *possible cause*, *type of defect* and *user's response and feelings* [P7]. Sometimes, *problem description* is very brief such as in the problem list format. To support this issue, a *redesign description* can additionally offer insight into the relevance of the usability defect. We found twelve studies that addressed detailed redesign proposals, but only four studies supplied an in depth justification for why the proposed solution is necessary.

Fifteen studies emphasized the *severity* rating in eight formats. However, there is no standard definition used to indicate severity assessment. Some studies have used seriousness [P3, P13], category [P15] and impact [P33] to describe the same meaning. Additionally, there are several severity schemes used for the rating purpose such as 1) minor, serious, critical [P5], 2) major or minor [P8], and 3) severe, moderate and minor [P10].

In terms of software *context*, fifteen studies mentioned the problematic location of elements in the user interface. This information can be either automatically collected [P15, P18, P30] or manually specified by the reporters [P6, P7, P10, P13, P14, P17, P33, P37].

Among the 13 formats, only redesign proposal, report, web-based form, multimedia, screen dump, redesign workshop provided *attachments*. The attachment can be log files [P8], core dumps [P9], screen image [P12, P18], webcam picture [P18] and video clip [P37].

5.1.3 Reporting Guidelines

The review uncovered four guidelines that suggest the way that usability defects should be reported, as shown in Table 8. Two studies provide experienced-based guidelines that were originated from practical lessons and usability experts' point of view [P2, P52]. According to Dumas et al., they way the usability defect is communicated to developers influences the acceptance of the usability defects. Instead of complaining about the negative aspect of the software product, the usability description should also address the positive findings in a clear and precise form.

Another two guidelines were constructed through empirical studies [P5, P37]. Among the four guidelines, the Capra et al. [P5] guideline is the most rigorous and complements Dumas's guideline. Capra's guideline was developed based on a survey of usability practitioners. This guideline is widely used as a criterion to evaluate the quality of usability description [79], [80], [81]. Besides that, the guideline may be used in training usability evaluators and as a checklist when writing a usability defect description. Meanwhile, the Nørgaard et al. [P37] guideline is based on Toulmin's model of argumentation and Aristotle's three modes of persuasion.

5.2 Research Question 2

Is there any evidence that usability defects have been studied from data in defect reports?

There were 23 papers that studied the use of usability defect data for understanding and improving defect management activities. We looked at the commonalities of the research and found five branches of research regarding usability defect reporting:

1. *Quality of usability defect reports*—research in this area investigated the quality of usability defect descriptions and ways to improve reporting.

2. *Classification of usability defects*—research in this area analysed the usability defect data to understand the characteristics of usability defects.
3. *Duplicate defect report analysis*—research in this area concentrated on approaches for identifying and merging similar usability defects.
4. *Estimation for usability defects*—research in this area used data from defect reports to estimate problem discovery rates.
5. *Discussion in usability defect reports*—research in this area examined the structure and content of defect report discussion, how usability resolutions are discussed and how decision—making is made.

In terms of attributes that are commonly used in empirical research (refer Table 11 of Appendix B, available in the online supplemental material), problem *description*, *impact*, and *title/summary* are most widely used. Attributes rarely used by researchers are *type of defect*, *likely difficulty*, *confidence*, *priority*, *software context*, *reporter*, *violated heuristic*, *business goals*, *assignee*, *milestone*, *time to fix*, and *defect fixes*. Research on classification and defect duplication favourably used *title/summary* and *description*, while research on defect report quality often used *observable user actions*, *impact*, *cause of the problem*, and *supplementary information*, *available online*. However, two studies do not report defect attributes used as they employed other metrics such as ISO/IEC 9126 quality model [P21] and IBM quality measurement model [P46].

Key findings from the analysis include:

- Five areas have been studied using usability defect data—report quality, classification of defects, duplicate reports, effort estimation and resolution discussion.
- 22 studies showed that expert evaluator defect reports appear to have better quality than non-experts (the later recruited from students in many studies), though the non-experts are far more numerous. To assess report quality most studies used observable user action, impact, supplementary information, available online, cause of the problem and solution proposal. Two studies used expert judgement assessment. Many studies found that non-experts and even experts struggled to describe many usability issues, especially impact and possible solutions. Six studies focused on improving usability defect reports and provided a range of recommendations, particularly focusing on better support for non-expert reporters.
- Usability defect classification was a focus in five studies, focusing mainly on one of defect characteristics, cause and impact. Some studies used pre-existing usability engineering definitions while some introduced proposed new approaches. All found deficiencies in existing defect reporting tools and usability defect reports in terms of classifying usability defects effectively.
- A small number of studies have focused on duplicate usability defect management, focusing on matching and merging. Most focus on using observable impact of the defect to determine similarity and manual merging processes.

TABLE 9
Categories of Assessment Criteria to Measure Quality of Usability Defect Reports

Category	Assessment criteria	Rank*	Study (s)
Report content	Impact	2	P1, P5, P7, P35
	Supplementary information	3	P1, P5, P7
	Cause of the problem	3	P1, P5, P7
	Observable user actions	2	P1, P3, P5, P7
	Solution proposal	3	P3, P5, P7
	Test procedure description	5	P8
	Executive summary	5	P8
	Business goals	5	P35
	Report layout	5	P8
Software quality model	Clarity	1	P1, P3, P5, P7, P8, P35, P44
	Persistent	5	P3
	Justified	5	P3
	Persuasive	5	P44
	Usefulness	4	P35, P44
	Data quality	5	P8
General	Expert judgment	5	P4, P7

(*) The ranking showed the assessment criteria used the most in measuring quality of usability defect reports (in ascending order).

- Only one study used usability defect reports to estimate defect discovery rates, and none were found using usability reports to estimate likely defect correction effort.
- Three studies were found investigating how defect resolution is carried out using defect reports. The focus has been on discussing validity of reported usability defects and critiquing candidate solutions proposed.

5.2.1 Quality of Usability Defect Reports

Of the 23 studies, 11 investigated the quality of the usability defect reports produced by expert and non-expert usability evaluators, and employed various assessment criteria. We identified two key topics under this research area: 1) measuring usability defect report quality; and 2) improving usability defect reports.

Measuring Usability Defect Reports Quality. In general, defect report produced by an expert evaluator had better quality than the defect reports produced by a non-expert evaluator. In most studies, non-expert evaluators are recruited from among students, while expert evaluators are from industrial practitioners. Based on the eight studies, we identified a set of criteria used to measure the quality of usability defect reports (see Table 9). We classified criteria into three categories: *report content*, *software quality model*, and *general* categories. One study did not mention assessment criteria that were used [P39].

