A Study of Bug Management Using the Stack Exchange Question and Answering Platform

Aaditya Bhatia, Shaowei Wang, Muhammad Asaduzzaman, and Ahmed E. Hassan

Abstract—Traditional bug management systems, like Bugzilla, are widely used in open source and commercial projects. Stack Exchange uses its online question and answer (Q&A) platform to collect and manage bugs, which brings several new unique features that are not offered in traditional bug management systems. Users can edit bug reports, use different communication channels, and vote on bug reports, answers, and their associated comments. Understanding how these features manage bug reports can provide insights to the designers of traditional bug management systems, like whether a feature should be introduced? and how would users leverage such a feature? We performed a large-scale analysis of 19,151 bug reports of the bug management system of Stack Exchange and studied the in-place editing, the answering and commenting, and the voting features. We find that: 1) The three features are used actively. 2) 57% of the edits improved the quality of bug reports. 3) Commenting provides a channel for discussing bug-related information, while answering offers a channel for explaining the causes of a bug and bug-fix information. 4) Downvotes are made due to the disagreement of the reported "bug" being a real bug and the low quality of bug reports. Based on our findings, we provide suggestions for traditional bug management systems.

Index Terms—Bug management systems, Stack Exchange, Question & answer platform

1 Introduction

Collecting and managing bug reports is a crucial part of software development due to the prevalence of bugs. As a result, a number of traditional bug management systems are commonly used in practice. For instance, Bugzilla[1], which is a popular online bug tracking system, is used by many open source projects like Apache[2], Linux[3], as well as a significant number of private organizations, such as IBM[4].

In traditional bug management systems, bugs are reported, discussed, and assigned to developers for fixing. However, there are several challenges associated with bug management [20]. For instance, in traditional bug management systems, like Bugzilla, changes to a bug report from the community (e.g., adding new information, making clarifications, and editing incorrect information) are usually made through sequential commenting, which can lead to very long comment threads. Finding useful information in such a long comment thread is a challenging task [2].

Unlike traditional bug management systems, the bug management process of Stack Exchange is performed on its Q&A platform, namely, Meta Stack Exchange[5]. The use of its platform for bug management results in several unique aspects in the bug management process. For example, Stack Exchange allows users to edit a bug report instead of performing sequential commenting (i.e., in-place editing feature). It also allows users to discuss bugs through two different communication channels, namely, comments and answers (i.e., answering and commenting features). Therefore, it is important to study how such unique features are used in Stack Exchange in the form of Q&A style. Understanding the use of such features can provide insights to the designers of traditional bug management systems, like Bugzilla, who might wish to integrate such features into their systems.

In this study, we performed a large-scale analysis of the bug management system of Stack Exchange by studying

- *Aaditya Bhatia, Muhammad Asaduzzaman and Ahmed E. Hassan are with Software Analysis and Intelligence Lab (SAIL), School of Computing, Queen's University, Canada.*
  E-mail: {aaditya.bhatia, muhammad.asaduzzaman, ahmed}@cs.queensu.ca
- *Shaowei Wang is with the Department of Computer Science, University of Manitoba, Canada.*
  E-mail: shaowei@cs.umanitoba.ca
- *Shaowei Wang is the corresponding author*

1. https://www.bugzilla.org/
2. https://bz.apache.org/bugzilla/
3. https://bugzilla.kernel.org/
4. https://www.ibm.com/developerworks/library/l-bugzilla/index.html
5. https://meta.stackexchange.com/

19,151 bug reports, including 15,904 answers, 42,050 edits, 68,207 comments on bug reports, and 28,785 comments on answers. We studied three unique features, namely, the in-place editing feature, the answering and commenting features, and the voting feature. We structure our study along the following three research questions:

- **RQ1: Does the in-place editing feature help improve the quality of bug reports on the Stack Exchange bug management system?**
  The in-place editing feature is used steadily over the years. In addition, 57% of the in-place edits improved the quality of bug reports, such as adding or correcting essential bug-related information (e.g., observed behavior and environment information). The ability to rollback edits provides a mechanism to resolve errors that arise during the in-place editing, which is not possible in the sequential commenting thread of traditional bug management systems.
- **RQ2: How do users leverage the answering and commenting features of the Stack Exchange bug management system?**
  In general, commenting is used on 76% of the bug reports, while answering is used on 60% of the bug reports. The usages of commenting and answering differ from each other. Commenting provides a channel for discussing bug-related information, whereas answering provides a channel for including the causes of a bug and bug-fix information.
- **RQ3: How do users leverage the voting feature of the Stack Exchange bug management system?**
  Upvoting is used more frequently on bug reports and answers than comments. The use of downvoting has increased gradually over the years, with more downvotes on bug reports than on their associated answers. Most of the downvotes on bug reports are made due to disagreement about whether the reported "bug" is a real bug and the low quality of bug reports (i.e., reported bugs are incorrect, insignificant, incomplete and non-reproducible).

Based on our findings, we provide insights to the designers of the traditional bug management systems, like Bugzilla, who might wish to add these features into their systems. For instance, we suggest that traditional bug management systems should consider introducing the in-place editing feature. Traditional bug management systems also should consider introducing the answering and commenting features to encourage its users to better structure their contribution. Separating the content into different channels can help to find the target information more easily.

Section 2 explains the bug reporting process using the Stack Exchange Q&A platform and its unique features (e.g., the in-place editing feature). While Section 3 discusses our research questions and our data collection process. Section 4 presents the motivation, approach, findings of our three research questions. Section 5 discusses the implications of our findings and Section 6 describes threats to the validity of our observations. We discuss the related work in Section 7. Finally, Section 8 concludes our paper.

2 Background

Stack Exchange[6] is a network of 173 websites, which provides a Q&A platform for its users to share knowledge across various domains (e.g., programming, statistics, and mathematics). One unique feature of the network is that it uses one of its Q&A website, Meta Stack Exchange[7], to manage its bugs. In this section, we introduce the bug management on Stack Exchange along the following two aspects: bug reporting at Stack Exchange, and community contributions to those bug reports. In the latter part, we describe how community users edit, answer, comment, and vote on bug reports. We summarize the differences between bug reports on Stack exchange and those on three traditional bug management systems (see table 2).

2.1 Bug Reporting at Stack Exchange

Users are encouraged to report bugs about all the Stack Exchange Q&A websites on the Meta Stack Exchange. Bug reports are treated as a regular question in the Meta Stack Exchange, and are managed through the same Q&A platform. All bug reports are labeled with the "bug" tag. Thus, we consider all questions with the "bug" tag as bug reports for our study. From here onwards, we refer a Meta Stack Exchange question with the tag "bug" as a bug report if not otherwise mentioned.

Stack Exchange expects its users to provide a title, body, and tags for each reported bug. There is a limit of 150 and 30,000 characters for the title and the body of a bug report, respectively. The title briefly describes the reported bug and the body provides a detailed description of the bug. Besides the tag "bug", users are also encouraged to label a bug report with other tags to better organize the bugs. All bug reports are open to the whole community to view, edit, vote on, leave comments, and provide answers. Figure 1 shows an example of a bug report. The bug is about an issue of tag display. Besides being tagged with "bug", the bug report is also tagged with "stackapps", indicating that the questions specific to stackapps.com are affected by the bug. The bug report also includes the reporter information, enabling any member of the community to visit the profile of the reporter.

On Stack Exchange, moderators are users with special privileges. Such moderators maintain the website content and guide other users in various community activities. Moderators have the unique ability to change the status of

6. https://stackexchange.com/sites
7. https://meta.stackexchange.com/
8. https://meta.stackexchange.com/questions/280304/
9. https://meta.stackexchange.com/posts/280333

Fig. 1: An example of a reported bug on the Meta Stack Exchange website [8].

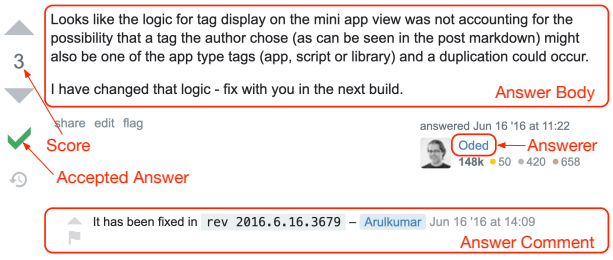


Fig. 2: An example of a developer's response in the form of an answer to the bug report shown in Figure 1[9]. The developer explained the cause of the bug and indicated that the bug had been fixed.

a bug report to indicate its fixing status. There are seven different "status" tags that can be applied to a bug report. Table 1 presents a detailed description for each of these "status" tags. For example, the status of the bug report, as shown in Figure 1, is "status-completed", indicating that the bug has been fixed and the fix has been deployed.

2.2 Community Contributions to Bug Reports

The Stack Exchange community is encouraged to make contributions to a bug report through the posting of answers and comments on bug reports as well as their associated answers. Users can perform the following actions to participate in the management of bugs on Stack Exchange:

Status tag	Description
status-completed	The bug has been fixed and deployed.
status-declined	The bug will not be fixed.
status-bydesign	The bug report refers to a feature that is misunderstood to be a bug.
status-norepo	The site developers were not able to replicate the reported bug.
status-deferred	The bug is intended to be fixed, but not in the near future.
status-planned	The bug is intended to be fixed, ideally in the near future.
status-review	The report contains merit to consider, but requires further investigation. A decision on its decline or approval requires additional investigation.
status-reproduced	Indicates that the site developers were able to replicate the buggy behavior, but are not yet addressing the cause at this time.

TABLE 1: Different categories of status tags and their description.

In-place editing: Users can edit a bug report to fix grammar and spelling mistakes; clarify the description of the bug (without changing its original meaning); add additional information (i.e., related links)[10]. We refer to the feature which enables users to directly edit a bug report as the *in-place editing feature*. In-place editing can be performed by all users of the community. However, edits from users with less than 2,000 reputation points must go through a review process. The edit needs to be approved by the owner of the bug report or moderators before being applied. For example, Figure 3 shows the edit history of a bug report. A developer edited the associated tags of the bug report, and performed grammatical changes in the third edit.

