

# Improving the Readability of Defect Reports

Bogdan Dit  
Wayne State University  
Department of Computer Science  
Detroit Michigan 48202  
+1 313 577 0565  
bdit@wayne.edu

Andrian Marcus  
Wayne State University  
Department of Computer Science  
Detroit Michigan 48202  
+1 313 577 5408  
amarcus@wayne.edu

## ABSTRACT

Bug reports and the user comments associated with them are an important source of information for developers. When many comments are posted in response to a bug description, the discussion becomes difficult to understand. In order to keep discussions coherent and easy to understand, comments should specify if they contain responses to previously posted comments and should not refer to more than one topic.

We propose a system, which recommends developers a set of comments related to their own comment, such that developers can explicitly make connections between the new comment and existing ones. The goal is to improve the readability of the discussion thread.

## Categories and Subject Descriptors

D.2.7 [Software Engineering] Distribution, Maintenance, and Enhancement.

## General Terms

Documentation, Human Factors,

## Keywords

Defect reports, Comprehension, Language analysis

## 1. INTRODUCTION

Defect tracking systems, such as Bugzilla, are extremely useful in large projects, which involve many, often geographically distributed developers. These systems are used not only for locating and fixing the bugs, but also for detecting bug duplicates, triaging incoming bugs, automatically assigning bugs to developers, etc. Given the importance of the textual information in the bug reports, it is desirable that this text is highly coherent, such that the readers can easily understand it.

Usually, each project defines its own guidelines on how to post comments in the bug reports. However, neither the Bugzilla documentation<sup>1</sup> nor the Bugzilla etiquette for the Mozilla project<sup>2</sup> do not specify any elements of discourse or style that the comments should follow. Previous work [4], which studied how

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

RSSE '08, November 10, Atlanta, Georgia, USA.  
Copyright 2008 ACM 978-1-60558-228-3 ...\$5.00.

developers describe bugs, did not address these issues as well.

We argue that developers should follow some well defined rules of discourse when posting comments. For example, it is very common for developers to misuse demonstratives (i.e., *this*, *that*, *these*, *those*) or other deictic words. The burden to connect these words to the proper context rests solely on the shoulders of the reader. Developers may refer to “*this*” in a comment, where “*this*” may be substituted with “*bug*”, or some other issue mentioned in a previous comment. If previously posted and related comments are not explicitly pointed out, it may require quite a bit of effort for the developer to understand such comments.

One easy way to set the proper context for a comment is to post it as a reply to the appropriate previous comment. The text of the replied comment is usually automatically included in the new comment. Alternatively, developers can include links to other comments in the new comment that they are posting. Another way to preserve the proper context and discourse is to avoid commenting on separate issues in one posting. When developers fail to follow these simple rules, the readability and understandability of their comments and ultimately of the entire discussion thread suffers. We propose a system, which detects such problems associated with new comments and suggests solutions to the developers.

### 1.1 An Example from Mozilla

Bugzilla already offers the possibility to reply to individual comments by clicking the “*Reply*” button associated with each comment (see Figure 1).

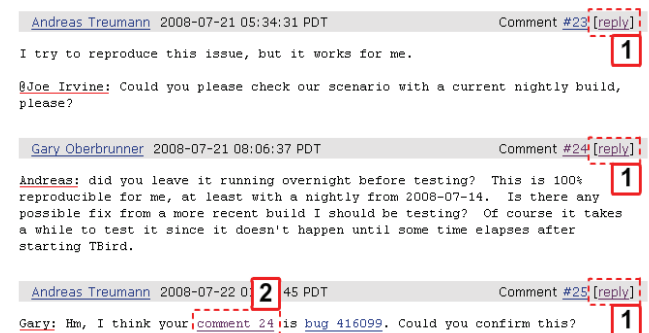


Figure 1. Three comments from Mozilla bug #412950.

[1] Reply button for each individual comment.

[2] Example of an explicit link to another comment.

<sup>1</sup> <http://www.bugzilla.org/docs/tip/en/html/hintsandtips.html#commenting>

<sup>2</sup> <https://bugzilla.mozilla.org/page.cgi?id=etiquette.html>

Developers often fail to hit the “Reply” button especially when they are responding to the latest posted comment. While this is not cause for major concern, it is much worse when they respond to a comment posted much earlier.