Report content was used by six studies to measure the quality of usability defect reports [P1, P3, P5, P7, P8, P35]. The findings showed that observable user action, impact, supplementary information, available online, cause of the problem and solution proposal were the most assessed information. Only one study measured the quality of the usability defect report content using test procedure descriptions, executive summary and report layout [P8] and business goals [P35]. Of the six studies, five [P1, P4, P5, P8, P39]

revealed that non-expert usability evaluators have difficulties in describing certain usability defect information, particularly the impact, solution, supplementary information, available online, cause of the problem, and recovery steps [P1, P5, P8, P39].

With regards to software quality criteria, seven of the nine studies used clarity attributes to assess if the usability defects were described precisely, meaningful, and contains unambiguous explanation [P1, P3, P5, P7, P8, P35, P44]. We also observed that even non-expert evaluators failed to fulfil all report content criteria, but they can describe some information clearly [P1], and provide positive findings for the evaluation [P39]. However, these studies did not indicate which information non-expert evaluators can explain precisely. Other studies uniquely defined their quality attributes such as persistent [P3], justified [P3], persuasive [P44], and usefulness [P35, P44].

The general category comprised of expert judgment. Studies that relied upon expert judgment measured quality using criteria such as how similar problems were identified and use of appropriate claims to justify the problems. In order to minimize judges' bias, measures of association, bias and distribution of the judgement were calculated. In our review, two studies [P4, P7] employed professional judge ratings.

Improving Usability Defect Reports. There are six studies that emphasize ways to improve usability defect description. We break the research down to the different aspects of improvement.

Defect report content and structure—Hornbaek and Frøkjær [P3] recommend four elements for good usability defect reports: 1) include solution proposal, 2) justify why something is a problem by referring to the behavioral consequences of a problem, 3) present descriptions of problems that are complex and persistent for users, and 4) make the problem description long enough. Hornbaek and Frøkjær [P35] proposed the use of business goals in justifying usability defects, as the information would give higher utility and impact to the company. Furthermore, they found that business goals help focus the evaluation. Ko et al. [P48] suggested that the defect report title should consists of software behavior, relevant quality attribute, problem, execution context, and if the report is a defect or feature request.

Defect report format—Norgaard and Hornbaek [P44] found that software developers highly prefer a multimedia presentation, screen dump and redesign proposal for presenting usability defects because they provide ideas on the problem context.

Defect reporting tool—Faaborg and Schwartz [P27] proposed the adaptation of usability heuristics when labelling usability defects. Furthermore, a usability-shared vocabulary (such as consistency, jargon, feedback) can be useful in describing the cause and impact of user interface problems. However, this approach is highly dependent on the clarity of each heuristic's definition and use of good examples, as users of defect reporting tools may have limited HCI knowledge.

Approach to capture usability defects—Howarth et al. [P7] proposed the usability problem instance approach to record usability defects. Using this approach, they found out that expert judgment provided higher ratings for describing the cause of the problem and solution proposal description. Hornbaek and Frøkjær [P3] found that usability evaluation methods influence the level of detail of defect descriptions. For instance, problems identified with the metaphor of human thinking are more justified compared to problems found with testing aloud.

5.2.2 Classification of Usability Defects

In existing defect repositories, defects are classified as either defect (blocker, critical, major, minor, normal) or enhancement. However, this labelling scheme does not have sufficient knowledge for understanding the properties and features of various types of usability defects. This is evidenced by a number of studies available in the literature [P21, P22, P42, P46, P50]. We group the research into different goals below.

Understanding usability defect characteristics—Lal and Sureka [P22] investigated the differences, similarities and correlation between terms and usability defect types. They found that terms present in defect report titles and description are related to usability defect type. For instance, usability defects have most frequent terms of "window", "user", "zoom", "menu", and "click". In relation to usability defects, they discovered that 1) usability defects are the largest contributor to regression defects, 2) the median mean time to repair (MMTR) value for usability defects is fairly high compared to other defect types (clean-up, crash, polish, performance, regression and security), and 3) usability defects are the second highest of duplicated defect reports. Xia et al. [P50] studied the relationship between types of defects and severity. They discovered that most user interface and usability defects are assigned as block and critical severity.

Understanding the cause of usability defects—Li et al. [P42] developed a classification model for classifying defects to root cause, impact and software component. They found that graphical software is impacted by graphical user interface (GUI) defects that are mostly caused by semantic errors, such as missing features and wrong functionality.

Understanding the impact of software defects—Vetro et al. [P21] conducted an experiment to classify software defects according to ISO/IEC 9216 quality model (functionality, reliability, usability, efficiency, maintainability and portability). They found functionality and usability were the most dominant impacted quality attributes. Kreyss et al. [P46] used IBM quality measurement to categorize defect report distribution. Across the nine quality attributes (capability, usability, performance, reliability, installability, maintainability, documentation, serviceability and overall) usability was ranked as the second highest problematic quality attributes. The results from the study gave an overview of where improvements should be focused.

5.2.3 Duplicate Defect Report Analysis

Many previous studies have reported how duplicate defect reports of any sort may slow down the defect fixing process as more resources and time are needed to identify and close duplicate defects [82]. However, duplicate defects should not be ignored because they may contain additional information that may be useful to resolve defects [58]. With regard to the latter concern, we identified four studies that addressed a way usability defects are detected and handled. In the usability engineering literature, duplicate defect

report classification is referred to matching and merging, and does not appear to be an area of active research.

Matching—a process to detect duplicate problems. Vermeerena et al. [P12] analysed the usability problem's similarity based on the situation in which the problem occurred, the user's observable behavior at the time the difficulty occurred and how the user thought, felt or understood certain tasks. Hornbæk and Frøkjær [P13] studied four matching techniques (similar changes, practical prioritization, a model of Lavery et al. (1997) and the User Action Framework). Their experiment showed that similar changes produce more single problems than the other techniques and practical prioritization reaches highest level of agreement among novice evaluators. Hindle et al. [P56] used the different contextual features—architecture words, non-functional requirement words, LDA topic words and random English words to improve the accuracy of defect report deduplication. Their experiments demonstrated the effectiveness of domain-specific context in improving the quality of duplicate defect detection.

Merging—a process to consolidate similar problems. When similar problems are identified, they must be linked to the primary report the current duplicate refers. To address this process, Law and Hvannberg [P36] used a manual merging process, where evaluators record every change made to the usability problems in their own consolidated list. The results of their study found that the merging process is influenced by the evaluator effect, in which the merging rate and severity in increased when evaluators performed merging process in a group. However, confidence level, which is influenced by personal experience, does not fluctuate with the merging process.

5.2.4 Estimation for Usability Defects

In this research area, we only found one study that used usability defect data to determine defect discovery rates. Using Good-Turing discounting with a normalization procedure, Lewis [P19] revealed that higher levels of description produce a higher estimate of discovery rate.