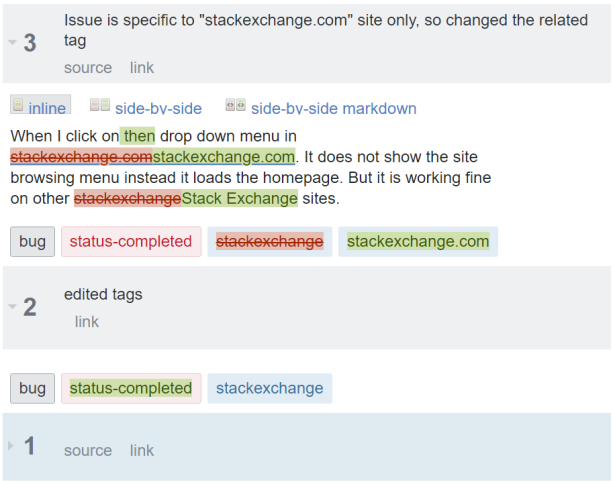


Fig. 3: The edit history of a bug report[11]. Note that the first version is the original bug report.

10. https://meta.stackexchange.com/help/editing
11. https://meta.stackexchange.com/posts/299290/revisions

Characteristics	Stack Exchange	Bugzilla	GitHub ITS[12]	Jira[13]
Voting mechanism	1) Upvoting can be done on bug reports, comments, and answers. 2) Downvoting is only available on bug reports and answers. No restrictions on upvoting.	1) Upvoting can only be done on bug reports. Denotes the community interest in having that particular bug resolved. 2) No mechanism to show that users are not interested in having a bug resolved via downvotes. 3) The owner will be notified when her bug report is upvoted.	Voting is not supported.	Only upvoting is supported. However, project administrators need to enable voting on issues as it is not enabled by default.
Augmentation of information	It supports: 1) answering/ commenting on bug reports; 2)commenting on answers of bug reports.	It supports sequential comments.	It supports sequential comments.	It supports sequential comments.
In-place editing	1) It supports in-place editing and shows the history of changes. 2) All users can make or suggest changes. 3) In place editing can be done for bug reports as well as their answers.	1) It does not support in-place editing. 2) Any changes appear as a sequential comment.	1) Only the bug reporter can perform in-place editing. 2) Other users cannot make or suggest any changes.	1) It supports in-place editing for only those project members who have edit permission.
Support for tagging of bug reports	Tagging is supported.	Tagging is not supported.	Tagging is supported.	Tagging is not supported.
Severity	It does not have a clear mechanism.	It can be selected from a configured levels of severity.	It does not have a clear mechanism.	It can be selected from a configured levels of severity.
Priority	It does not have a clear mechanism.	It can be selected from a configured levels of priority.	It does not have a clear mechanism.	It can be selected from a configured levels of priority.
Closure of the bug report	1)'status-completed' tag is used to indicate the closure of a bug report. 2) The bug report can be changed even after applying the 'status-completed' tag.	1) Issue thread is closed. 2) No further changes can be done to a bug report after its closure.	1) Issue thread is closed. 2) No further changes can be done to a bug report after its closure.	1) Issue thread is closed. 2) No further changes can be done to a,bug report after its closure.

TABLE 2: Comparison between bug reports on Stack Exchange and those on three traditional bug management systems.

Some edits can be automatically performed by the community user (i.e., a bot)[14]. Based on the description on Stack Exchange, the community user is a background process that automates some of the tasks of managing websites. To understand what the bot edits, we manually analyzed 100 revisions that are performed by community bots. We observed that the examined edits were performed for the following four reasons. First, edits were performed to update the protocol part of URLs to access Stack Exchange websites. For example, an edit by the community user replaced http://meta.stackexchange.com with https://meta.stackexchange.com[15]. The second group of edits were performed to add duplicate links[16]. The third group of edits fixed the links for those questions that were migrated from Meta Stack Overflow to Meta Stack Exchange[17]. The last group of edits updated URLs when a website name was changed. For example, an edit replaced http://programmers.stackexchange.com with https://softwareengineering.stackexchange.com because of the change in website name[18]. Such edits are performed by the bot rather than users and they are not directly related to the bug management on Stack Exchange. Therefore, we excluded such edits from our study.

In a traditional bug management system, the original bug report posted by a reporter usually cannot be edited by the community directly. Any changes (e.g., adding new information and correcting wrong information) to the bug report is done through commenting, which can lead to a bug report with a long list of comments, making it difficult for developers to retrieve useful information from such a long thread of comments [2]. In-place editing could be a way to address this problem since **in-place editing enables the content to be located and polished in one place (i.e., within a bug report).** Therefore, in Section 4.1, we investigate how the in-place editing feature is used in the bug management system of Stack Exchange and provide insights to traditional bug management systems who might be considering integrating such a feature into their systems.

13. https://guides.github.com/features/issues/
13. https://www.atlassian.com/software/jira
14. https://meta.stackexchange.com/questions/19738/
15. https://meta.stackexchange.com/posts/37975/revisions
16. https://meta.stackexchange.com/posts/51141/revisions
17. https://meta.stackexchange.com/posts/37975/revisions
18. https://meta.stackexchange.com/posts/243554/revisions

Answering and Commenting: All users are allowed to post answers to bug reports. Figure 2 shows an example of an answer to a bug report. If the reporter is satisfied with an answer that solves her/his question, then she/he can accept the answer by clicking the check mark beside the answer. As shown in Figure 2, the green tick indicates that the answer is accepted by the reporter. Commenting is available for both bug reports (i.e., referred to as a *bug-comment*) and answers (i.e., referred to as an *answer-comment*). Unlike answers, comments are only available to users who have more than 50 reputation points. However, bug reporters and answerers can comment on their own posts without such a restriction. Users can only provide upvotes on comments (in contrast to both upvotes and downvotes on bug reports and their answers). The maximum limit of a comment is 600 characters[19]. We refer to a bug report, all its associated answers, and their all associated comments as a *bug thread*.

Unlike traditional bug management systems, which only allow sequential commenting, the Q&A platform enables its users to contribute to the bug management process through the "answering" of a bug report or through the "commenting" on a bug report or its associated answers. This offers two subtle differences: 1) users have different channels to contribute to discussions about bugs (commenting or answering), and the appropriate channel is not clear from the Q&A oriented user interface (UI); 2) The ability to comment on an answer (*answer-comments*) enables threaded comments. These features are missing in current popular bug management systems, such as Bugzilla. In Section 4.2, we investigate how users leverage such commenting and answering features.

Voting: Stack Exchange allows users to vote on bug reports (i.e., *bug-vote*), answers (i.e., *answer-vote*), bug-comments (i.e., *bug-comment-vote*), and answer-comments (i.e., *answer-comment-vote*). Stack Overflow official documentation defines votes as follows: "votes reflect the perceived usefulness: well-written, well-reasoned, well-researched posts tend to get more attention and more upvotes."[20] An upvote leads to a gain of reputation points for the owner of the post (i.e., questions or answers), whereas a downvote leads to a loss of reputation points for both the owner of the post and the voter. The score of a post (a bug report or an answer) is calculated as follows: number of upvotes - number of downvotes, Unlike questions and answers, comments can only be upvoted and such votes do not lead to a gain or loss of reputation points. In Section 4.3, we investigate how voting is used in a bug thread.

19. https://meta.stackexchange.com/questions/71283
20. https://stackoverflow.com/help/whats-meta

3 Research Questions & Data Collection

3.1 Research Questions

Stack Exchange encourages the whole community to report bug reports through the posting of questions. Unlike traditional bug management systems, Stack Exchange provides three unique features (i.e., in-place editing, answering and commenting, and voting) to encourage the community contributions to bug management (e.g., reporting a bug). The objective of our study is to investigate if these three unique features available on Stack Exchange help the bug management in terms of the quality of bug reports, the organization of contribution under bug reports, and the received feedback from the community. Therefore, we conduct our study to answer the following three RQs:

- **RQ1: Does the in-place editing feature help improve the quality of bug reports on the Stack Exchange bug management system?**
  Motivation: Ensuring the quality of bug reports is crucial for bug fixing [7]. In traditional bug management systems, like Bugzilla, the community is usually not allowed to edit bug reports directly. Any changes to a bug report in terms of adding new information, removing irrelevant information, making clarifications, or editing incorrect information are made through sequential commenting, leading to a long thread of comments for each report. Unlike traditional bug management systems, Stack Exchange incorporates the in-place editing feature in an effort to address the aforementioned difficulties. In this RQ, we examine whether the in-place editing feature helps improve the quality of bug reports. For instance, what is the relationship between bug-fixing time/rate and the number of in-place edits? What is the rationale for such edits (e.g., adding bug-related information or fixing grammatical mistakes) and whether such edits can improve the quality of bug reports? Answering this research question can provide us with insights about the potential benefits of traditional bug management systems supporting the in-place editing feature.
- **RQ2: How do users leverage the answering and commenting features of the Stack Exchange bug management system?**
  Motivation: In traditional bug management systems, sequential commenting is the only medium for discussions – such a single medium for discussions brings challenges for organizing and retrieving the information within such discussions [2]. In contrast, Stack Exchange allows the community to discuss and contribute to a bug report through two different channels, namely, commenting and answering. For example, Stack Exchange recommends users to submit a comment for requesting clarification, or leaving constructive criti-

cism, while to submit a new answer for providing an alternative solution. However, a prior study observes that users do not always follow Stack Exchange's recommendations, e.g., users leverage comments and answers alternatively [36]. The goal of this RQ is to understand how answering and commenting features are used by Stack Exchange users. For example, what do users discuss in answers and comments? Do users discuss different issues in answers versus comments? The answer to this RQ can provide insights to the designers of traditional bug management systems enabling them to make informed decisions on whether to integrate such answering and commenting features into their systems, e.g., introducing different channels to better organize and structure the contributions to bug reports.

