GSDLAB TECHNICAL REPORT

Improving Bug Report Comprehension

Rafael Lotufo, Krzysztof Czarnecki

GSDLAB–TR 2012–09–01 Sep 2012

Generative Software Development Lab

University of Waterloo

Generative Software Development Laboratory
University of Waterloo
200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1

WWW page: http://gsd.uwaterloo.ca/

Improving Bug Report Comprehension

Rafael Lotufo, Krzysztof Czarnecki

May 1st, 2011

Abstract

Developers need to reason about data in bug reports to diagnose problems and compare proposed solutions. Current bug tracking systems' interfaces, however, collect and present data as a conversation. Previous work shows that this hinders the ability of users to consult, reason, and analyze information important to resolve bugs. This work implements BugBot, a bug tracking system that eliminates comments as the main component of bug reports, promoting information instead of conversation. An evaluation for 6 months on a group of 9 users finds that BugBot facilitates reasoning about bug report information and improves its archival value.

1 Introduction

Bug tracking systems are a communication medium for users to report and resolve software bugs. Since it is estimated that as much as 50% of software development efforts are spent in bug resolution [3], bug tracking systems often become a fundamental ecosystem for software development projects, and virtually all software quality assurance processes rely on one.

Bug resolution is a highly collaborative task. It progresses as users share *diagnostic* data about a failure, allowing developers to characterize and understand its causes. Developers need to reason about the execution results of test cases for a series of software revisions, for example, to conclude that a bug is a regression. Once the bug has been understood, users share *solutions* to eliminate the fault, either as ideas, or source code [2, 4]. To select the most appropriate solution, developers then need to compare the proposed solutions and reason about their differences. Diagnostic and solution data, however, should be presented appropriately so that it becomes *information* that can be easily analyzed [5]. We claim, therefore, that data in bug tracking systems should be organized to *facilitate reasoning* about diagnostics and about solutions.

Bug tracking systems also serve as an archive of existing and past bugs. Users can, for example, check if a failure they have experienced has already been acknowledged by the development community, check how the resolution of a particular bug is progressing, look for workarounds to similar issues, and gather analytics on previous bugs and their causes. A random sample of 600 bugs from the Mozilla, Chrome, and Debian bug tracking systems shows that 50% of bugs take at least 18, 31, and 71 days, respectively, to be closed, and 50% of bugs still receive comments 7, 12, and 27 days after being closed. This indicates that the attention that bug reports receive spans large periods of time. For users to easily understand and locate relevant information the first time they consult a bug report or after much time since last consulting it, bug reports should, thus, have high *archival value*.

The way that traditional bug tracking systems collect and organize data as a linear sequence of *comments* ordered by time, however, increases the difficulty for reasoning about data and diminishes bug report archival value. Ko and Chilana [7] find that the linear presentation of messages hinders the ability to view and compare design proposals, critiques, and software behaviour; Lotufo [9] shows that developers appreciate summarization tools that facilitate consulting bug reports; Konstan [8] and Mamykina [11] argue that the conversational nature of comments is highly contextual and requires users to read the entire dialog to understand it.

We propose to facilitate reasoning about bug reports and to increase their archival value through a significant change on how its data is collected and organized. We create BugBot, a bug tracking system that replaces the traditional forum-like user interface of today's bug tracking systems with an interface that encourages users to focus on the most relevant information for bug reports (diagnostics and solutions), to create informational content instead of conversational content.

We run BugBot with a team of 9 users throughout a period of 6 months and evaluate its benefits and drawbacks via interviews and quantitative and qualitative usage analysis. Users find that the focus on informational instead of conversational content helps them build a better mental model of the bug—thus facilitating reasoning—and eases locating relevant information—thus improving the archival value of bugs. While there has been a reluctance from the development community to change the traditional interface for bug tracking systems, this work shows that it is possible and advantageous to do so.