5.2.5 Design Discussion in Usability Defect Reports

We found only three studies in software engineering that focus on correction discussion and had goals concerning user interfaces and interaction design.

Addressing and resolving usability defects—Twidale and Nichols [P21] identified two topics commonly discussed by users: 1) debate about the validity of usability problems; and 2) critiques and refinement of candidate solutions. They also express concern about usability defect solutions that may introduce ripple effects.

Understanding structure and content of design discussion—Ko and Chilana [P26] observed trends in online design discussions including establishing scope, proposing ideas, identifying design dimension, defending claims with rationale, moderating process, and making decisions. However, the temporal presentation of discussion comments was inadequate to support proposals and critiques among a broad range of users.

Supporting online forums—Raza et al. [P31] discovered that the open source community works in a collaborative environment to identify and find possible solutions to usability defects. The number of active mailing lists and messages posted on online forums indicated a significant and active support from open source community.

5.3 Research Question 3

What are the identified challenges of usability defect reporting in usability and software engineering fields?

Addressing the identified challenges of existing usability defect reporting processes and tools serves as basis for any improvement hoped for. We identified these reported challenges from both software engineering and usability engineering studies. From the software engineering perspective, these challenges include difficulties faced by reporters to report, track and manage usability defects in existing defect tracking systems. Most challenges identified in the usability engineering field are related to human factors and usability evaluation methods, while challenges in software engineering field are due to limitation of existing defect repositories. Key reported challenges we found from the analysed studies are summarized below.

Developer mindset—one of the prominent dilemmas among evaluators is when their usability defects reported get a lot less attention than they think they deserve from software developers. This situation seems to happen when software developers cannot understand the problems, especially when they do not participate in the evaluation or witnessed how users struggled to accomplish certain tasks [P1, P9, P37]. In some cases, software developers do not always agree with the higher severity ratings of usability defects given by reporters. In fact, software developers usually assess severity somewhat differently from reporters, and usability defects often end up with low severity rating and lower priority than functional defects [P37, P53]. Therefore, comparing usability defects in the context of functional defects is impractical as usability defects can be overlooked [P54].

Subjective bias—evaluating usability aspects of a system is highly subjective to an individual and thus the reporter [P26, P40]. That is, one might see one aspect of an interface is problematic, but others may not. It is thus difficult to persuade software developers or designers that the usability defects raised are indeed a real defect, that they require the same attention as functional defects, and need fixing. In fact, an agreement/disagreement between severity ratings is also seen as an effect of subjective bias when software developers or by designers evaluating their own designs [P1, P23]. This has raised questions as to whether usability defects should be reported into a shared defect database or usability defects should have their own database [P54].

Evaluator effect—our review observed that the way usability defects are described is influenced by skill and experience levels of the evaluator. From the usability engineering literature, most authors reported that expert usability evaluators are better at identifying and describing usability problems than the software developers or novice evaluators [P1, P5, P7, P8, P14]. It should be noted that inexperienced usability evaluators might feel that not all problems should be reported, and when they found a problem they do not know what information should be reported. This challenge has led to incomplete usability defect descriptions, in which a report usually does not contain

possible causes of the problems, recovery steps, possible solutions and clear reasons why something is a problem [P20, P39, P53].

Defect discovery methods—the completeness of usability defect descriptions also depends on the defect discovery methods used. [P3, P20] reported that certain methods, such as metaphors of human thinking, are more likely to have more information to justify a problem found compared to think aloud. In other words, selection of appropriate testing techniques may help evaluators to identify usability problems effectively and collect necessary usability defect information to report. However, in open source projects where often no formal usability testing is conducted, there is still a lack of mechanisms to discover and report usability defects, especially those encountered by typical or non-experienced users [P20].

Complexity management—the process of managing usability defects is a largely human task, especially when discussing design solutions in defect repositories. There are two aspects of complexity in managing usability defects we found from literature. First, the linear temporal discussion structure may not be sufficient to enable users to keep track of all the discussion elements, such as elaboration, confirmation, allocation of works, proposed fix, and revision [P40]. This makes it difficult for users to compare and critique a correction proposal, as they have to read through all of the comments. One way to minimize this challenge, is to use nested comments [3], [50], [83] so that the critiques in design discussion can be more explicit. Second, the changes to interface design might be risky, as any changes may have impacts on the other components of the system [P40], cause confusion to existing users [P26], and may involve major changes to business and technical constraints [P29, P53]. In this case, some usability defects are difficult to explain and proposing useful and usable recommendations may be hard.

Lack of appropriate channels for reporting usability defects—existing defect repositories, such as Bugzila, Trac and JIRA, were designed as text-centric mediums for functional defect reporting. This causes some usability defects that relate to user's feelings, emotions and "struggles" with an interface are difficult to explain textually [P20, P40, P57]. To overcome this limitation, defect repositories, such as Bugzilla could have a mechanism to easily and interactively record, upload, show, maintain and comment user submitted videos, images and voice [P10, P38, P57]. Furthermore, some defect repositories that were developed by and for software developers have caused usability defect reporters to fill in considerable amount of information, much of them not relevant for usability defects [P6, P38, P57]. Considering these challenges, several studies have suggested a mechanism to support non-expert users in terms of automated collection contextual metadata and cognitive information [P20, P23] and less user registration [P40].

Lack of specific guidelines for usability defect information reporting—although generic defect report templates and evaluation reports are available, most of them do not clearly define specific information that should be reported for usability defects in general and different kinds of usability defects [P29, P36, P54, P57]. For example, in assigning usability defect severity, there is no standard guidelines and rules available. According to [P36], users usually use their personal experiences as a benchmark to judge problem severity. Similarly, a lack of guidelines and exemplary recommendations make the quality of fix recommendations highly varied [P29].

6 Threats to Validity

Even though this systematic review was performed according to a well accepted process [68], [69], we cannot guarantee that we have covered all studies in this area. Each systematic literature review process described in Section 3 was exposed to some threats. We describe the threats associated with each process and the mitigation strategies used for this review.

Data source and search strategy. This review is limited to studies that were published from the year 2000 onwards. Thus, it neglects studies that were published before the year 2000. We were aware that a few studies on usability defect reporting were published in 1997 [84] and 1999 [5], but these studies were extended in other studies [85], [86], which were included in our review. Other than that, we cannot guarantee the selection of the search strings covers all terms used in both software engineering and the usability-engineering field. In this case, we tried to derive a different set of search strings for different fields of study and these are adjusted accordingly to each search engine (as described in Section 3.2.2 and Appendix B, available in the online supplemental material). Additionally, we included a *reference chaining* search as a secondary search to minimize this threat.