- **RQ3: How do users leverage the voting feature of the Stack Exchange bug management system?**
  Motivation: Traditional bug management systems provide only limited support for voting (see Section 2). For instance, in Bugzilla, votes can be given on bug reports, indicating that users want those bugs to be fixed. This is analogous to upvotes in the voting system of Stack Exchange. However, there is no support for downvotes in Bugzilla[21]. Moreover, the gamification of the bug management system of Stack Exchange ensures that such votes are integrated more widely across the system (e.g., not just the report but also on the contributions of other users, like answers and comments). Furthermore, such Stack Exchange votes have some intrinsic value. For example, one has a limited number of downvotes to offer (since a user would lose reputation points for each downvote), ensuring that one would not downvote bug reports arbitrarily and would put some deeper thought in their voting. In this RQ, we aim to provide the designers of traditional bug management systems an empirical understanding of how the voting feature is used in the bug management system of Stack Exchange, enabling them to decide if they should consider integrating such a feature into their systems. For example, we study how the voting feature, typically downvoting (not offered in traditional bug management systems), is used during the management of a bug report.

3.2 Data Collection

This section discusses how we collect the dataset that we used to answer our research questions.

For this study, we download a publicly available data dump of Meta Stack Exchange from archive.org[22]. The data dump contains all the site activities between June 28, 2009, and March 3, 2019. The data dump consists of a collection of XML files containing information about questions, associated answers, post histories, post links, comments, and votes.

A brief description of the data collection process is summarized in Figure 4. To collect the bug reports, we collected all questions with the tag "bug". We then collected the answers that are associated with those bug reports. The comments that are attached to those questions (i.e., bug-comments) and answers (i.e., answer-comments) are also collected using the Comments.xml file. Likewise, we collected all the edits that were performed on those bug reports by leveraging the PostHistory.xml file. We focus only on those edits that were performed on the title, body, and tags of bug reports, including those edits that were rolled back. In addition, we collected all the votes that are associated with those bug reports (i.e., bug-votes), answers (i.e., answer-votes), and comments (i.e., bug-comment-votes and answer-comment-votes). In total, we collected 19,151 bug reports with 15,904 associated answers for our analysis. Table 3 gives an overview of our studied dataset.

We characterize our selected dataset on the following three dimensions: a) the top-20 popular tags that are associated with bug reports other than the tag "bug" and status tags; b) the number of users that are associated with bug reports; c) the number of yearly submitted bug reports. A median of three tags are associated with a bug report (minimum is one and maximum is five). There are a total of 931 tags that are associated with these bug reports. A median of seven bug reports are associated with a tag (minimum is one and maximum is 19,095). The top-20 popular tags and the percentage of the bugs that are associated with these tags are shown in Figure 5. To determine the number of users participating in a bug thread, we consider bug reporters, answerers, bug-commenters and answer-commenters. A median of four users are associated with a bug thread (minimum is one and maximum is 82). The number of bug reports over years is shown in Figure 6. The number increases from year 2009 to 2013 and drops after that.

The dataset we used in this study including our results and scripts are publicly available at the following link to support future replication: https://github.com/SAILResearch/replication-18-Adi-bug_management_SO.

Data	Count
Bug reports	19,151
Answers	15,904
Bug-comments	68,207
Answer-comments	28,785
Bug-votes	137,673
Answer-votes	77,734
Edits (Title, Body and Tag)	42,050
Bug-comment votes	59,605
Answer-comment votes	27,090

TABLE 3: Overview of the studied dataset.

21. https://www.bugzilla.org/docs/4.4/en/html/voting.html
22. https://archive.org/download/stackexchange

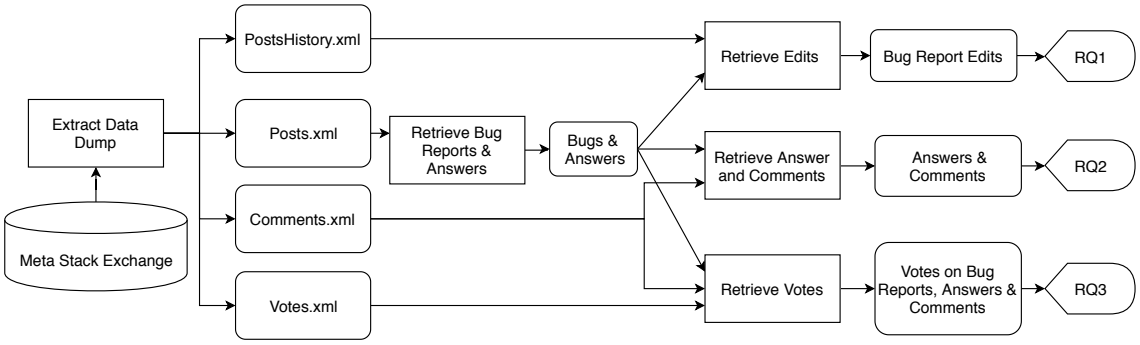


Fig. 4: An overview of the data collection process.

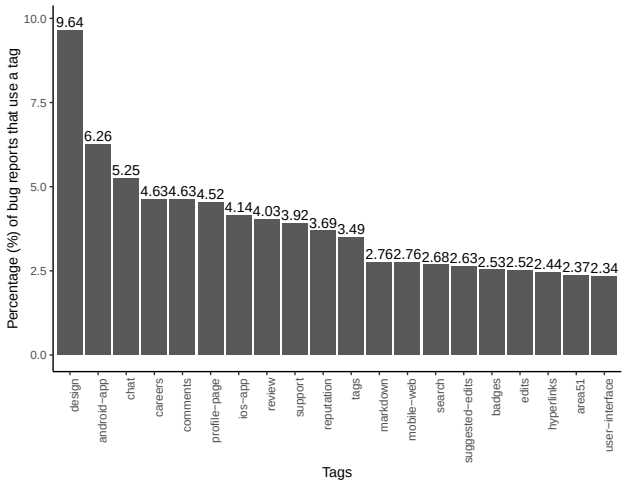


Fig. 5: Percentage of bug reports that are associated with top-20 tags.

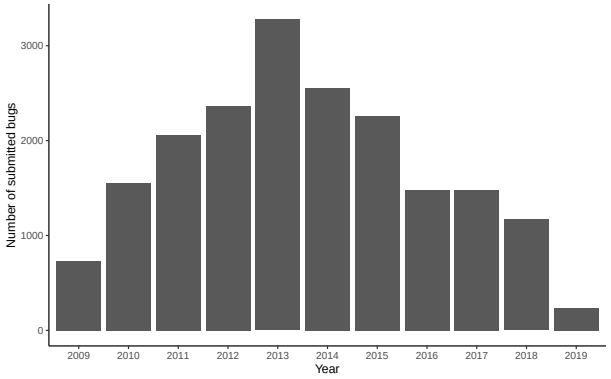


Fig. 6: Number of yearly submitted bug reports.

4 Research Questions and Results

4.1 RQ1: Does the in-place editing feature help improve the quality of bug reports on the Stack Exchange bug management system?

We first conduct a quantitative analysis to understand how often the in-place editing feature is used for bug reports and who perform those edits. We also examine the relationship of the number of edits that bug reports received with their bug-fixing time and bug-fixing rate. Then, we conduct a qualitative analysis to understand the rationale for editing bug reports. The following subsections describe the approach and findings for our quantitative and qualitative analysis.

4.1.1 Quantitative Analysis

Approach: We calculate the total number of bugs that were edited each year to understand how the in-place editing feature has been used over time and visualize the results in a plot. We also check the role of users who are involved in the in-place editing process. We categorize the stakeholders of a bug management system as reporters and non-reporters (i.e., any users other than the reporters.) We are interested in learning if edits that are performed on a bug report help improve the quality of that bug report. We study the relationship between the number of received edits by bug reports before they are fixed and their bug-fixing time and likelihood. Our assumption is that the quality of a bug report improves as it receives more edits, which leads to a higher likelihood of that report getting fixed and a shorter fixing time for that report.

Findings: The in-place editing feature has been used steadily over the years. As shown in Figure 7, the proportion of the reported bugs that were edited over the years is consistently above 80%. In total, 83% of the bug reports were edited through the in-place editing feature.

All stakeholders (i.e., reporters and non-reporters) are editing bug reports. We observed that 43% and 90% of the bug reports were edited by reporters and non-reporters. We also observed 33% of the bug reports that were edited by both reporters and non-reporters. In total, 39% and 61%

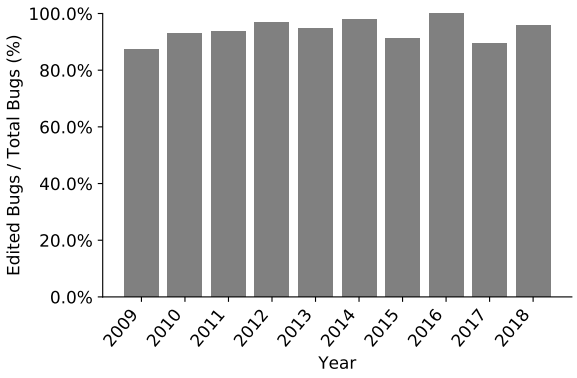


Fig. 7: The proportion of the reported bugs that are edited over the years.

of the edits were performed by reporters and non-reporters, respectively.

In general, the likelihood of bug reports getting fixed is positively associated with the number of edits that were performed before getting fixed. Figure 8 shows the ratio of fixed bugs among different groups of bugs that received different numbers of edits before getting fixed. The ratio of fixed bugs is defined by the total number of fixed bugs with respect to the total number of reported bugs. We observe that the ratio of fixed bugs has a positive association with the number of edits when the number of edits is between 1 and 15. One possible reason is that the received edits produce further information (such as observed behavior, screenshot, and steps to reproduce) on reported bugs that increased the likelihood of a bug getting fixed. The assumption is supported by our qualitative analysis as shown in Section 4.1.2. We also observe that the bug-fix ratio drops when bug reports received more than 15 edits. One possible explanation for such a drop is that the size of the group consisting of bug reports with more than 15 edits (i.e., 315) is much smaller compared to the other groups (i.e., 32,062, 8,493 and 1,059), which may affect the result. In addition, Figure 9 shows the time of getting a bug fixed (in hours) among the different groups of bug reports. We observe that a positive association between the number of edits that were performed on bug reports and their fixing-time when the number of edits increases from 1 to 15. One possible explanation is that bug reports with more edits are more difficult and require more time to fix. Similar to the prior analysis on the ratio of fixed bugs, we also observe a drop for more than 15 edits. We assume it is because of the same reason - the small size of the last group of bug reports. The Wilcoxon rank-sum test shows the difference between bug reports with 1-5 edits and bug reports with 6-10 edits is statistically significant ($p - value < 0.5$), while there is no significant difference between the groups with 6-10 edits and those with 11-15 edits.