2 Background

2.1 Improving Content in Bug Tracking Systems

Previous work on improving content in bug tracking systems has focused exclusively on aggregating external information to enrich content or on filtering out content that is presumed to be irrelevant. We are not aware of previous work abolishing the long-lived convention of building bug reports as a list of comments.

To aid answering questions that commonly occur during bug resolution, Ankolekar et al. [1] enriches bugs with links to external resources mentioned in bug report comments, such as commit logs and developer profiles. Bettenburg et al. [2] analyzes the quality of a bug report's description and suggests users to add information that is absent, such as stack traces, steps to reproduce, or screen shots.

As for filtering content from bug reports, Rastkar et al. [12] and Lotufo et al. [9] implement *automatic* methods of information filtering to summarize bug reports. Alternatively, Lotufo et al. [10] propose the use of game mechanisms in bug tracking systems to implement a collaborative filtering mechanism.

2.2 Mailing Lists, Forums, Q&A Sites

Discussions in bug reports are structured similar to discussions in mailing lists and community forums. Previous work studying communication in such ecosystems have found that they contain much frequent changes of topic, thus making such content hard to follow and digest [8, 11]. Threaded comments have been used to try to better handle changes of topic, but with limited success, since conversations do not work well with threads [13].

Q&A sites improve on mailing lists and forums by organizing information differently: they categorize content as being either a question or answer and allow localized comments to discuss each question or answer individually. Harper [6] argues, however, that many Q&A sites suffer much of the issues found in mailing lists and forums, since they allow conversational types of questions, which also have low archival value.

Informational questions, however, have much higher archival value. Mamykina et al. [11] investigate Stack Overflow, one of the most successful Q&A sites, and find that the quality of its content comes from valuing information over conversation. This success is achieved, however, not only through a careful design of its interface and its game mechanisms, but also by its designers participating in the community and effectively discouraging conversations and promoting information.

3 Bug Tracking Systems, Reloaded

The approach we take to overhaul bug tracking systems builds upon the previous successes of Q&A sites, particularly of Stack Overflow, of discouraging conversational content and promoting informational content.

3.1 Diagnostics and Solutions

To discourage conversational content, we eliminate comments as the main focus of bug reports, and allow only two main types of informational content as input from the user: either *diagnostic* or *solution* posts. Diagnostic and solution posts will be listed following the bug description as the main components of the bug report. Additionally, we will group and present first all diagnostic posts and then all solution posts.

Diagnostic information characterizes the bug and differentiates it from other bugs and is fundamental for developers to reproduce the bug. Such information includes the software version, steps to reproduce, external dependencies, and execution results (stack traces, screen shots, textual descriptions, etc). As for solutions, these are generally abstract or concrete proposals for how to resolve the issue, such as ideas or source code patches. Previous work has shown that diagnostic information and solutions are the information developers seek most from bug reports [9, 4]. This should, thus, improve the cognitive fit between the mental representation of the bug and the bug report.

3.2 Contextual Comments

Although conversations are difficult to read and digest, we believe that they are necessary for the bug resolution process and content creation in general. Wikis are a prime example

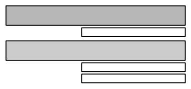

Figure 1: Layout for diagnostic and solution posts and nested comments.

of a collaborative ecosystem for creating content with high archival value. To this end, wikis provide a 'background' channel for collaborators to discuss the content of wiki pages.

To support the discussion of diagnostic information and of solutions, which are crucial for the resolution of bugs [4, 10], we provide the ability for users to post comments that are specific for each diagnostic or solution. Such comments will be nested underneath its respective post and will be presented with secondary importance, as illustrated in Figure 1, which shows two diagnostic or solution posts (grey boxes), each followed by their own comments (white boxes).

3.3 Editing and Improving Content

Although we allow users to discuss diagnostic and solution posts, we consider that users consulting bug reports should not have to read the comments for diagnostics or solutions to find the information they seek, just as a user consulting a wiki page does not have to read its discussion page. To allow diagnostic and solution posts to be updated with the main resolutions and findings of the discussions from comments and continuously improved, we allow users to edit their own posts.