Study selection. The selection of studies was performed by one researcher only, which may have resulted in missing studies. However, the other authors provided detailed feedback during the review process and monitored the systematic literature review protocol execution closely. We have used clear inclusion and exclusion criteria to reduce selection bias.

Data extraction and synthesis. We found that some studies do not have clear details about the format used for reporting usability defects. In this case, we had to make assumptions on the basis of our judgment. Therefore, there is a possibility that some of the extracted results are partially inaccurate. In order to mitigate this, the other authors randomly picked several studies, refined and verified the extracted data. The earlier data extraction was then rechecked by the first author. Overall agreement was very high between the authors in terms of classification of studies and agreement on extracted data.

7 Discussion

Section 5.3 summarised the key findings to date in terms of current usability defect reporting limitations in the software engineering and usability literature. Here we draw on these findings, and the findings of other studies and surveys from Section 2.3 to provide a set of key recommendations for further research in usability defect reporting. These provide a road map for further usability defect reporting research and while many are complimentary, we order them roughly in our suggested priority order to address.

Key challenges in usability defect reporting from our systematic literature review and previous studies are that:

- Existing usability defect reporting tools are not capturing the needed information to fix the defects effectively;
- Many reporters are unsure what they need to capture or how to capture information in usability defect reports;
- There is a lot of manual effort going into usability defect reporting;
- There is a difference between reporter and evaluator/fixer points of view, in terms of information to capture and prioritization of usability defect severity;
- Usability engineering methods and tools are distinct from software engineering defect repositories resulting in unnecessary duplication of data and effort;
- Existing taxonomies, classifications, severity ratings and terminology are all limited and often inconsistent; and
- The community is unsure what information actually influences fixing of reported usability defects.

Our key recommendations to address these issues include:

- Development of a new, more comprehensive and consistent usability defect taxonomy and associated consistent terminology;
- Identify the key usability defect attributes that need to be captured to support defect correction and ensure they are captured in sufficient detail and quality;
- Provide reporters, especially novice ones, contextualized guidelines for reporting defects, including teasing out user versus system usability "difficulties" in reports;
- Better define usability defect severity and prioritization attributes;
- Provide reporters more customized reporting forms for different kinds of usability defect reporting; and
- Provide higher degrees and more effective automation in usability defect reporting tools, including leveraging usability engineering tool results better in software engineering defect reporting tools.

Addressing these key issues will both improve the reporting of usability defects but also usability defect management and correction. Less effort will be involved in reporting, more accurate reports will be produced, developers and reporters will both gain the information they need to carry out their respective roles, and more important defects will be identified, prioritized and corrected. We discuss each of these recommendations below, including proposed research to address current limitations and challenges in usability defect reporting. This research agenda requires a combination of both HCI and Software Engineering research contributions that we identify.

Recommendation R1—Develop an improved taxonomy for classifying usability defects

There are currently many usability defect taxonomies, classifications and attributes of usability defects identified in HCI literature and software engineering literature. We found many studies identifying that many usability defect reports lack sufficient attributes for classifying usability defects. A key obstacle of using existing usability defect report data is the widespread use of unstructured textual features in most current defect tracking systems. Lack of usability knowledge or different usability knowledge among reporters has produced reports that use a wide range of non-standard usability terms that complicates usability defect classification and identification. In addition, existing defect report attributes do not capture usability related information that can be directly used to filter usability defects. We observed only two studies [P30, P32] that specify the types of defects to describe usability defects across the 13 formats we identified.

There are several reasons for classifying usability defects: 1) to better identify and disclose the probable causes of the defect; 2) to highlight the impact of usability defects on the intended user task outcome; 3) to treat usability defect priorities the same way as for other defects; and 4) to quantitatively track usability defects, defect impact and defect resolution over time.

We also observed a great deal of inconsistency in the terms used in usability defect reports for specifying the same usability defect across the 13 formats found. For instance, a "severity" attribute was used in most of the formats to denote the importance of the defects to be fixed [87]. However, other than severity, some studies used impact [P12, P16], seriousness [P13, P53] and category [P5] to refer to severity. This variation of terms for one usability attribute can also be found in use of "minor, major, enhancement" for a defect's severity, while others used "severe, critical", which resulted in inconsistent data which was not comparable. Many other usability attributes are used inconsistently in terms of both name and value. This leads to even within the same project inconsistent reports that are hard to read, understand, track and prioritize.

To solve these issues, the HCI and software engineering communities need to develop a more comprehensive and agreed usability defect taxonomy. Much of this work has been established in terms of HCI usability evaluation terminology and attributes, but has been inconsistently or not applied in software engineering practice around usability defect reporting. Along with comprehensive, agreed usability defect taxonomy, an agreed set of names and meanings for usability defect attributes are needed.

Recommendation R2—Provision of key usability-related defect attributes

Following on from R1, we observed that many of the usability defect description formats in use do not define separate attributes to indicate specific key information about a usability defect. This results in many software developers with little experience reporting "usability" issues finding difficulties in understanding the reported issue. This means software developers do not always agree or understand the usability defects actually reported, even if reported at all. As a result, usability defects get less attention or are sometimes even closed off as not valid.

One way to overcome this issue is to define and capture usability defect attributes at a fine-grained level, which can reveal more detailed issues with usability characteristics, such as heuristics, defect category, location and impact. Additionally, by introducing dedicated fields/attributes to address likely interaction difficulties, the end user's feeling, and how they see an interface as problematic—so that the

usability "struggles" exemplified in the usability defect reported can better understood and appreciated by software developers using the usability defect report. This information can be used by project managers and software developers for defect management purposes, as well as provide researchers richer information on which to conduct empirical analysis of usability defect cause, impact, tracking and resolution.

In order to identify critical usability defect attributes to report, research needs to be carried out to determine both: (1) what reporters are reporting and think they should be reporting, and what developers require in order to fix usability defects; and (2) what usability defect attributes actually impact defect understanding and correcting. This could be done via surveys and interviews of reporters and developers, to get opinions of attributes required, and mining of existing defect repositories, to understand what is being reported and its impact on resolution.

To minimize unnecessary or irrelevant attributes for a particular usability defect, usability defect reporting forms could be adjusted using e.g., a contextualised question-based design so that a reporter can select specific attributes that are relevant to them. We return to this issue below.

Recommendation R3—Provide good contextualized guidelines for well-written usability defect reports

This study identified some research that defined guidelines for characterizing how usability defects should be reported. However, these lack content-related criteria that would assist reporters in collecting important and useful information for describing the defect and correcting the defect [P2, P5, P37, P52]. For instance, a good usability defect report should describe the issue precisely, but often the information that really needs to be reported is not explained clearly or even not captured at all. Inexperienced reporters in particular may think that their reports are complete, but they may actually be providing irrelevant or inadequate information.