4.1.2 Qualitative Analysis

Approach: We perform a qualitative analysis to understand the rationale for the performed edits on the collected bug reports. Similar to prior studies [12], [31], to achieve a confidence level of 95% with a confidence interval of 5%, we randomly sampled 380 edits from all the edits of the studied bug reports (i.e., 22,105 edits in total) and identified the rationale for such edits. Bettenburg et al. investigated the types of bug-related information that are desired by developers when resolving bugs [7]. If a user edits a bug report to add such information, we consider that such an edit helps improve the quality of the bug report. Table 4 (a) provides a brief description of rationales for editing bug reports that help improving the quality of such reports. To examine if the edits improve the quality of bug reports, we reuse the types of information that were defined by Bettenburg et al. to label our studied edits. We also observed several types of Stack Exchange-specific edit rationales that are not defined in Table 4 (a), such as fixing grammar issues and formatting the text. We note that such types of edits are only possible due to the in-place editing feature of Stack Exchange. We performed the following process to derive a list of rationales for edits and labeled the randomly sampled edits. This process involves three phases and is performed by the first two authors (i.e., A1 & A2) of this paper:

- Phase I: A1 started with the rationales for edits defined by Bettenburg et al. [7] and derived a draft list of Stack Exchange-specific rationales for edits based on 50 randomly sampled edits. Then, A1 and A2 used the draft list to label the edits collaboratively. During this phase, the Stack Exchange-specific rationales for edits were revised and refined (the rationales are shown in Table 4 (b)).
- Phase II: A1 and A2 independently applied the resulting rationales from Phase I to label all 380 edits. A1 & A2 took notes regarding the deficiency or ambiguity of the labeling for the rationale(s) of each edit. During this phase, no new rationales were introduced.
- Phase III: A1, A2 discussed the coding results that were obtained in Phase II to resolve any disagreements until a consensus was reached. The inter-rater agreement of this coding process has a Cohen's Kappa of 88.4% (measured before the start of Phase III), which indicates that the agreement level is substantial.

Findings: the majority (57%) of the studied edits improved the quality of their associated bug reports by adding/correcting/clarifying essential bug-related information (e.g., observed behavior and environment information). In general, 59% of such edits enhanced the content of the bug report (i.e., content enhancement edits). The remaining non-content enhancement edits were concerned with grammatical and formatting changes, accounting for

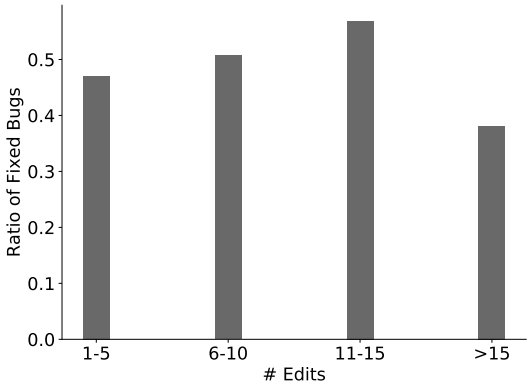


Fig. 8: The ratio of fixed bugs among different groups of bugs that received different number of edits before getting fixed.

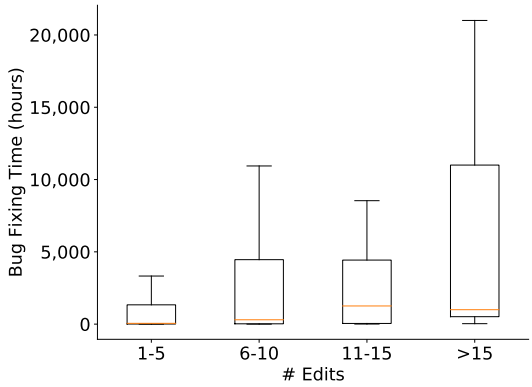


Fig. 9: The time of getting fixed among different groups of bugs that received different number of edits before getting fixed.

(a) Rationale for edits as documented by Bettenburg et al. [7] (*Bug-related rationales*)		
Edit Rationale	**Description (D) - Example (E)**	**Percentage**
Observed behavior	(**D**) Adding a more detailed observation of the bug. (**E** #60189) regarding buggy behavior in tags, an edit added: "Edit: The tag has disappeared from the tags page."	41.3%
Screenshot	(**D**) Addition of a screenshot to visually demonstrate the occurrence of the reported bug. Editors add additional text to demonstrate the screenshot. (**E** #259358) "Please see in this figure a screenshot of the error: .. http://i.stack.imgur.com/PBcTD.png" was added.	26.9%
Version	(**D**) Adding information about the software version on which the reported bug was observed. (**E** #273515) "UI items displayed twice bug in android: Stack Exchange Android App Version 1.0.77" was added.	13.5%
OS	(**D**)Adding information about the operating system on which the reported bug was observed. This also includes browser information. (**E** #273724) "On iOS 9.2.1 Mobile Safari." was added.	9.9%
Severity	(**D**) Stressing on the importance of the reported bug. (**E** #279117) regarding text, "Seriously, this is slightly irritating, I am not just posting this to point out how annoying.." was added.	4.9%
Expected behavior	(**D**) Adding a description of what is expected as a normal/optimal behavior to emphasize that the abnormality caused by reported bug. (**E** #311991) the editor added "But the warning as stated is misleading and could possibly be replaced by a different warning about a poorly phrased title or something similar."	4.5%
Steps to reproduce	(**D**) Adding additional information steps for replicating the bug. (**E** #142696) "Steps followed: *Go to chat site. *Found that I was logged in as account 1 * Where is logout? (bug 1) ... " was added.	4.0%
Stack traces	(**D**) Adding stack traces related to the reported bug. (**E** #263612) on keyboard shortcuts, the original content "the user presses 'Command + L' the 'Insert Hyperlink'" was edited to "the user presses >kbd<Command>/kbd< + >kbd<L>/kbd< the 'Insert Hyperlink'".	2.7%
Hardware	(**D**) Adding information related to hardware in the bug report. (**E** #75805) regarding lag in website: "I just noticed that the bug seems absent on a fourth PC, my old 1.5 GHz laptop with Win XP SP 3 and 500 MB RAM. I have only tried .." was added	1.3%
Summary	(**D**) Adding a summary of the bug report to emphasise the important content. (**E** #296240) editor (who is also the reporter) added "To recap, the bug is: the title on the page doesn't..."	0.9%
(b) Stack Exchange specific in-place edit rationales (*Stack Exchange-specific rationales*)		
Clarification	(**D**) Adding non-bug-related information to further clarify the reported bug. (**E** #299048) editor clarified his original post by replacing the text "My reputation remains unchanged (6)" with "My reputation on stack overflow was 6, at the time of asking the above question ..."	10.8%
Correction	(**D**) Correctting bug-related information, such as the OS information. (**E** #217448) "This happens for me on Ubuntu 13.04 (64-bit)" was edited to "This happens for me on Ubuntu 13.10 (64-bit)".	7.6%
Ad-hoc solution	(**D**) Editing the original bug report to add an ad-hoc solution or workaround. (**E** #194556) with incorrect coloring, editor added "EDIT4: Can be fixed simply by running the JavaScript $('rect').css('stroke-width', '0') or a userstyle rect { stroke: none }. I may make a user[style—script] to fix this."	2.2%
Adding bug-fix information	(**D**) Information about the bug fix or the deployment of a bug fix. (**E** #229833) "I'm fairly sure that this has happened before but it was fixed and I can't find the bug report." was added.	1.8%

TABLE 4: The identified rationales for editing bug reports along with an example for each rationale. Edit rationales that provide bug-related information were previously documented by Bettenburg et al. [7] (however, the description and examples of these rationales are provided by us to highlight them in the context of our study) (a). The Stack Exchange-specific rationales (b) are derived from our manual study. The complete URL for each of the above-mentioned examples is https://meta.stackexchange.com/posts/post-ID, where post-ID (e.g., **E** #259358) is mentioned in each example in the table. The rationales are ordered by their percentage.

37% and 9% of the edits respectively. Note that the sum of rationale percentages exceeds 100%, since in some cases, one edit could have more than one rationale.

We further focus on the edits that are related to content enhancement. A brief description of all content enhancement edits and their percentages are provided in Table 4. We observe that 81% of the content enhancement edits added essential information that is crucial to understand, reproduce, and fix a bug as defined by [7] (i.e., bug-related information). Bettenburg et al. identified incomplete information as the most severe problem for bug reports [7]. Results from our manual analysis suggest that **in-place editing improves the quality of bug reports by adding bug-related information.** We also observe other rationales for edits, which were not observed by [7], to improve the quality of bug reports. These new rationales include making corrections (8%), and providing clarifications (11%). All content enhancement edits, except for *"Ad-hoc solution"* (2%) and *"Adding bug-fix information"* (2%), improved the quality of the bug reports. **In total, 57% (96% out of the 59% content enhancement edits) of the edits improved the quality of bug reports.**

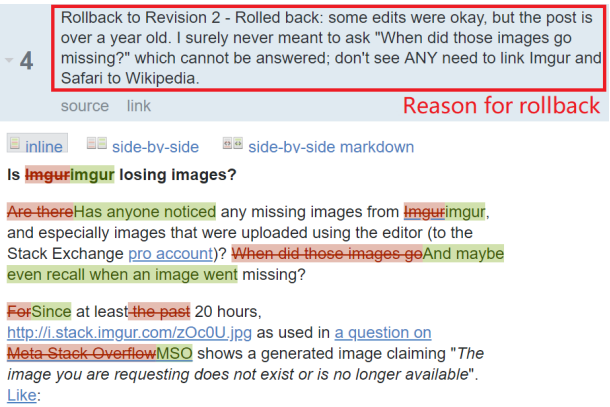


Fig. 10: An example of a rollback on a bug report[23].