Figure 1 shows comments #23, #24, and #25 from bug # 412950<sup>3</sup> in the Mozilla project. This is still an open bug as we write this paper. Developers refer here to previous comments through other developers’ names and links, rather than replying to them.

Some cases are even worse. Figure 2 shows an example of another comment from bug #412950 in Mozilla, where the developer makes references to two previous comments, identified via the name of other developers. Such comments may also negatively affect comprehension.

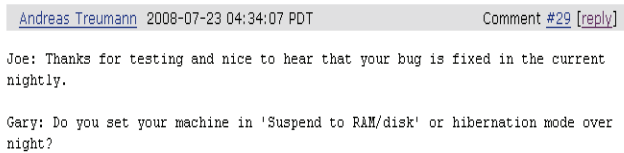


Figure 2. Comment #29 from Mozilla bug #412950.

Although bug #412950 is a recent one, still open as of September 18, 2008, with only 33 comments (many bugs have well over 100 comments) posted by 12 developers, the discussion is already difficult to follow. Figure 3 shows how comments relate to each other in this bug. Each node corresponds to a comment. The node on the top left hand side of the figure corresponds to the bug description. A black solid arrow from node X to node Y indicates that comment #X is a reply to comment #Y (i.e., the “Reply” button (see Figure 1) was used or an explicit link (see Figure 1) is contained in the comment). A red dotted arrow from node Z to

node W indicates that comment #Z refers to comment #W; however, the “Reply” button associated with comment #W was not used, and also comment #Z does not contain any explicit links, such as those shown in Figure 1. There are three blue nodes in Figure 3 corresponding to comments #9, #12, and #13, which report bug duplicates.

The figure reveals the following about the discussion. There are only 11 black solid arrows, while there are 24 red dotted ones (i.e. there more implicit references than explicit references, which may indicate comprehension difficulties while reading the comments). Three comments (i.e., #16, #29, and #30) refer to more than one other comment at the same time (i.e., they have more than one outgoing edge). Ideally, most links should be black and the only red ones should be towards the immediately preceding comment.

## 2. RECOMMENDATIONS FOR COMMENTS IMPROVEMENT

Our proposed recommender attempts to improve situations like the one described above, where non-adjacent comments do not reference each other explicitly or when comments reference more than one other comment. The recommender is envisioned to work in two situations: when new comments are written and when old comments are read.

As a developer writes a comment, the recommender scans the comment and if no links are included and if the comment is not an explicit “reply”. Then the recommender suggests a ranked list of previous comments identified as related to this one. The developer has the option to choose one of them or ignore the recommendations. If a comment is chosen by the developer, an explicit link will be automatically included in the new comment.

The proposed recommender would also work on previously written comments, while these are read by developers. The recommender system will list next to each existing comment a