These issues require further research into what influences the fixing of usability defects. This might include mining defect repositories for evidence of useful attributes and reports, and surveying and interviewing both reporters and developers. The findings from these kinds of studies could be used to produce better contextual guidelines that assist both reporters and software developers. Another related area for both HCI and software engineering research is studying the "evaluator effect" in terms of how it impacts the usability defect reporting. A related concept we call the "reader effect"—how software developers read, interpret and action usability defect reports—appears to be an as yet unstudied area, that with better knowledge also may improve defect reporting.

Recommendation R4—Prioritize usability defect attributes by their level of importance for software engineers

There are many separate usability defect attributes that we have identified from usability engineering studies (33 attributes). Many of them are do not appear to be important for understanding, replicating or correcting the usability defect from a software engineering perspective. Since a key aim in our research was to simplify and improve the defect reporting process, we have to identify which of the attributes have the greatest influence on defect fixing process. We could focus on capturing the ones that will have the greatest impact in convincing software developers of a problem and assisting them in prioritizing, diagnosing cause, and correcting. Related to this, we found little work on how to best prioritize usability defect reports to provide best value to end users i.e., fix those most seriously impacting usability first. As above, this requires better ways to characterise usability defects, classify, determine severity, and convey this to software project managers and developers.

To advance this research and practice outcomes a detailed survey and interviews with a large number of software engineers to determine critical attributes for them is needed. Additionally, understanding better the difference between usability defect reporter and consumer perspectives is essential. Improved usability attribute terminology and understanding in terms of impact on usability defect description and diagnosis is also needed. Mining existing defect repositories to understand what attributes seem to lead to improved correction may also assist this.

Recommendation R5—Provide reporters customised usability defect report forms

The static reporting template offered by most functional requirements-oriented software defect repositories is generally universal. These do not consider the influence of the different types of reporters, different kind and use of diverse usability evaluation methods, and the phase of development where the usability defects are found. Almost all research shows that all defect types are reported using the same generic defect reporting template. In some cases the information requested on the form is simply not relevant and some is beyond the reporter's knowledge [11]. Most are text only and do not support other forms of input collection, or make it difficult to capture and attach.

A number of enhancements to existing reporting tools have been suggested in the literature [P45, P49, P51, P55], or can be deduced from the related usability defect reporting issues discussed above. We think that using a guided reporting method where reporters are assisted with predefined attributes for input selection, online help and question/wizard-based interaction may greatly improve capture and quality of usability defect reports [31], [32]. In this way, even if the reporter has less knowledge about usability, they can still be guided to capture reasonable quality defect reports. As a result, the recorded data will be more structured, fine-grained and uniform for usability defect report management. Users should be prompted/allowed to capture relevant attributes based on types of usability defects, the reporter's profile (e.g., non-technical user, technical user, usability experts and etc.), and usability defect report attributes be prioritized based on the types of defects and relevancy.

Additionally, usability defect report form should be simple. Simplicity—the art of minimizing the amount of requested attributes in a usability defect report—is a necessary quality focus, by including only what software developers need rather than what reporters think—to make it easier for software developers to understand, replicate and fix the problems [P55]. Giving a reporter a simple set of explicitly usability-focused defect reporting forms for different kinds of defects could encourage them to report more usability defects with better outcomes, rather than imposing on them many complex, irrelevant attributes.

Another issue in usability defect reporting is that usability engineering tools and techniques are quite distinct from software engineering defect repository reporting and management tools and software engineering unit testing methods. This can cause repeated defect reporting effect when transferring usability defect information found during formal usability evaluations to project defect repositories, a waste of time, and possibility of information loss. Having a standard format that can be shared between the usability and software engineering communities would add value. However, further research to empirically study the impact of using separate and shared defect repositories would suggest a better usability defect reporting approach.

A further area for future research is to investigate what are the keys factors influencing quality usability defect reporting, from the perspective of non-technical reporters. Using this knowledge, how can next generation usability defect reporting tools be better-designed to leverage HCI knowledge, domain knowledge and end user knowledge.

Recommendation R6—Develop more automation in usability defect reporting

Much current usability defect reporting in software teams is still highly text-based and manually captured. Apart from better information capture for usability defect reports, as discussed above, more automated data capture and richer kinds of information capture are needed. Many usability engineering tools provide both of these e.g., instrumenting applications to capture traces and user interaction, recording richer user interaction and mapping to user task, and capture of video, audio, screenshots, diverse interaction (touch, sketch, gesture, accelerometer etc. as well as keyboard and mouse). However, most software engineering defect tracking tools make capture of this highly manual, uni-format (usually free format text), or make adding and manipulating attachments difficult (or impossible). There may be entirely novel approaches to usability defect reporting possible combining HCI usability engineering methods and tools with software defect reporting and management repositories.

Where possible, supporting automated capture of usability-related defect issues would enhance the reporting process, but also the replication, solution discussion and correction processes. Such data collection should include structured, contextualized reporting forms as above, but also event traces, interaction traces, screenshots, audio and video, a variety of interaction styles, especially for mobile applications, and enable software developers to view this in context with the usability defect report attributes captured. Attachments such as audio, video and interaction recordings should be interactively manipulable as in some HCI-oriented usability assessment tools.

8 Conclusion

The aim of this study was to identify the state of the art in usability defect reporting in both the software engineering and usability studies areas for fruitful future research. We performed a comprehensive literature search on five reputable online databases using multiple search strings and a two-phase screening of papers. As a result, 57 papers were selected. We developed a classification scheme (see Fig. 2) to classify these papers in accordance to our research questions. This allowed us to examine the trends and motivations in this line of research.

We divided the papers into three main categories; 1) reporting usability defect information—which is related to research on reporting the usability defect, 2) analysing usability defect data—which is related to researching the use of defect data, and 3) challenges—which refer to issues identified in current approaches to usability defect reporting and management.

In usability engineering and HCI studies, evidence showed that various diverse mechanisms are used to capture and record usability defects. This is supported by numerous defect report content and formats to present the information. However, most of these mechanisms and formats were used in isolation. That is, each mechanism and format was designed to the specific usability evaluation method and does not integrate with the central defect database. Furthermore, existing guidelines to assist reporters in writing a good usability defect description lack guidance for collecting usability defect data.

As far as quality of defect data is concerned, some studies evaluated usability defect report quality. The evaluations were conducted through comparative studies between reports produced by an expert and a non-expert evaluator. In general, defect reports produced by an expert usability evaluator had better quality than the defect reports produced by a non-expert evaluator. In terms of improving usability defect management, defect data was used for matching, merging and estimation purposes.