In-place edits are sometimes rolled back to resolve errors that arise when multiple users perform in-place editing. To understand the rationale behind such rollbacks, we randomly examined a sample of 50 rollbacks out of the 335 rollbacks in our dataset. From our manual analysis, we identified the following three main rationales for such rollbacks: 1) to correct bug-related information (38%), 2) to undo incorrect grammatical changes (28%), and 3) to undo unnecessary formatting changes (15%), indicating that rollbacks provide a mechanism to resolve errors that arise due to in-place editing. An example of a rollback is shown in Figure 10. The prior edit changed the original meaning of the bug report and the author of the bug report then rolled back the edit.

23. https://meta.stackexchange.com/posts/75105/revisions

	Min	Q1	Median	Q3	Max
Bug-comment	0	1	2	5	52
Answer	0	0	1	1	20
Answer-comment	0	0	0	2	96

TABLE 5: The five-number summary of bug-comments, answers and answer-comments.

Summary of RQ1

In-place editing feature is commonly used. More specially, 57% of the in-place edits improved the quality of bug reports, e.g., adding/correcting/clarifying essential bug-related information (e.g., observed behavior and environment information). In-place edits are sometimes rolled back to resolve errors that arise due to in-place editing.

4.2 RQ2: Is the commenting feature used differently from the answering feature on the Stack Exchange bug management system?

We first conduct a quantitative analysis to understand how frequently answering and commenting features are used in Stack Exchange bug threads. We then conduct a qualitative analysis to understand what types of issues are discussed in bug-comments and answers.

4.2.1 Quantitative Analysis

Approach: We investigate the use of answering and commenting. Hence, we calculate the number of comments and answers that were posted on each bug report. Users are allowed to post comments on bug reports (bug-comments) and answers (answer-comments). Therefore, we also compare the proportion of the comments that were posted on a bug report and those that were posted on its answers after a bug report received a status-completed tag, to investigate if users' interests shift or not.

Findings: In general, commenting is used more frequently than answering significantly. We observed that 76% of the bug reports have at least one bug-comment, whereas 60% of the bug reports have at least one answer. A bug report has a median of two bug-comments, one answer, and zero answer-comment. Table 5 presents the five-number summary of bug-comments, answers, and answer-comments. The Wilcoxon signed-rank sum shows that the difference between bug-comments and answers is statistically significant (p-value < 0.5). In addition, we observe that after receiving a status tag, the number of bug-comments decreases, whereas the number of answer-comments increases. In 56.8% of the bug threads, after receiving a status-completed tag, only answers received comments (answer-comment), while bug reports stop receiving any comment. In other words, once a status tag

was applied to a bug report, users shifted their contributions from commenting on the bug report to commenting on its answers.

4.2.2 Qualitative Analysis

Approach: We perform a qualitative analysis of the discussed issues in bug-comments and answers to understand if users discuss different issues through these two channels. We randomly sampled a statistically representative sample of 380 bug-comments from 68,207 bug-comments and 380 answers from 15,094 answers of bug reports to achieve a 95% confidence level and a confidence interval of 5%. We employed the same process that we performed in RQ1 on these 380 bug-comments and 380 answers. The identified types of the discussed issues in bug-comments and answers are shown in Tables 6 and 7, respectively. Cohen's Kappa values are 89.1% and 80.0% for our tagging of the bug-comments and answers, respectively.

Findings: Users leverage bug-comments to ask for or to provide more bug-related information (e.g., observed buggy behavior and replication confirmation) as well as to clarify the reported bug, whereas answers are commonly used to provide the solution and to explain the cause of a bug. We observe that more than 66% of the bug-comments were posted to report the observed behavior, replication confirmation, related links, or to ask more information from the reporter in an attempt to clarify the reported bug (see Table 6). Interestingly, we also observed in some cases, users integrated the information that was discussed in a bug-comment into the corresponding bug report via the in-place editing feature. For example, in a bug report[24], a user asked the reporter a clarification question and the reporter used the in-place editing feature to add the requested information to clarify the reported bug. In other words, bug-commenting provides a mechanism to ask and collect additional information for a bug report.

Answerers are more likely to report the deployment time for a bug-fix (60%) and explain the cause of a bug (35%) (see Table 7). For example, an answerer mentioned that *"We threw 10,230 errors here (network-wide) due to a web server exhausting memory (due to another, competing application pool being a bully). I posted some details ..."*[25], which indicates the *cause of the bug*. In another example, the answerer mentioned that "The bug happened when submitting a comment while the text cursor was not at the end of the comment..."[26]. The *cause of bug* provides a technical reason for which the bug was manifested, and *reasoning for no-fix* provides a justification for the reason behind *no-fix*. The rationale *reasoning for no-fix* (27%) includes those examples where features of a system were

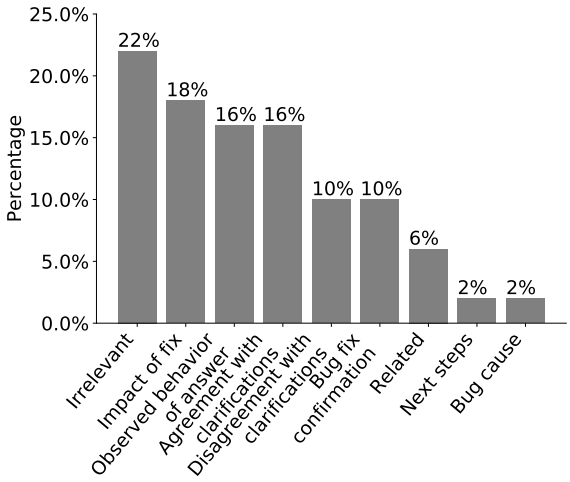


Fig. 11: Qualitative analysis of answer-comments.

incorrectly reported as bugs. Such bugs received "status-declined" or "status-bydesign" tags from the moderators respectively (see Section 2). Interestingly, we observed only a few cases (1%) where users provided screenshots of the bugs in the answers.

We also observe some issues that are discussed in both bug-comments and answers, such as information about the bug-fix deployment, ad-hoc solutions, and bug image information. As to why such issues are discussed in answers and not in bug-comments? We conclude the following possible reasons: 1) Stack Exchange does not provide any guideline for using commenting and answering channels. Thus, it may cause confusion about where to discuss such information for users. For instance, a user asked a question "When to answer vs when to just comment?"[27] on Meta Stack Exchange, which indicates some users are confused about the use of commenting and answering. 2) Comments have a maximum length of 600 characters, which is much shorter than answers. Thus, Stack Exchange users have to use answering channel to provide bug-fix information or solution when the content is beyond the limitation. 3) Answers can help the owner make reputation points, but comments do not. This may motivate the users to post answers instead of comments.

Comments are expected to be used for: 1) requesting clarifications from the question askers or the answerers; 2) providing criticism; and 3) adding minor information[28]. However, we observed that Stack Exchange users leveraged comments for providing Ad-hoc solutions (4.2%) and bug-fix deployment information (5.3%), which are recommended to post through an answer. We also observed that both answer comments and answers were used to explain why bugs occurred. This perhaps because Stack Exchange

24. https://meta.stackexchange.com/questions/233775
25. https://meta.stackexchange.com/questions/296438
26. https://meta.stackexchange.com/questions/222292
27. https://meta.stackexchange.com/questions/
28. https://meta.stackexchange.com/questions/19756

Type	Description(D) - Example (E)	Percentage
Observed buggy behavior	(**D**) Providing more information about the buggy behavior. (**E** #254351) the commenter added the following information: "Note it seems to randomly affect Question edits now. Question posting does not seem affected but ..."	23.2%
Replication confirmation	(**D**) Confirming the reproducability of a reported bug. (**E** #66417) a commenter provided replication confirmation by including the following comment: "yup it is reproducible."	20.0%
Irrelevant	(**D**) Providing any comments that is not related to bug-fix or bug-related information. (**E** #26740) a commenter mentioned: "This is the right place to report bugs and support questions. '	12.1%
Asking more information about bug	(**D**) Asking more information about the buggy behavior. (**E** #196930) the following question was asked in a comment: "Do you have a link to the question?"	11.6%
Links related to the bug	(**D**) Adding possible duplicates of the bug report or related links relevant to the reported bug. (**E** #112096) a link related to the bug report was added in a comment: "Sort of related: meta.stackexchange.com/questions/105030/"	10.5%
Expected behavior of the system	(**D**) Providing what is expected (non-buggy) behavior of the system. (**E** #141867) a commenter mentioned "This doesn't look like it's behaving as it should, ... the suggested edit should be recorded as improved, not rejected. "	6.8%
Clarification of system design	(**D**) Providing clarifications about how the system works. (**E** #209229) a commenter clarified the problem by adding the following comment: "This isn't really a bug. This is typical, as double clicking something will highlight it. In this case, it's just highlighted oddly due to the positioning of the element."	5.8%
Bug-fix deployment information	(**D**) Indicating when a bug-fix will be deployed. (**E** #83661) the expected time to fix the bug was reported in the comment: 'fixed to be deployed some time tomorrow"	5.3%
Ad-hoc solution	(**D**) Providing an unofficial fix or ad-hoc solution to the reported bug. (**E** #19622) an unofficial bug-fix was provided by the following comment: "temporary solution: meta.stackexchange.com/questions/27702/.."	4.2%
Bug image link	(**D**) Providing a URL for an image to illustrate the reported bug. (**E** #163475) a URL of an image was added in the comment: "imgur.com/aL6zn it's white."	0.8%

TABLE 6: The types of discussions identified in bug-comments. The types are ordered by their percentage.