4 Evaluation

To evaluate these ideas, we implement BugBot, a bug tracking systems that implements the ideas from the previous section: diagnostic and solution posts, contextual comments, and editing and improving content.

We set BugBot as the bug tracking system for Clafer, a lightweight modelling language in active development, written in Haskell, that currently has 13k LOC. The bug tracking system has been used actively by the Clafer team since Feb 2012 by a total of 9 users. For this evaluation we will consider its usage from February to August 2012, at which it had a total of 138 bugs, of which 85 (62%) have been resolved and 48 (35%) are feature requests. Of the 9 users, 4 of them are Clafer developers and five are Clafer users. All 9 BugBot users have software development experience. As for training, we simply presentend the users a document outlining its philosophy and intented usage. To understand the advantages and disadvantages of BugBot we use grounded theory, as proposed by Strauss and Corbin [14], triangulating interviews, an online questionnaire, and a qualitative and quantitative analysis of BugBot's usage.[1] Since the number of bug reports is not too large, we have read and analyzed all bug reports.

[1]Questionnaire at `http://gsd.uwaterloo.ca/bug-bot-eval`

4.1 Findings

We now present our findings for each of BugBot's features.

Diagnostics and Solutions Users found that having informational content such as diagnostic and solution posts greatly facilitated locating relevant information and consulting bugs since, differently from conversational comments, each post does not assume a predefined order and is highly independent of other posts.

For the 138 bug reports, there was a total of 206 diagnostic and solution posts, of which 37% were diagnostic and 63% were solutions. Excluding 32 bugs that received zero posts, the mean number of posts per bug is 1.9 and median 2; 26 bugs received three or more posts.

Solution posts were mostly as expected: ideas and proposals for resolving the bug. Users took advantage of the ability to post several different solutions to discuss the benefits and drawbacks of each one. Diagnostic posts were mostly used to present execution results and a comparison with the expected results. Execution results were mostly either error messages, logs, or textual output. Apart from software revision, environment settings were hardly ever mentioned, most likely since Clafer's code is highly portable across operating systems. For feature requests, diagnostic posts were used to motivate the need for the requested improvement. During the first two weeks of usage, however, users were unclear about what information diagnostic posts should contain.

Although users appreciated the benefits of diagnostic and solution posts to improve archival value and facilitate reasoning, it was clear that users needed some time to get familiarized with the new approach. The biggest problem faced by users was deciding if their contribution should be a diagnostic or a solution post, since there are some content for which such distinction is unclear. For example, a workaround to a bug might also be perceived as diagnostic information. Take the following post: "The software crashes when I invoke with the `--debug` flag. It doesn't crash without the flag." One might see that this should be posted as a solution, others will think that this is not a solution, but diagnostic information. We also found that many posts contained both diagnostic information and solutions and could have been split into two diagnostic and solution posts.

Users also found that some content they wished to post were *neither* diagnostic nor a solution, such as discussing who should fix the bug, or what the bug's priority is. For these cases, users used the comments section of the bug description to discuss priority and the comments section of the solution to discuss who will implement the solution. In one particular case, a user posted a solution proposing that the failure the bug report describes is not really a bug. The comments for this solution were then used to discuss this merit.

Contextual Comments Contextual comments are clearly the most appreciated change over traditional bug tracking systems. They are effective in BugBot because each comment thread belongs to a specific diagnostic or solution post and thus has a clear topic. The following comments from two users summarize its advantages:

My favourite feature is the nesting. It makes reading bug reports simpler since all the discussion and replies are grouped together.

It is especially important if there are multiple solutions and we need to discuss each of them. [. . .] It is easier to distinguish more important threads and to disregard those that are either incorrect or insignificant.

Most posts received at least one comment. Of the 206 posts, 44% received no comments, 33% received one comment, 10% received two comments, and 13% received three or more comments. The maximum number of comments for a post was 12, for a contentious bug that required much explanation of why it was not a bug. Solution posts in average receive 50% more comments than diagnostic posts: of the total of 266 comments, 76 were for diagnostic posts, 190 for solution posts.