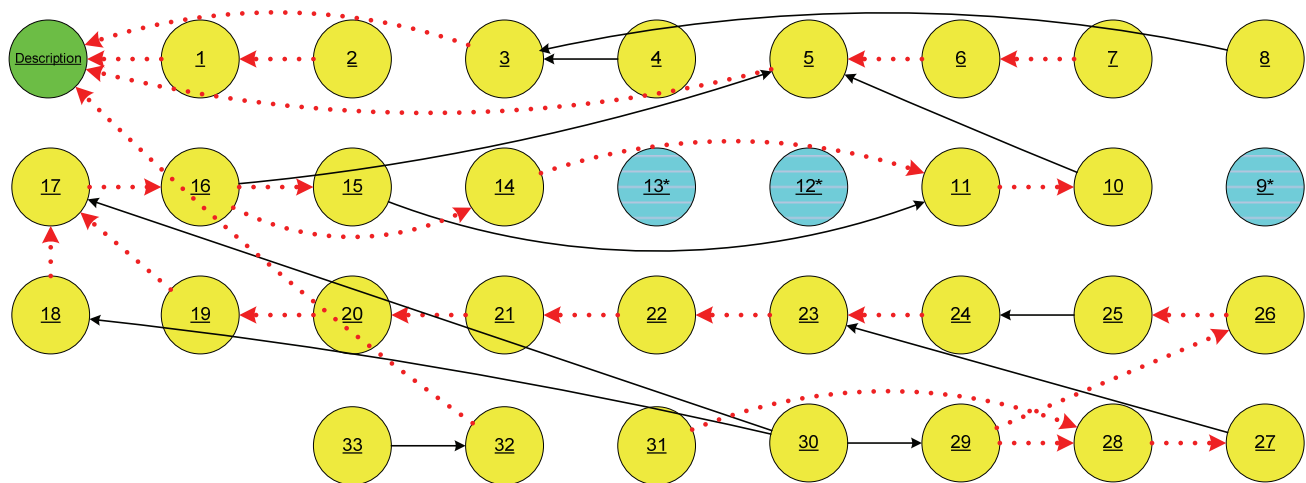


Figure 3. The discussion in bug report #412950 from Mozilla (as of September 18, 2008). Each node corresponds to a comment. Black solid arrows indicate explicit reference to a previous comment (via a reply). Red dotted arrows indicate implicit references (via context). Blue nodes correspond to duplicate messages.

<sup>3</sup> [https://bugzilla.mozilla.org/show\\_bug.cgi?id=412950](https://bugzilla.mozilla.org/show_bug.cgi?id=412950)

short list of related comments in a "see also" fashion.

The recommender will be based on the use of Information Retrieval (IR) methods combined with Natural Language Processing (NLP) and discourse analysis techniques to implement this recommender system. IR methods have been used before to index bug reports and comment in order assess similarities between bug reports to support triage [1], impact analysis [2], etc.

We used IR methods in a similar fashion in previous work on indexing bug reports [3]. A corpus is built using the bug description and comments. Once a developer writes a new comment, similarities between the new comment and all the previous comments in the same bug (including the description) are computed. The comments are ranked based on the similarities and the top ones are recommended to the user such that he can include explicit references to them if needed. In addition, the new comments are scanned for the names of developers and if such names are encountered, the system recommends replacing them with explicit references to the latest comments posted by these developers. Finally, when more than one developer is mentioned in the new comment or more than one explicit link exists, the systems recommends splitting the comment into multiple pieces, posted separately, and referenced appropriately.

The recommender will need a calibration system, which will adjust from system to system. This will depend on the developers and the domain of the system.

The challenging part of the recommender development is the reference resolution part. This is a notoriously difficult problem in NLP and discourse analysis. One thing differentiates this work from a more common application of NLP. The context of each discussion is quite narrow. For example, when developers use words like "this" or "that", there are only a few items they can really refer to. Deciding among these is simpler than in more complex discourse settings.

Another usage we envision for our recommender is to verify whether posted comments follow certain discourse rules. Project managers may want define their preferred rules and make sure developers follow them. This application extends beyond bug description, to the generation of many other natural language

artifacts. We are designing a specification language for discourse rules and the recommender will check new comments against such rules. Each project can develop and express their rules. For example, the misuse of demonstratives can be detected. In addition, the system will learn from older comments posted by developers and check if the developers perpetuate old habits. For example, if an author consistently posted comments in the past without using "Reply", the recommender will see if this behavior is corrected.

Upon implementation, an extensive empirical validation of the recommender is planned. The key factors to be evaluated are usability and usefulness, in addition to the accuracy of the recommendations.

### 3. ACKNOWLEDGMENTS

This work is supported in part by grants from the US National Science Foundation (CCF-0438970 and CCF-0820133). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

### 4. REFERENCES

- [1] Anvik, J., Hiew, L., and Murphy, G. C., "Who should fix this bug?" in Proceedings 28th International Conference on Software Engineering (ICSE'06), 2006, pp. 361-370.
- [2] Canfora, G. and Cerulo, L., "Impact Analysis by Mining Software and Change Request Repositories", in Proceedings 11th IEEE International Symposium on Software Metrics (METRICS'05), September 19-22 2005, pp. 20-29.
- [3] Dit, B., Poshyvanyk, D., and Marcus, A., "Measuring the Semantic Similarity of Comments in Bug Reports", in Proceedings 1st International Workshop on Semantic Technologies in System Maintenance (STSM'08), 2008
- [4] Ko, A. J., Myers, B. A., and Chau, D. H., "A Linguistic Analysis of How People Describe Software Problems in Bug Reports", in Proceedings IEEE Conference on Visual Language and Human-Centric Computing (VL/HCC), 2006, pp. 127-134.