However, usability defect reporting has been less investigated to date in the software engineering domain. Existing studies that investigated usability defect reporting have focused especially on addressing the limitations of open source defect repositories to support usability defects. Other work focuses on improving the overall defect data quality, and discussed defect forms for functional or non-functional defects (performance, reliability, security). In contrast, our focus blends HCI, usability and software engineering needs to create defect informational content.

We observe that usability defects reported in defect tracking system and usability evaluation documents commonly suffer from mixed, inconsistent terms and values of usability defect data and insufficient attributes to classify usability defects. Although mailing lists and online forums have become an alternative interaction hub for users to discuss usability defects, especially in open source development communities, the linear sequence of communication makes it hard to extract the contextual information for developers to fix the problems. For this reason, a guided-defect reporting with more structured non-textual information to augment the unstructured textual defect reporting approach may increase the information archival value. In addition, usability defects by their nature need richer feedback such as screen, video and audio to describe and replicate. However, the use of guided-defect reporting for reporters raises several challenges and new opportunities for research of new reporting approaches, as well as the investigation of what makes a good usability defect report, what terminology to use and what critical usability defect attributes to capture.

For future research, we plan to design new methods, processes and tools for eliminating the aforementioned

limitations. In particular, we plan to develop an improved usability defect taxonomy to characterise usability defects and attributes; survey reporters and developers on their respective usability defect report needs; mine existing defect repositories to investigate more deeply what usability defect information is actually being captured; and design and prototype extensions to existing defect repositories and tools to support improved usability defect reporting process to overcome key weaknesses of the existing approaches.

Acknowledgements

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

References

[1] ISO/IEC, "Information technology—Software product quality — Part 1: Quality model," IEEE Standard ISO/IEC FDIS 9126-1:2000, 2000.
[2] C. Wilson and K. P. Coyne, "The whiteboard: Tracking usability issues: To bug or not to bug?" *Interactions*, vol. 8, pp. 15–19, 2001.
[3] M. B. Twidale, D. M. Nichols, and N. Zealand, "Exploring usability discussions in open source development," in *Proc. 38th Annu. Hawaii Int. Conf. Syst. Sci.*, 2005, pp. 1–10.
[4] M. G. Capra, "Usability problem description and the evaluator effect in usability testing," PhD Thesis. Viginia Tech, Blacksburg, VA, USA, 2006.
[5] S. L. Keenan, H. R. Hartson, D. G. Kafura, and R. S. Schulman, "The usability problem taxonomy: A framework for classification and analysis," *Empir. Softw. Eng.*, vol. 4, pp. 71–104, 1999.
[6] T. S. Andre, S. M. Belz, F. A. McCrearys, and H. R. Hartson, "Testing a framework for reliable classification of usability problems," in *Proc. Human Factors Ergonomics Soc. Annu. Meet.*, 2000, vol. 44, no. 37, pp. 573–576.
[7] 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.
[8] F. P. Simões, "Supporting end user reporting of HCI issues in open source software," PhD Thesis. Pontificia Universidade Catolica, Rio De Janeiro, 2013.
[9] A. Raza, L. F. Capretz, and F. Ahmed, "Usability bugs in open-source software and online forums," *IET Softw.*, vol. 6, no. Nov. 2011, 2012, Art. no. 226.
[10] R. Lotufo and K. Czarnecki, "Improving bug report comprehension," Generative Software Development Laboratory, University of Waterloo, Waterloo, Ontario, Canada, Tech. Rep. GSDLAB–TR 2012–09–01, 2012.
[11] 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.
[12] A. Fernandez, E. Insfran, and S. Abrahão, "Usability evaluation methods for the web: A systematic mapping study," *Inf. Softw. Technol.*, vol. 53, no. 8, pp. 789–817, Aug. 2011.
[13] R. T. Høegh and J. Stage, "The impact of usability reports and user test observations on developers' understanding of usability data: An exploratory study," *Int. J. Human–Comput. Interact.*, vol. 21, no. 2, pp. 173–196, 2006.
[14] S. Blair, "A guide to evaluating a bug tracking system," White Paper, 2004.
[15] K. Hornbaek and E. Frokjaer, "What kinds of usability-problem description are useful to developers?" in *Proc. Human Factors Ergonomics Soc. Annu. Meet.*, vol. 50, no. 24, pp. 2523–2527, 2006.
[16] 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.
[17] L. Despalatović, "The usability of free/libre/open source projects," *Int. J. Comput. Inf. Technol.*, vol. 2, no. 5, pp. 958–963, 2013.
[18] Y. C. Cavalcanti, et al., "Challenges and opportunities for software change request repositories: A systematic mapping study," *J. Softw.-Evol. Process*, vol. 26, pp. 1–37, 2013.
[19] P. K. Chilana, A. J. Ko, J. O. Wobbrock, T. Grossman, and G. Fitzmaurice, "Post-deployment usability: A survey of current practices," in *Proc. CHI Conf. Human Factors Comput. Syst.*, 2011, pp. 2243–2246.
[20] A. J. Ko, M. J. Lee, V. Ferrari, S. Ip, and C. Tran, "A case study of post-deployment user feedback triage," in *Proc. 4th Int. Workshop Cooperative Human Aspects Softw. Eng.—CHASE'11*, 2011, pp. 1–8.
[21] A. Raza, L. F. Capretz, and F. Ahmed, "Maintenance support in open source software projects," in *Proc. 8th Int. Conf. Dig. Inf. Manage.*, 2013, pp. 391–395.
[22] J. Harty, "Finding usability bugs with automated tests," *Commun. ACM*, vol. 54, pp. 44–49, 2011.
[23] F. T. W. Au, S. Baker, I. Warren, and G. Dobbie, "Automated usability testing framework," *Australas User Interface Conf.*, vol. 76, pp. 55–64, 2008.
[24] S. Baker, F. Au, G. Dobbie, and I. Warren, "Automated usability testing using HUI analyzer," in *Proc. Australian Softw. Eng. Conf.*, 2008, pp. 579–588.
[25] A. Alsumait and A. Al-Osaimi, "Usability heuristics evaluation for child e-learning applications," *J. Softw.*, vol. 5, pp. 654–661, 2010.
[26] C. Gutwin and S. Greenberg, "The mechanics of collaboration: Developing low cost usability evaluation methods for shared workspaces," in *Proc. Workshop Enabling Technol. Infrastructure Collaborative Enterprises*, 2000, pp. 98–103.
[27] J. Mankoff, et al., "Heuristic evaluation of ambient displays," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2003, pp. 169–176.
[28] D. Pinelle, N. Wong, T. Stach, and C. Gutwin, "Usability heuristics for networked multiplayer games," in *Proc. ACM Int. Conf. Supporting Group Work*, 2009, pp. 169–178.
[29] L. O. Bligård and A. L. Osvalder, "Enhanced cognitive walkthrough: Development of the cognitive walkthrough method to better predict, identify, and present usability problems," *Adv. Human-Comput. Interact.*, vol. 2013, 2013, Art. no. 9.
[30] M. P. González, J. Lorésc, and A. Granollers, "Enhancing usability testing through datamining techniques: A novel approach to detecting usability problem patterns for a context of use," *Inf. Softw. Technol.*, vol. 50, pp. 547–568, 2008.
[31] P. K. Chilana, A. J. Ko, and J. O. Wobbrock, "LemonAid: Selection-based crowdsourced contextual help for web applications," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2012, pp. 1549–1558.
[32] J. Matejka, T. Grossman, and G. Fitzmaurice, "IP-QAT: In-product questions, answers, & tips," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 175–184.
[33] J. Itkonen, M. V. Mäntylä, and C. Lassenius, "Defect detection efficiency: Test case based vs. exploratory testing," in *Proc. 1st Int. Symp. Empirical Softw. Eng. Measurement*, 2007, pp. 61–70.
[34] H. Yehuda and J. McGinn, "Coming to terms: Comparing and combining the results of multiple evaluators performing heuristic evaluation," in *Proc. Conf. Human Factors Comput. Syst.*, 2007, vol. 2, pp. 1899–1904.
[35] H. Petrie and C. Power, "What do users really care about? A comparison of usability problems found by users and experts on highly interactive websites," in *Proc. Conf. Human Factors Comput. Syst.*, 2012, pp. 2107–2116.
[36] H. R. Hartson, T. S. T. Andre, and R. R. C. Williges, "Criteria for evaluating usability evaluation methods," *Int. J. Hum. Comput. Interact.*, vol. 13, pp. 1–35, 2001.
[37] P. Koutsabasis, T. Spyrou, and J. Darzentas, "Evaluating usability evaluation methods: Criteria, method and a case study," in *Proc. 12th Int. Conf. Human-Comput. Interaction: Interaction Design Usability*, 2007, pp. 569–578.
[38] A. Seffah, M. Donyaee, R. B. Kline, and H. K. Padda, "Usability measurement and metrics: A consolidated model," *Softw. Qual. J.*, vol. 14, pp. 159–178, 2006.
[39] J. Sauro and E. Kindlund, "A method to standardize usability metrics into a single score," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2005, pp. 401–409.
[40] R. T. Høegh, "Case study: Integrating usability activities in a software development process," *Behav. Inf. Technol.*, vol. 27, pp. 301–306, 2008.
[41] J. G. Redish, et al., "Usability in practice: Formative usability evaluations—evolution and revolution," in *Proc. Comput. Human Interaction*, 2002, pp. 885–890.
[42] K. Hornbæk, "Current practice in measuring usability: Challenges to usability studies and research," *Int. J. Hum. Comput. Stud.*, vol. 64, pp. 79–102, 2006.