Comments for diagnostic posts were mostly used to verify or dispute the validity of the diagnostic. Comments for solutions were used mostly to get feedback about the solution: if it is a good idea, if it has been proven to resolve the issue or not, its execution results, and ideas for improving it. Comments for solutions were also often used to disclose the status of solutions: in development, committed, released, etc. A user wanting to understand if a diagnostic post was valid or if a solution has been proven to work, need only read the comments for the post. Finally, comments for solutions were also used for coordination, such as deciding who will implement the solution.

Comments, in general, also allowed users to digress, which is what commonly occurs in typical bug tracking systems. Common examples of this is users discussing interesting, but not crucial, details about solutions, and posting references to external sources of information.

Just as users were sometimes unsure if their contribution should be posted as a diagnostic or solution, users were also unsure if they should contribute with a comment or with a new diagnostic or solution. A user wanting to suggest an improvement to a solution, for example, could either suggest it through a comment or post a new solution that extends the original solution with improvements.

Editing and Improving Content In traditional bug tracking systems, content improvement can only be done by posting additional comments. BugBot users claimed that being able to edit posts increases the archival value of bugs since it does not require additional comments. And most edits were substantial: 50% of them having a *Levenshtein edit distance* of 100 characters or more. 15 posts were edited three or more times.

We found, however, that this feature was not used as much as it could have been: only 35% of diagnostics posts and 27% of solutions were edited. Users are aware of this problem:

Our users do not have enough self discipline and motivation to write the solutions that actually were implemented. Sometimes, the solutions are what was proposed, something slightly different was implemented, but the solutions were not updated.

This could be explained by two findings. First is that users stated that editing existing posts is time consuming, since they have to revise an existing content. Users were also not clear on the benefits of editing content for *all bugs*. They consider it is most beneficial for bugs with higher activity and importance. Second is that users were again unsure, similar to the decision of posting a comment or a new post, if they should either improve a diagnostic or solution by editing it, by posting a new one, or by adding a comment.

Effects on Archival Value and Reasoning Users were convinced that BugBot at least marginally improved archival value of bugs compared to traditional bug tracking systems, with three users affirming that it significantly improved archival value. The features that mostly improved archival value in the perception of users were contextual comments and being able to edit posts.

Users were also convinced that BugBot improved reasoning compared to traditional bug tracking systems; only one user said that it was only marginally better. Users found that diagnostics and solutions and contextual comments were the features that most contributed to facilitate reasoning, since it allows bug report information to be structured hierarchically (via contextual comments) and topically (via diagnostics and solutions) which "*. . . is helpful for building a mental model of the bug report*".

4.2 Discussion

This evaluation finds that users were much satisfied with BugBot. Users clearly perceived that classifying posts as diagnostic or solution, having contextual comments, and editing posts allowed significant improvements to archival value and to facilitating reasoning. All but one user, who was unsure about his preference, affirmed that they would choose BugBot over traditional bug tracking systems. In fact, some users asked for further installations of BugBot for other projects.

Nevertheless, BugBot can be improved. The first issue comes from leaving the responsibility of organization to users: they must decide if they should contribute by commenting or editing an existing post or by posting a new diagnostic or solution. We acknowledge that there is no rule of thumb for when to post something as a diagnostic, solution, comment, or edit. Users will have to decide on this depending on the situation, and there will certainly be disagreements. Given the advantages that it brings, however, we consider that many communities will appreciate the afforded power and learn to deal with the learning curve. A future direction to minimize this problem is to attempt to aid users in submitting their contributions by automatically classifying a user's contribution and placing it appropriately, such as in a list of nested comments.

Users might also find BugBot to be too restrictive, since it only allows users to post diagnostics or solutions. From our evaluation, however, users managed to discuss everything that they needed to discuss. We consider that this perception is a result of the learning curve that BugBot presents to new users, which is natural, since BugBot breaks several paradigms.

