Towards the Next Generation of Bug Tracking Systems

Sascha Just
Saarland University
Saarbrücken, Germany
just@st.cs.uni-sb.de

Rahul Premraj
Saarland University
Saarbrücken, Germany
premraj@cs.uni-sb.de

Thomas Zimmermann
University of Calgary
Calgary, AB, Canada
tz@acm.org

Abstract

Developers typically rely on the information submitted by end-users to resolve bugs. We conducted a survey on information needs and commonly faced problems with bug reporting among several hundred developers and users of the APACHE, ECLIPSE and MOZILLA projects. In this paper, we present the results of a card sort on the 175 comments sent back to us by the responders of the survey. The card sort revealed several hurdles involved in reporting and resolving bugs, which we present in a collection of recommendations for the design of new bug tracking systems. Such systems could provide contextual assistance, reminders to add information, and most important, assistance to collect and report crucial information to developers.

1. Introduction

In many software projects, bug tracking systems play a central role: they allow users to communicate with developers to let them know about any problems and to request new features. In addition, developers can keep track of any unresolved bugs and request more information from users. However, most bug tracking systems are far from perfect. Many of them are merely better interfaces to a database that stores all reported bugs. As a result, they often ask too much from end-users who are not familiar with development practices. At the same time they cause frustration for developers who are disappointed about the quality of bug reports submitted by users.

In order to find out which information matters most for developers and what problems they face while fixing bugs, we asked 872 developers (out of which 156 responded) from the APACHE, ECLIPSE, and MOZILLA projects to complete a survey on the quality of bug reports. In addition, we contacted 1,354 reporters (out of which 310 responded) from the same projects to complete a similar survey, in which we asked about the information they typically provide and the information that is most difficult to collect.

In an earlier paper [4], we reported quantitative results of our survey, which can be briefly summarized as follows:

- Most helpful for developers are steps to reproduce, stack traces, and test cases. The most serious problems that they face are incorrect steps to reproduce and incomplete information.
- Bug duplicates were not considered as harmful, because they often provide additional information.
- Reporters consider test cases, steps to reproduce, code examples, and stack traces as most difficult to collect.
- There was evidence for an *information mismatch*, i.e., reporters provide not the information that developers need most. Yet reporters seem to know what is important for developers.

These results indicate a need for better bug reporting tools that assist reporters in collecting relevant information. As part of the survey, we received 175 comments by developers and reporters about what makes good bug reports and how bug reporting systems could be improved.

In this paper, we report the results of a qualitative analysis of these comments. To get a first grasp on the feedback, we organized comments with a card sort (Section 2) to identify common themes and problems in bug reporting (Section 3). Knowing what information developers need and what hurdles developers and users face with today's bug tracking systems is crucial to develop the next-generation of bug tracking systems. We close the paper with related work (Section 4) and consequences (Section 5).

2. Card Sort

We applied a card sort to organize the comments into hierarchies to deduce a higher level of abstraction and identify common themes in the participants' feedback. Card sorting is an inexpensive and user-centered sorting technique that is widely used in information architecture to create mental models and derive taxonomies from input [2].

There are three phases within a card sort: (1) *preparation*, in which participants and contents of the card sort are selected; (2) *execution*, where the indexed cards are sorted into meaningful groups with a descriptive title; and lastly, (3) *analysis*, in which the cards are sorted to form more abstract hierarchies that are used to deduce themes.

In traditional card sorts, every statement by a participant results in the creation of exactly one card. However, in our case the participants often provided several unrelated statements in a comment. Therefore we split comments into more atomic parts (statements), which was done by the first author of this paper.

2.1. Data Preparation

We first cleaned the data because not every comment contained useful information ("no comment" or "please send me the results of the survey"). Filtering out such comments left us with 120 comments that served as input for card creation. As described above, comments addressing multiple topics were split using an interactive command line tools we designed for this specific task.

The resulting 237 statements were printed on index cards, containing a short title describing the central theme of the card, the participant's role (reporter/developer) and affiliation (APACHE/ECLIPSE/MOZILLA), and the respective text of the statement.

2.2. Card Sort Execution and Analysis

We sorted the cards manually because none of the available card sorting tools supported building hierarchies. Our aim was to derive topics from the initial set of 237 cards, which is why we chose to do an open card sort. Here, groups are established while doing the card sort as opposed to having pre-determined groups. Our card sort yielded the following nine topics:

1. *Important information.* This topic contains everything related to information provided in bug reports, e.g., the value of information and how difficult it is to provide. Most participants commented on steps to reproduce, test cases and the difference between observed and expected behavior. (→ Section 3.1)
2. *User component.* The second largest topic contains comments related to users. Many of them wished for more feedback on reported bugs. Several comments pointed out the different levels of users. On the one hand there are experienced reporters who can easily provide information needed by developers. On the other hand, others struggle with how to collect relevant information. (→ Section 3.2)
3. *Quality of information.* The quality of bug reports is another important topic. Many comments discussed the consequences of missing information. In this topic more developers provided feedback than reporters.
4. *Communication.* In this topic we collected the comments with respect to communication problems between developers and reporters and how they slow down the process. The largest concept was abuse of the system and several reporters were dissatisfied with the large and overwhelming number of discussions in bug reports.
5. *Reduce the work of developers.* This topic contains ideas how work for developers could be reduced, either by better tool support (e.g., automatic setup of workspaces) or better bug reports with more data.
6. *Duplicates and search.* Many participants commented on duplicates and on how the search feature in bug tracking systems could be improved. (→ Section 3.3)
7. *Automation.* The comments in this topic are about automation in bug tracking systems. Reporters consider automatic closing of bug reports as annoying. At the same time they acknowledged the value and importance of tools that automatically collect data.
8. *Creation process.* This topic describes issues with the interface of bug tracking systems when filing new bug reports, e.g., difficult attachment handling.
9. *Triaging.* This topic describes comments on the triaging process. Most comments discussed that assigning of bugs takes too long.

3. Discussion

In this section, we discuss several comments by the survey participants that lead to recommendations how bug tracking systems could be improved.

3.1. Better Tool Support

Collect information. Several reporters pointed out the need for tools that help them to collect information that they need to file bug reports. Ideally, such tools would be integrated in the software itself or in its bug reporting system.

> *"Sometimes I wish for a special UI-tracker, which tells me what I have done to get into this."* **[Comment 64]**

> *"[M]aybe somehow it could be made possible to report bugs more like recording a macro"* **[Comment 65]**

Such support is likely to lead to better bug reports. One example is ECLIPSE bug 113206, which was awarded "Best of Bugzilla" by Ward Cunningham [5]. In this bug report, Martin Burger used a flash movie to demonstrate the rather complicated steps to reproduce. He stopped the video at important points where he added annotations to draw the attention of the developers to the crucial parts.

Recommendation #1: *Provide tool support for users to collect and prepare information that developers need.*

Internationalization. Bug reports that are not written in English are often closed immediately, although the software is internationalized.

> *"Another frustrating issue with bug reporting sites is insensitivity to language issues. I've seen bugs immediately closed because they weren't filed in English, without even asking or waiting for someone to translate it into English."*
> **[Comment 56]**

Recommendation #2: *Find volunteers to translate bug reports filed in foreign languages. Ideally the bug tracking system should provide support for this.*

3.2. Engage Reporters

User levels. Often users of a software have different levels of knowledge, as pointed out by one developer.

> *"In OSS, there is a big gap with the knowledge level of bug reporters. Some will include exact locations in the code to fix, while others just report a wierd behavior that is difficult to reproduce. In Eclipse, experienced users know that the Error log exists, so they can provide stack traces and errors [. . .]"* **[Comment 98]**

Guidance through bug reporting is desirable, especially for less experienced users, e.g., by helping to collect certain information, splitting up bug reporting across multiple pages. However, experienced bug reporters prefer to have one page where they can provide everything.

In addition to different levels of knowledge, not all reporters know what information is important for developers.

> *"it's easily forgotten that many peoples' brains just aren't wired the same as ours, and they just don't understand what we're asking for in bug reports, and why!"* **[Comment 97]**

Recommendation #3: *Provide different user interfaces for each user level (novice, expert). Give cues to inexperienced reporters what information they should provide and how they can collect it.*

Reward reporters. Good quality reports are worthy of rewards: the "Mozilla Security Bug Bounty Program" [11] awards US $500 and a Mozilla T-shirt for every critical security bug reported. In 2006, the Eclipse Foundation had a "Callisto Simultaneous Release Bug Finding Contest" [6]. In this contest, any developer who saw a great bug report marked that bug with the "greatbug" keyword. Both the reporter and triager of this bug then received a "I Helped Callisto" shirt and participated in a random drawing of prizes such as iPods and mountain bikes.

Recommendation #4: *Do not just fix bugs, also reward reporters, when they do a good job.*

Reporter reputation. Several developers pointed out that reporters who are well known, either personally or through well-written past bug reports, will get more attention.

> *"Well known reporters usually get more consideration than unknown reporters, assuming the reporter has a pretty good history in bug reporting. So even if a "well-known" reporter reports a bug which is pretty vague, he will get more attention than another reporter, and the time spent trying to reproduce the problem will also be larger."* **[Comment 55]**

An improvement to bug tracking systems would be to introduce *reputation* into user profiles. This would help developers to quickly identify the experience of a reporter, even when they do not know him personally. Hooimeijer and Weimer measured the reputation of reporters as the success rate, i.e, the percentage of submitted bug reports that were fixed [8]. Ideally, reputation would have two components: a project-independent one that tells the experience with bug reporting in general and a project-specific one that tells the experience for a given project.

Recommendation #5: *Integrate reputation into user profiles to mark experienced reporters.*

3.3. Duplicate Bug Reports and Search

Better search facility. Most reporters are aware that they should check first, whether a bug has already been filed. However, nine reporters commented on the limited search functionality in bug tracking systems and requested features such as regular expressions and a Google like search.