[43] M. Y. Ivory and M. A. Hearst, "The state of the art in automating usability evaluation of user interfaces," *ACM Comput. Surv.*, vol. 33, no. 4, pp. 470–516, 2001.
[44] N. E. Jacobsen, M. Hertzum, and B. E. John, "The evaluator effect in usability tests," in *Proc. Human Factors Ergonomics Soc. Annu. Meet.*, 1998, pp. 255–256.
[45] 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.
[46] A. Raza, L. F. Capretz, and F. Ahmed, "Improvement of open source software usability: An empirical evaluation from developers' perspective," *Adv. Softw. Eng.*, vol. 2010, pp. 1–12, 2010.
[47] A. Følstad, E. L.-C. Law, and K. Hornbæk, "Outliers in usability testing: How to treat usability problems found for only one test participant?" in *Proc. 7th Nordic Conf. Human-Comput. Interaction Making Sense Through Design*, 2012, Art. no. 257.
[48] K. Hornbæk and E. Frøkjær, "A study of the evaluator effect in usability testing," *Human-Comput. Interact*, vol. 23, pp. 251–277, 2008.
[49] E. L. Law and E. T. Hvannberg, "Consolidating usability problems with novice evaluators," in *Proc. 5th Nordic Conf. Human-Comput. Interaction: Building Bridges*, 2008, pp. 495–498.
[50] F. Botella and A. Peñalver, "A new proposal for improving heuristic evaluation reports performed by novice evaluators," in *Proc. Chilean Conf. Human-Comput. Interaction*, 2013, pp. 72–75.
[51] M. Aziz and R. D. Macredie, "Proposing a perceived ease of use factors taxonomy for information system use," in *Proc. IEEE SoutheastCon*, 2005, pp. 468–476.
[52] L. Gorlenko and P. Englefied, "Usability error classification: Qualitative data analysis for UX practitioners," in *Proc. ACM Conf. Human Factors Comput. Syst.*, 2006, vol. 2, pp. 803–808.
[53] R. Khajouei, L. W. P. Peute, A. Hasman, and M. W. M. Jaspers, "Classification and prioritization of usability problems using an augmented classification scheme," *J. Biomed. Inform.*, vol. 44, no. 6, pp. 948–957, 2011.
[54] D.-H. Ham, "A model-based framework for classifying and diagnosing usability problems," *Cogn. Technol. Work*, vol. 16, pp. 373–388, 2014.
[55] M. Hassenzahl, "Prioritizing usability problems: Data-driven and judgement-driven severity estimates," *Behav. Inf. Technol.*, vol. 19, pp. 29–42, 2000.
[56] A. Sureka, "Learning to classify bug reports into components," in *Proc. 50th Int. Conf. Objects, Models, Components, Patterns*, 2012, pp. 288–303.
[57] S. Breu and J. Sillito, "Information needs in bug reports: Improving cooperation between developers and users," in *Proc. ACM Conf. Comput. Supported Cooperative Work*, 2010, pp. 301–310.
[58] 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, Sep./Oct. 2010.
[59] E. I. Laukkanen and M. V. Mantyla, "Survey reproduction of defect reporting in industrial software development," in *Proc. Int. Symp. Empirical Softw. Eng. Measurement*, 2011, pp. 197–206.
[60] T. Zimmermann and S. Breu, "Improving bug tracking systems," in *Proc. 31st Int. Conf. Softw. Eng. - Companion*, 2009, pp. 247–250.
[61] S. Zaman, B. Adams, and A. E. Hassan, "Security versus performance bugs: A case study on Firefox," in *Proc. 8th Working Conf. Mining Softw. Repositories*, 2011, pp. 93–102.
[62] A. Nistor, T. Jiang, and L. Tan, "Discovering, reporting, and fixing performance bugs," in *Proc. 10th Working Conf. Mining Softw. Repositories*, 2013, pp. 237–246.
[63] S. Zaman, B. Adams, and A. E. Hassan, "A qualitative study on performance bugs," in *Proc. 9th IEEE Working Conf. Mining Softw. Repositories*, 2012, pp. 199–208.
[64] P. Anbalagan, M. Vouk, C. Science, and N. Carolina, "An empirical study of security problem reports in Linux distributions," in *Proc. 3rd Int. Symp. Empirical Softw. Eng. Measurement*, 2009, pp. 481–484.
[65] M. Terry, M. Kay, and B. Lafreniere, "Perceptions and practices of usability in the free/open source software (FOSS) community," in *Proc. 28th Int. Conf. Human Factors Comput. Syst.*, 2010, pp. 1–10.
[66] D. M. Nichols and M. B. Twidale, "The usability of open source software: Analysis and prospects," *Open Source Softw. Business: Issues Perspectives*, Ravi Kumar, Hyderabad, India: The ICFAI University Press, pp. 167–188, 2006.
[67] B. A. Kitchenham, et al., "Refining the systematic literature review process—two participant-observer case studies," *Empir. Softw. Eng.*, vol. 15, no. 6, pp. 618–653, Jun. 2010.
[68] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proc. 12th Int. Conf. Eval. Assessment Softw. Eng.*, 2008, pp. 68–77.
[69] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University, United Kingdom, Tech. Rep. EBSE-2007-01, 2007.
[70] P. Mclnerney, C. Pantel, and K. Melder, "Managing usability defects from identification to closure," in *Proc. Extended Abstracts Human Factors Comput. Syst.*, 2001, pp. 497–498.
[71] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, Jan. 2009.
[72] D. Maplesden, E. Tempero, J. Hosking, and J. C. Grundy, "Performance analysis for object-oriented software: A systematic mapping," *IEEE Trans. Softw. Eng.*, vol. 41, no. 7, pp. 691–710, Jul. 2015.
[73] N. Salleh, E. Mendes, and J. Grundy, "Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review," *IEEE Trans. Softw. Eng.*, vol. 37, no. 4, pp. 509–525, Jul./Aug. 2011.
[74] P. Achimugu, A. Selamat, R. Ibrahim, and M. Naz, "A systematic literature review of software requirements prioritization research," *Inf. Softw. Technol.*, vol. 56, no. 6, pp. 568–585, 2014.
[75] G. S. Walia and J. C. Carver, "A systematic literature review to identify and classify software requirement errors," *Inf. Softw. Technol.*, vol. 51, no. 7, pp. 1087–1109, Jul. 2009.
[76] B. A. Kitchenham, E. Mendes, and G. H. Travassos, "Cross versus within-company cost estimation studies: A systematic review," *IEEE Trans. Softw. Eng.*, vol. 33, no. 5, pp. 316–329, May 2007.
[77] M. Nørgaard and R. T. Høegh, "Evaluating usability – using models of argumentation to improve persuasiveness of usability feedback," in *Proc. 7th ACM Conf. Designing Interactive Syst.*, 2008, pp. 212–221.
[78] E. T. Hvannberg, E. L.-C. Law, and M. K. Lárusdóttir, "Heuristic evaluation: Comparing ways of finding and reporting usability problems," *Interact. Comput.*, vol. 19, no. 2, pp. 225–240, Mar. 2007.
[79] A. Bruun and J. Stage, "Barefoot usability evaluations," *Behav. Inf. Technol.*, vol. 33, no. 11, pp. 1148–1167, Feb. 2014.
[80] M. G. Capra, "Comparing usability problem identification and description by practitioners and students," in *Proc. Human Factors Ergonomics Soc. Annu. Meet.*, 2007, pp. 474–477.
[81] 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.
[82] Y. C. Cavalcanti, et al., "The bug report duplication problem: An exploratory study," *Softw. Qual. J.*, vol. 21, pp. 39–66, 2013.
[83] R. V. Lotufo, "Towards next generation bug tracking systems," Univ. Waterloo, Waterloo, Ontario, Canada, 2013.
[84] D. Lavery, G. Cockton, and M. P. Atkinson, "Comparison of evaluation methods using structured usability problem reports," *Behaviour Inf. Technol.*, vol. 16. pp. 246–266, 1997.
[85] T. S. Andre, H. R. Hartson, and R. C. Williges, "Determining the effectiveness of the usability problem inspector: A theory-based model and tool for finding usability problems," *Hum. Factors J. Hum. Factors Ergon. Soc.*, vol. 45, pp. 455–482, 2003.
[86] G. Cockton, A. Woolrych, and M. Hindmarch, "Reconditioned merchandise: Extended structured report formats in usability inspection," in *Proc. Extended Abstracts Human Factors Comput. Syst.*, 2004, pp. 1433–1436.
[87] I. Herraiz, D. M. German, M. Jesus, U. Rey, and J. Carlos, "Towards a simplification of the bug report form in eclipse," in *Proc. Int. Working Conf. Mining Softw. Repositories*, 2008, pp. 145–148.