Type	Description (D)- Example (E)	Percentage
Bug-fix deployment information	(**D**) Same as Bug-fix deployment information in Table 6. (**E** #226249) the following answer was provided: "You should see this changed with our next build today. Thanks for catching it."	59.7%
Cause of bug	(**D**) Explaining why a bug occurred. (**E** #250507) an answerer replied: "... The problem was that the web has a few permutations of MathJax: Most sites."	34.7%
Reasoning for no fix	(**D**) Clarifying about how the system should work, and why the bug will not be fixed/a fix is not required. (**E** #175684) an answerer replied: "We will not fix this. The issue is in a specific combination of OS and browser, and ..."	27.4%
Ad-hoc solution	(**D**) Same as Ad-hoc solution in Table 6. (**E** #27494) an answerer suggested a possible solution: "As others have mentioned, try in a different browser to see if that helps as well."	4.5%
Replication confirmation	(**D**) Same as Replication confirmation in Table 6. (**E** #101453) an answerer confirmed the replication of the bug: "I was able to reproduce it in Safari on Snow Leopard."	3.4%
Bug image	(**D**) Providing images to demonstrate the bug (and not the answer). (**E** #267214) a screenshot was added to explain the bug: "Here's a screenshot of the one I currently can't pass ... https://i.stack.imgur.com/EY8nE.png"	1.3%

TABLE 7: The types of discussions identified in answers. The types are ordered by their percentage.

users do not use the answer channel unless they think their contribution is significant.

Furthermore, we observe that once the status-completed (i.e., indicating a bug is resolved) tag was applied to a bug report, users tend to shift their contributions from commenting on the bug report to commenting on its answers (as the platform permits users to comment on the bug reports or answers). We performed a qualitative analysis of the content of answer-comments by randomly sampling 50 answer-comments that were posted after the status tag was added to a bug report.

Users are more likely to discuss the impact of the bug-fix (or no-fix) in answer-comments compared to bug-comments. We observe that a number of answer-comments (60%) discuss the impact of the bug-fix or no-fix (18%), observed behavior of the bug-fix (16%), and discuss agreement (16%)/disagreement (10%) regarding a no bug-fix decision. The types of discussions identified in answer-comments are provided in Figure 11. For example, in a bug report[29] regarding syntax highlighting in Pascal, the answer explained how the bug-fix integrated the required libraries. A user concerned about the impact of the bug-fix: "Could this cause code-comments to break in Delphi?". Commentary about the solution also involved confirmation for the working of bug-fix (10%) and additional information related to the answer (6%). We also observe few cases where subsequent actions after the bug-fix(i.e., "Next steps") and the cause (i.e., "Bug cause") of the bug were reported.

In summary, bug-comments and answers serve different purposes for discussions regarding the bug and its fix or no-fix decision and users tend to shift their contribution from bug reports to answers after a bug is fixed. The

29. https://meta.stackexchange.com/questions/171666

answering and commenting features probably provide a solution for the long sequential comments by structuring the contribution of community in a better way. However, clear guidelines regarding where to contribute (e.g., answers or comments) needs to be provided since users may be confused sometimes.

Summary of RQ2

In general, commenting is used more frequently than answering and they are used for different purposes. Bug-comments provide a channel for asking and providing more bug-related information, whereas answering provides a channel for including the causes of a bug and bug-fix information.

4.3 RQ3: How do users leverage the voting feature of the Stack Exchange bug management system?

4.3.1 Quantitative Analysis

Approach: We perform a quantitative analysis to understand how the use of voting differs across different components of a bug thread. We consider votes on bug reports, answers, bug-comments, and answer-comments as bug-votes, answer-votes, bug-comment-votes, and answer-comment-votes, respectively.

First, we calculate the proportion of each component of a bug thread that has at least one upvote or downvote over the years. To show the differences between upvoting and downvoting, we divide our analysis into the use of the upvoting and downvoting, respectively.

Second, we calculate the ratio of votes that were received by different components with respect to the total number of votes received across all the components of a bug thread. For example, the ratio of answer-votes is measured as the total number of received votes on all answers of a bug report divided by the total number of votes received in all components of a bug thread.

Findings: More than 80% of the bug reports and answers receive upvotes consistently over the years. The use of upvoting on bug-comments and answer-comments is less frequent than bug reports and answers significantly. Figure 12 shows the use of the upvoting and downvoting on the different components of a bug thread (i.e., bug reports, answers, bug-comments, and answer-comments) over the years, in terms of the proportion of received votes across all the components. As mentioned in Section 2, both upvotes and downvotes can be given on bug reports and answers, however, comments (i.e., bug-comments and answer-comments) can only receive upvotes. Compared with upvotes on bug reports and answers (more than 80%), a smaller proportion of bug-comments and answer-comments (less than 40%) received upvotes.

In addition, if we look at the proportion of votes that were received by each component of a bug thread, the bug report received the largest proportion of votes (median proportion is 57%), while bug-comments (median proportion is 5%) and answer-comments received the least amount of votes (median proportion is 0%) (see Figure 13). The wilcoxon signed-rank shows that the differences of proportion of votes between each component of a bug report are all statistically significant (p-value < 0.5).

The use of the downvotes has increased gradually over the years, with more downvotes on bug reports than on answers (see Figure 12 (b)). In recent years, the significant increase of downvotes is surprising because users lose reputation points when they downvote. One possible explanation of this phenomenon is that over the years, users have earned more reputation points to spare. Another possible reason is that users have become more critical about bug reports, thus are more likely to be more vocal and to express their opinions through the downvoting.

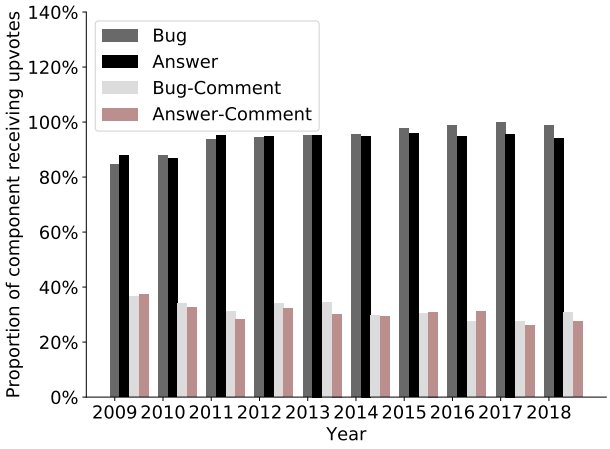
4.3.2 Qualitative Analysis

Approach: We observed that some users left the rationale for downvoting in their bug-comment. Therefore, we perform a qualitative analysis to understand the rationale for the downvoting of bug reports and answers. First, we find all the bug reports that satisfy the following criteria:

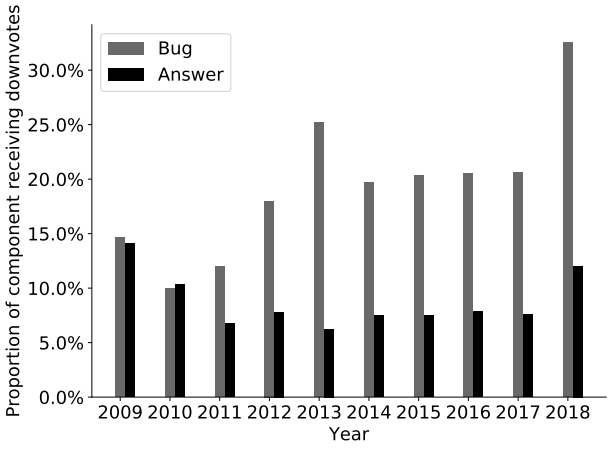
1) Reports having one of the following keywords “downvote”, “down-vote” or “down vote” in their associated comments. Please note that Stack Exchange does not mandate reasons for such downvotes. However, we observed several cases where users mentioned the reasons for downvotes in the comments of those bug reports.
2) Reports having a negative score. As mentioned in Section 2, the score is the sum of upvotes and downvotes. A negative score indicates that a bug report receives more downvotes than upvotes.

We added the second criterion since in some cases, a comment with the listed keywords (i.e.,“downvote”, “down-vote” or “down vote”) does not necessarily mean the commenter share the reason for downvoting (e.g., users discussed the downvote itself in comments). We ended up with 98 bug reports that satisfy the above criteria. We performed the same process as we did in RQ1 to understand the rationale behind those downvotes. We also performed the same process to investigate the rationale for downvotes on answers. We ended up with 44 answers. Our Cohen’s Kappa for these two manual studies are 0.72 and 0.92, respectively, which indicate a sufficient agreement.

Findings: Most of the downvotes on bug reports were made due to the disagreement about whether the reported “bug” is a real bug or the low quality of the bug reports. A brief overview of the rationales that we identified from our qualitative analysis results is shown in Table 8. We observed that 60% of the studied downvotes



(a) The use of upvoting



(b) The use of downvoting

Fig. 12: The use of upvoting and downvoting across different components of a bug thread, namely: the bug report, answers, bug-comments, and answer-comments. Note that downvoting is not allowed on comments.

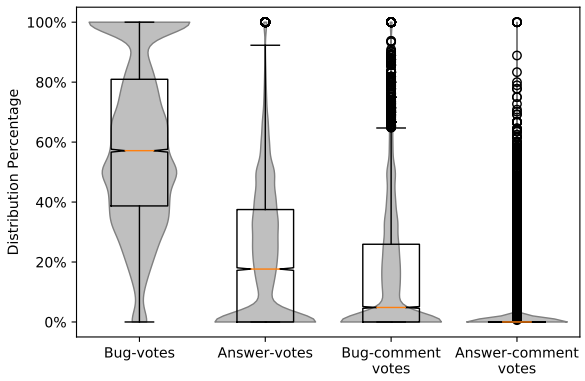


Fig. 13: The ratio of bug-votes, answer-votes, bug-comment-votes, and answer-comment-votes.

were given to express disagreement with the reported "bug" being a real bug. We also observed that the proportion of bug reports that were tagged with "status-bydesign" and have a negative score (37%) is much higher than those with a positive score (10%), which also suggests that users are likely to use downvotes to disagree with the reported bug is a real bug. More specially, 4%, 9%, and 1% of the bug reports were downvoted due to the incomplete, incorrect, and duplicate bug information, respectively. For example, one commenter mentioned "i downvoted because the bug report is incomplete and not helpful. the first step to debugging is almost always to reproduce the problem, and by not providing the browser the op is making it exceedingly hard to verify this bug report"[30]. In other words, 15% of the downvotes were made due to the low quality of bug reports. In addition, 17% of the bug reports received downvotes due to the insignificant or low impact of the bug.