> *"It's very hard to find possible duplicates, when filing bugs. The 'search' tool in Bugzilla is very poor - it seems that a search for 'quick brown fox' will return results for 'quick OR brown OR fox', without prioritizing 'quick AND brown AND fox'. Furthermore, there's no way to assign or search keywords for a bug [. . .]"* **[Comment 71]**

> *"let me at a minimum enter ebay style search strings for finding relevant bugs [. . .] I'd expect regular expressions."*
> **[Comment 72]**

Recommendation #6: *Provide a powerful, yet simple and easy-to-use feature to search bug reports.*

Information in bug duplicates can be useful. Many bug reporting guidelines consider duplicates of bug reports to be harmful. When a bug report is identified as a duplicate, it is simply closed and the information discarded, which in the long term discourages users from submitting bug reports. They get reluctant to provide additional information, once they see a bug report has already been filed.

> *"Typically bugs I have reported are already reported but by much less savvy people who make horrible reports that lack important details. It is frustrating to have spent lots of time making an exceptionally detailed bug report to only have it marked as a duplicate [...]"* **[Comment 18]**

Developers also suggested that bug duplicates are not always bad, they often add important details.

> *"Duplicates [...] often add useful information. That this information were filed under a new report is not ideal thought."* **[Comment 19]**

Recommendation #7: *Encourage users to submit additional details, ideally to an already existing bug report. Provide tool support for merging bugs.*

4. Related Work

There is plenty of anecdotical evidence on what makes good bug reports and how to efficiently report bugs. For instance, Joel Spolsky described how to achieve painless bug tracking [12] and numerous articles and guidelines on effective bug reporting float around the Internet (e.g., [7]). Still, the comments by the participants of our survey suggest that bug reporting tools are far from being painless and that there is lots of room for improvement.

In our own previous work, we reported the results of our survey without a detailed analysis of the comments. In addition, we proposed a tool to measure the quality of bug reports that also provides feedback to reporters how they can improve their bug report and increase their chances of getting it fixed quickly [3,4].

In order to inform the design of new bug reporting tools, Ko et al. [10] conducted a linguistic analysis of the titles of bug reports. They observed a large degree of regularity and a substantial number of references to visible software entities, physical devices, or user actions. Their results suggest that future bug tracking systems should collect data in a more structured way. In 2004, Antoniol et al. [1] pointed out the lack of integration between version archives and bug databases. Providing such integration allows queries to locate the most faulty methods in a system. While the lack of integration was problematic a few years ago, things have changed: the Mylyn tool by Kersten and Murphy [9] allows to attach a task context to bug reports so that changes can be tracked on a very fine-grained level.

5. Conclusion and Consequences

In this paper we presented the results from a quantitative analysis on the feedback that we received by 175 developers and users of the APACHE, ECLIPSE, MOZILLA projects. The result of this discussion is a list of seven recommendations for the design of new bug tracking systems. In our future work, we will construct design prototypes for such systems and test their usability.

To learn more about our work on bug tracking systems and mining software archives, visit

`http://www.softevo.org/`

Acknowledgments. Many thanks to Nicolas Bettenburg, Martin Burger, Maximilian Grothusmann, Christian Holler, and the anonymous VL/HCC reviewers for valuable suggestions on earlier revisions of this paper. A special thanks to all developers and users of the APACHE, ECLIPSE, and MOZILLA projects who responded to our survey.

References

[1] G. Antoniol, H. Gall, M. D. Penta, and M. Pinzger. Mozilla: Closing the circle. Technical Report TUV-1841-2004-05, Technical University of Vienna, 2004.

[2] I. Barker. What is information architecture? KM Column, available at http://www.steptwo.com.au, April 2005.

[3] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. Quality of bug reports in Eclipse. In *Proceedings of the 2007 OOPSLA Workshop on Eclipse Technology eXchange (ETX)*, pages 21–25, October 2007.

[4] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In *FSE '08: Proceedings of the 16th ACM SIGSOFT International Symposium on the Foundations of Software Engineering*. ACM Press, November 2008. To appear.

[5] W. Cunningham. Best of bugzilla. http://eclipse-projects.blogspot.com/2005/12/best-of-bugzilla.html, 2005.

[6] Eclipse Foundation. Callisto simultaneous release bug finding contest. http://www.eclipse.org/projects/callisto-files/callisto-bug-contest.php.

[7] E. Goldberg. Bug writing guidelines. https://bugs.eclipse.org/bugs/bugwritinghelp.html. Last accessed 2007-08-04.

[8] P. Hooimeijer and W. Weimer. Modeling bug report quality. In *ASE '07: Proceedings of the International Conference on Automated Software Engineering*, pages 34–43, 2007.

[9] M. Kersten and G. C. Murphy. Using task context to improve programmer productivity. In *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2006)*, pages 1–11, 2006.

[10] A. J. Ko, B. A. Myers, and D. H. Chau. A linguistic analysis of how people describe software problems. In *Proceedings of the Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2006)*, pages 127–134, 2006.

[11] Mozilla.org. Mozilla security bug bounty program. http://www.mozilla.org/security/bug-bounty.html.

[12] J. Spolsky. *Joel on Software*. APress, US, 2004.