Nor Shahida Mohamad Yusop received the MSc degree in computer science - real time software engineering from the Universiti Teknologi Malaysia, in 2004. Before joining academia, she worked nearly five years as software tester in telecommunication industry. She is currently working towards the PhD degree in the Department of Computer Science and Software Engineering, Swinburne University of Technology, Australia. Her study is sponsored by the Ministry of Higher Education Malaysia. She holds a lecturing position in the Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Malaysia. Her research interests are in the areas of open source software, usability testing, software testing, and requirement analysis.

John C. Grundy received the BSc (Hons), MSc, and PhD degrees in computer science from the University of Auckland, New Zealand. He is currently pro vice-chancellor ICT innovation and translation and professor of software engineering with Deakin University, Melbourne, Australia. Previously, he was professor of software engineering and the head of electrical and computer engineering with the University of Auckland, New Zealand, and professor of software engineering and the dean of the School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia. He is an associate editor of the *IEEE Transactions on Software Engineering*, the *Automated Software Engineering Journal*, and the *IEEE Software*. His current interests include domain-specific visual languages, model-driven engineering, large-scale systems engineering, and software engineering education. He is a member of the IEEE and the IEEE Computer Society. More details about his research can be found at https://sites.google.com/site/johncgrundy/

Rajesh Vasa received the PhD degree from Swinburne University of Technology, Melbourne, Australia. He is a professor of software and technology innovation, and currently leads the innovation efforts at Deakin Software and Technology Innovation Lab. He has more than 2 decades of experience spanning both industry and academia with deep skills in data science, artificial intelligence and complex software systems design. His career spans roles in development, operations and executive leadership in projects and organisations across the world. Recent work included building intelligent homes for aged care, reducing traffic congestion, deep learning and neural networks in healthcare, and automating the process of innovation. In the context of software engineering, his focus is on sustainable software evolution, and software architecture.