Summary of RQ3

Upvoting is used more frequently on bug reports and answers than comments over the years. The use of the downvoting has increased gradually over the years, with more downvotes on bug reports than on their associated answers. Most of the downvotes on bug reports were made due to disagreement about whether the reported "bug" is a real bug and the low quality of bug reports (i.e., reported bugs are incorrect, insignificant, incomplete and non-reproducible).

5 Discussion

5.1 Studying Mislabeled Issues Reports

We observed 278 bug reports that were labelled as feature requests at the time of submission and 559 feature requests that were submitted as bug reports initially. To further understand the reason behind such mislabeled cases, we performed a manual analysis of 50 questions from both groups.

Bug reports were submitted as feature requests due to users considering their associated problems as design choices and thus asked for changes as feature requests. However, incorrect implementation or failure to make necessary changes caused those problems. For example, a user reported that "Have you considered accepting an answer or starting a bounty" message on the questions page appeared out of place considering questions that were listed on the

30. https://meta.stackexchange.com/questions/151844

Rationale for downvotes	Description (D) - Example (E)	Percentage
Not a bug	(**D**) The reported "bug" is not a real bug. (**E** #308111) commenters responded "No, that is by design, not a bug. That is how Markdown works."	59.7%
Insignificant	(**D**) The reported bug has a low impact or does not create a significant difference to the software. (**E** #96300) commenters mentioned: "I don't really care where it's from, if it's helpful and could be helpful to others...", indicating that the bug isn't helpful and is insignificant.	16.9%
Incorrect	(**D**) Reporting a bug without a proper analysis of the buggy behavior, e.g., reporting mistaken bug information. (**E** #145250) regarding reputation loss by 2 points, a commenter mentioned: "It looks fine to me. Are you sure it decreased twice?"	9.1%
Non-reproducible	(**D**) Bug reports that were not reproducible. (**E** #159376) a commenter responded: "Nope, don't see that kind of behaviour here at all"	9.1%
Incomplete	(**D**) The bug report does not contain complete or adequate information about the bug. (**E** #112819) regarding reputation decrease, a commenter mentioned: "what went down by 2? the question/answer score? the user's rep? **your** rep? care to provide a link?"	3.9%
Duplicate	(**D**) Reporting a duplicate bug report. (**E** #99429) "it seems to be a duplicate, yes. thanks guys. you can vote to close."	1.3%
Offensive	(**D**) Using offensive language to express anger while reporting a bug. (**E** #292490) "I took the liberty of changing the title to your post. calling things "bogus" attracts downvotes. hopefully this more neutral title will make the question better received."	1.3%

TABLE 8: The identified rationales for downvotes on bug reports. The rationales are sorted from the most observed to the least observed rationale. The complete URL for each of the above-mentioned examples is https://meta.stackexchange.com/posts/Bug-ID, where Bug-ID is mentioned in each example (**E**) in the table.

same page. The user suggested to move the text to the right side of the page and submitted a feature request. However, a moderator confirmed that the issue was caused by a bug[31]. We also observed cases where users anticipated missing features and submitted feature requests. However, incorrect implementation prevented the system from behaving as expected. For example, a user observed that the Stack Exchange software showed an error when someone forgot to enter her username on the sign up page. A feature request was submitted to prompt users to enter the username. An answer from a Stack Exchange developer confirmed that the problem was caused due to a bug in the implementation that will be fixed in the next build[32].

Our manual investigation revealed three main reasons for submitting feature requests as bug reports. First, users were not aware of specific design choices or considered that the current design of the system needs to be corrected, which led to the submission of bug reports. For example, a user submitted a bug report because the bottom of a code snippet was clipped on the submission and the user was not aware of that, this happened because the answer contains more than the 30,000 character limits that is supported by the site[33]. Second, users were not aware of recent changes and incorrectly submitted bug reports. For example, Stack Exchange developers made a change to remove shading on accepted answers. A user using IE8 browser considered that the browser might cause the issue and reported that issue as a bug[34]. Third, we observed cases where users considered that observed issues were caused by incorrect implementation whereas the issues were caused by missing features.

31. https://meta.stackexchange.com/questions/317431
32. https://meta.stackexchange.com/questions/237795
33. https://meta.stackexchange.com/questions/13748
34. https://meta.stackexchange.com/questions/22510

5.2 The Implications of our Findings

This section discusses the implications of our findings.

Traditional bug management systems should consider introducing the in-place editing feature. Finding information from a long sequential comment thread of a bug report is a challenging task [2]. The in-place editing feature enables users to edit a bug report directly instead of leaving long thread of comments to change/add information about the bug report. As we observed in RQ1, the in-place editing feature is steadily used over the years in the Stack Exchange bug management system and 57% of the studied edits improved the quality by adding/correcting/clarifying bug-related information (e.g., observed behavior and environment information). **The rollback feature is used to resolve any error due to in-place editing.**

Traditional bug tracking systems should consider introducing the answering and commenting features to encourage the community to better organize and structure their contribution to bug reports. As we observed in RQ2, users discuss different issues in bug-comments and answers, i.e., users tend to clarify bug-related information in bug-comments; provide deployment time of a bug-fix and cause of the bug in answers; while discussing the expected impact of the bug-fix in answer-comments. Therefore, the answering and commenting features probably provide a solution for the long sequential comments by structuring the contribution of community in a better way. We also

observed some issues that are discussed in both bug-comments and answers, such as information about the bug-fix deployment, which may indicate that users are confused about where to discuss such information. **Clear guideline about where (e.g., answers or comments) to discuss a particular issue may need to provide to users.**

Traditional bug reporting systems should consider support for downvoting. In RQ3, we observed that 60% of the studied downvotes were made on bug reports due to the disagreement about a reported "bug" is not a real bug. Traditional bug management systems can include the downvoting feature to eliminate bug reports that are not real bug, rather a feature of the system. Bettenburg et al. reported that bug reporters do not always provide the content required by developers to resolve the bug [7]. Voting on bug reports and answers may provide a mechanism for traditional issue reporting systems to ingrain desirable behavior in reporting, and better maintain bug reports. For example, in RQ3, we observed that users downvote a bug report if the bug report does not contain complete or adequate information about the bug. Meanwhile, **the rationales for downvoting should be explained when performing downvoting (something that is not implemented by the Stack Exchange platform) if the downvoting feature is introduced.** Such explanations can help in improving the bug report and can provide feedback and help novice users to learn how to write a good bug report. However, note that any rules may have undesirable side effects. For example, introducing downvoting feature or voting the quality of bug reports may affect the user participation in reporting bugs and contributing to those bug reports, since prior studies from social science and psychology report that negative feedback negatively impacts user participation [16], [19], [32]. The downvoting feature may also have a negative impact on the health of the community. For example, Cheng et al. observed that downvoted users go on to downvote others and suggest that negative behavior can persist in and permeate throughout a community when left unchecked [13]. Hence, future studies should investigate mechanisms to ensure the quality of user contributions while minimizing the negative impact of such mechanisms on user participation and community health. For instance, one possible mechanism is the one that is currently used by Stack Exchange for avoiding the abuse of the downvoting feature – making cost for downvoting (i.e., every downvote costs the downvoter two reputation points).

6 Threats to Validity

External validity: Threats to external validity are related to the generalizability of our findings. In this study, we performed a large-scale analysis of the Stack Exchange bug management system. Our results may not generalize to other Q&A platforms. However, we would like to point to the fact that leveraging a Q&A website for the purpose of bug management is not very common due to the prevalence of traditional bug management systems. Thus, our studied dataset provides a rare opportunity for the designers of traditional bug management systems to learn about the unique features of managing bugs through the Q&A platform.

In RQ1, we examined the relationship between the number of received in-place edits by a bug report and the bug-fixing time as well as the likelihood of that bug getting fixed. Although we observed a positive association for both examined relationships, we cannot conclude a causation relationship between them (i.e., in-place edits led to a higher likelihood of a bug getting fixed or fixed faster). To alleviate this threat, we also performed a qualitative study in RQ1 to examine the rationale for the performed edits and observed that Stack Exchange users leverage the in-place editing feature to include information that improves the quality of bug reports (e.g., expected behavior, screenshots, and steps to reproduce).

Internal validity: Threats to internal validity relate to experimenter bias and errors. Our study involved several qualitative analyses in each of our RQs. To reduce the bias from human factors, two authors independently labeled the studied data for each analysis and discrepancies were discussed until a consensus was reached. We do note that the levels of inter-rater agreement of all our qualitative studies are high. When conducting the qualitative analysis on edits, answers, and comments in our RQs, it is practically impossible to manually analyze all of them due to their large size and needed effort to perform such in-depth analysis. To minimize the bias in our findings, we studied statistically representative random samples of our studied datasets to ensure a 95% confidence level and 5% confidence interval commonly done by prior studies [12], [31].

7 Related Work

In this section, we discuss prior studies that are related to our study.

7.1 Studying and Improving the Bug Reporting Process

Several studies have been conducted to understand the challenges in managing bug reports. Just et al. performed a study to understand the associated challenges with reported bugs in open source projects (e.g., missing crucial bug information) and provided some suggestions (e.g., providing tool support for users to collect and prepare the information that developers need) to improve next-generation bug management systems [20]. Bettenburg et al. interviewed Eclipse developers and noted that the *steps to reproduce* a bug was highly needed by developers and that inaccurate information hinders the resolution of bugs [6]. The authors also revealed a mismatch in the information provided by reporters and information required by developers to fix bugs.