> *With BugBot, reading a bug report takes some adjusting to, since the posts are grouped by category instead of arranged chronologically.*

> *The structured entries [. . .] and certain philosophy we could follow allowed us to improve the quality of the bug/new feature reports. [. . .] We could, with some*

discipline [. . .] replicate the same in a traditional bug tracker. It wouldn't be that nice and convenient, however.

Finally, users find editing posts to be time consuming. Since they acknowledge the advantages of keeping posts updated, we argue that this could be enforced or encouraged through moderation. A promising approach for encouragement is Stack Overflow's game mechanisms, which encourages desirable behaviour by rewarding users with administrative privileges and reputation points [11, 10]. Adding game mechanisms could then both improve moderation, as well as help users locate reliable information through collaborative filtering.

Overall, BugBot motivates future work on creating ecosystems optimized in helping large groups of users solve problems as a collective, primarily by organizing the enormous amounts of input they should receive. Bug tracking systems are just one case in which problem-solving is performed collaboratively. The open-source movement, which is expanding to areas other than software development, is a prime driver of this need.

5 Threats to Validity

The main threats of this study relates to the group of users who evaluated BugBot and the project it was used for: Clafer might not be representative of all types of software due to its domain and size; we received responses from only 7 users which might not be representative of the all the population of bug tracking system users. Furthermore, the Clafer team and the first author of this paper work physically close to each other in the same research laboratory, increasing the potetial for biased responses. The Clafer developer group, however, is an independent and productive team with complete authority to decide which bug tracking systems they use. They would not have kept using BugBot if they found it unuseful. They are still using it, plan to continue using it indefinitely, and have asked for further installations for other projects.

6 Conclusion

We implement and evaluate BugBot, a bug tracking system that promotes information instead of conversations, with the intention of improving bug report archival value and facilitate reasoning about bugs. Users find that categorizing information as diagnostic or solutions, allowing for contextualized comments for editing of content offers significant improvement over traditional bug tracking systems, even considering the paradigm shift and learning curve.

References

[1] Ankolekar, A., Sycara, K., Herbsleb, J., Kraut, R., and Welty, C. Supporting online problem-solving communities with the semantic web, 2006.

[2] Bettenburg, N., Just, S., Schrter, A., Weiss, C., and R. What makes a good bug report? *FSE* (2008).

[3] Boehm, B., and Basili, V. R. Software defect reduction top 10 list. *IEEE Computer* (2001).

[4] Breu, S., Premraj, R., Sillito, J., and Zimmermann, T. Information needs in bug reports: improving cooperation between developers and users. *Computer Supported Cooperative Work* (2010).

[5] Green, T., and Petre, M. Usability analysis of visual programming environments: A cognitive dimensions framework. *Journal of visual languages and computing* (1996).

[6] Harper, F. M., Moy, D., and Konstan, J. A. Facts or friends ? distinguishing informational and conversational questions in social Q&A sites. *CHI* (2009).

[7] Ko, A., and Chilana, P. Design, discussion, and dissent in open bug reports. *iConference* (2011).

[8] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM* (1997).

[9] Lotufo, R., Malik, Z., and Czarnecki, K. Modelling the 'hurried' bug report reading process for bug report summarization. *ICSM* (2012).

[10] Lotufo, R., Passos, L., and Czarnecki, K. Towards improving bug tracking systems with game mechanisms. *MSR* (2012).

[11] Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G., and Hartmann, B. Design lessons from the fastest q&a site in the west. *CHI* (2011).

[12] Rastkar, S., and Murphy, G. Summarizing software artifacts: a case study of bug reports. *ICSE* (2010).

[13] Spolsky, J. Building communities with software. *http://www.joelonsoftware.com/articles/BuildingCommunitieswithSo.html* (2003).

[14] Strauss, A., and Corbin, J. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage Publications, 2008.