A number of studies have been conducted to understand the fixing time and rate of bugs that are tracked using traditional bug management systems. Zhang et al. observed that the median bug-fixing time of Eclipse bugs is 586 hours (0.6 months) [35]. Bhattacharya et al. studied the bug-fixing time in open source Android Apps, in which bugs are tracked using Bugzilla and Google Code [8]. The average bug-fixing time ranges from 0.31 months to 4.79 months for different projects. Meanwhile, the bug-fixing rate of these studied projects varies from 0.2% to 49.1%. The bug-fixing times for the two biggest projects (i.e., Android Platform and Firefox Mobile) are 3.3 and 4.7 months, respectively; the bug-fixing rates for these two projects are 65% and 29.9%, respectively. Zou et al. investigated the bug-fixing rate for Eclipse and Mozilla – both projects manage their bugs using Bugzilla [40]. The overall bug-fixing rates for Eclipse and Mozilla are 66% and 44%, respectively. Ripon et al. investigated bugs in four open-source software systems (i.e., JDT, CDT, PDE, and Platform of Eclipse product family) and found that 46.4% to 54.3% of the bugs were fixed within seven days of their submission [27]. In this study, we observed that the median bug-fixing time of Stack Exchange bug management system is 1.6 days, which is much faster than the studied projects of prior studies. One possible explanation is that Stack Exchange is a web application, which requires bugs to be fixed as soon as possible or that bugs in these Q&A based bug management platforms are more trivial. Another explanation is that such Q&A based bug management platforms facilitate bug-fixing. The bug-fixing rate is 38.3%, which is lower than Android Platform but higher than Firefox Mobile.

Furthermore, a considerable number of prior studies have been done to study and improve the quality of bug reports. Sasso et al. performed a large-scale study of bug reports to better understand the components of a bug report that impact its resolution time. They observed that some core elements (e.g., screenshot and the stack trace) of a bug report do impact the resolution time of the report [14]. In our study, we observed that developers asked for more information about a bug in its comments (e.g., screenshoot, observed behavior, and stack traces), and a large percentage of edits were performed to add missing information that improves the quality of the report. In other words, such information is important for developers to fix bugs. Hooimeijer et al. modeled the quality of a bug report by analyzing their features [17]. Surprisingly, the self-reported severity of a bug report is not a reliable indicator of the importance of the bug. Chaparro et al. designed an automated approach to detect the absence (or presence) of observed behavior and the steps to reproduce a bug report [10]. Rejaul et al. proposed an approach to predict the missed key features (e.g., expected behavior) in a bug report [26]. Our observations also highlight the need for such tools to detect the missed key features in a bug report, since we observe that it is common for developers to ask for more information about a bug report in comments. Tian et al. studied the severity of bug reports and proposed an approach to mitigate unreliable factors that lead to the false assessment of bug severity [29].

While prior studies focus on improving bug reports by studying which types of information are important and how to detect those information automatically, we focus on investigating new features (e.g., in-place editing) that could be potentially used in bug management systems. Our downvoting study highlights that such a feature can be used by the community to collaboratively improve the reporting of bugs.

7.2 Leveraging Crowdsourcing to Support Bug Management Activities

Crowdsourcing could be leveraged to improve the bug management, such as bug reporting, bug triaging, and bug fix. Breu et al. observed that interacting with developers provides help in fixing bugs in term of shortening the resolution time [9]. Zhou et al. found that users involved in the development activity, like bug reporting and participating in the community, are more likely to become long-time contributors [39]. Several prior studies proposed bug triaging and assignment approaches by leveraging developers' contribution to Q&A websites (i.e., Stack Overflow) to estimate their expertise in particular domain [4]. Liu et al. developed an approach to leverage the crowdsourced knowledge of Stack Overflow to repair bugs automatically [21].

Different from prior studies which leveraged crowdsourcing to support bug management activities, we investigate the use of three unique features (e.g., in-place editing, answering and commenting, and voting) in Q&A style bug management systems and provide insights for traditional bug management systems. We observe that such unique features help the community to work collaboratively on improving the quality of their bug reports by instilling best practices through voting and downvoting.

7.3 Studying Technical Q&A Websites

Technical Q&A websites like Stack Overflow have accumulated a large volume of such valuable knowledge for developers and software engineering researchers. Various studies have studied how to ensure the quality of knowledge of such Q&A websites. A considerable number of prior studies studied how to improve the quality of content on such Q&A websites [11], [15], [22], [24], [30], [31], [37], [38]. For example, Wang et al. observe that edits on Stack Overflow answers improve the quality of such answers (e.g., fixing bugs in code snippets) [31]. Similarly, our findings suggest that in-place edits improve the quality of bug reports which are expressed as questions on Stack

Exchange, e.g., adding/correcting/clarifying essential bug-related information. In addition, researchers conducted studies on such Q&A websites to extract software engineering related knowledge, e.g., code search and code reuse [1], [3], [25], [28], [34], comment generation [23], [33] and developer discussions [5], [18].

Different from existing studies which focus on studying the quality of technical Q&A websites and leveraging the knowledge from technical Q&A websites, our study provides insights into how to leverage the unique features of technical Q&A websites to improve the traditional bug management systems.

8 Conclusion

In this paper, we study how three unique features, namely, the in-place editing feature, the answering and commenting features, and the voting feature are used in the Stack Exchange bug management system. We find that: 1) 57% of the edits improved the quality of bug reports, such as adding/-correcting/clarfying essential bug-related information (e,g., observed behavior and environment information). 2) The commenting and answering features are used differently, i.e., commenting on a bug report provides an avenue for discussing bug-related information, while answering offers an avenue for providing the solution and explaining the causes of those bugs. 3) Most of the downvotes on bug reports were made due to disagreement about whether the reported "bug" is a real bug and the low quality of bug reports.

Based on our analysis, we offer the following suggestions: 1) Traditional bug management systems should consider incorporating the in-place editing and the roll-back features to avoid long hard-to-follow threads that are associated with bug reports. 2) Bug management systems should consider incorporating the answering and commenting features that can help better structure discussions about bugs and their resolution, however clear guidelines regarding where to contribute (e.g., answers or comments) needs to be provided. 3) Downvoting provides a corrective feedback to bug reporters. While Stack Exchange does not employ a mechanism to explain the reasons for downvotes, we suggest downvotes should be accompanied with an explanation. Such an explanation can help in improving the bug report, provide feedback and help novice users to learn how to write a good bug report.

Acknowledgement This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider, "CAPS: a supervised technique for classifying stack overflow posts concerning api issues," *Empirical Software Engineering*, vol. 25, pp. 1493–1532, 2019.

[2] D. Arya, W. Wang, J. L. C. Guo, and J. Cheng, "Analysis and detection of information types of open source software issue discussions," in *Proc. of the 41st IEEE/ACM International Conference on Software Engineering*, 2019, pp. 454–464.

[3] S. Azad, P. C. Rigby, and L. Guerrouj, "Generating api call rules from version history and stack overflow posts," *ACM Transactions on Software Engineering and Methodology*, vol. 25, no. 4, p. 22, 2017.

[4] A. S. Badashian, A. Hindle, and E. Stroulia, "Crowdsourced bug triaging: Leveraging Q&A platforms for bug assignment," in *Proc. of the International Conference on Fundamental Approaches to Software Engineering*, 2016, pp. 231–248.

[5] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[6] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "Quality of bug reports in eclipse," in *Proc. of the OOPSLA Workshop on Eclipse Technology eXchange*, 2007, pp. 21–25.

[7] ——, "What makes a good bug report?" in *Proc. of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2008, pp. 308–318.

[8] P. Bhattacharya, L. Ulanova, I. Neamtiu, and S. C. Koduru, "An empirical analysis of bug reports and bug fixing in open source android apps," in *Proc. of the 17th European Conference on Software Maintenance and Reengineering*, 2013, pp. 133–143.

[9] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: improving cooperation between developers and users," in *Proc. of the ACM Conference on Computer Supported Cooperative Work*, 2010, pp. 301–310.

[10] O. Chaparro, J. Lu, F. Zampetti, L. Moreno, M. Di Penta, A. Marcus, G. Bavota, and V. Ng, "Detecting missing information in bug descriptions," in *Proc. of the 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 396–407.

[11] C. Chen, X. Chen, J. Sun, Z. Xing, and G. Li, "Data-driven proactive policy assurance of post quality in community Q&A sites," *in Proc. ACM Hum.-Comput. Interact.*, vol. 2, no. CSCW, pp. 33:1–33:22, Nov. 2018.

[12] T.-H. Chen, M. Nagappan, E. Shihab, and A. E. Hassan, "An empirical study of dormant bugs," in *Proc. of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 82–91.

[13] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec, "How community feedback shapes user behavior," in *Proc. of the Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[14] T. Dal Sasso, A. Mocci, and M. Lanza, "What makes a satisficing bug report?" in *Proc. of the IEEE International Conference on Software Quality, Reliability and Security*, 2016, pp. 164–174.

[15] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack overflow considered harmful? the impact of copy&paste on android application security," in *Proc. of the IEEE Symposium on Security and Privacy*, 2017, pp. 121–136.

[16] A. Fishbach, T. Eyal, and S. R. Finkelstein, "How positive and negative feedback motivate goal pursuit," *Social and Personality Psychology Compass*, vol. 4, no. 8, pp. 517–530, 2010.

[17] P. Hooimeijer and W. Weimer, "Modeling bug report quality," in *Proc. of the 22nd IEEE/ACM International Conference on Automated Software Engineering*, 2007, pp. 34–43.

[18] Y. Huang, C. Chen, Z. Xing, T. Lin, and Y. Liu, "Tell them apart: distilling technology differences from crowd-scale comparison discussions." in *Proc. of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 214–224.

[19] D. Ilgen and C. Davis, "Bearing bad news: Reactions to negative performance feedback," *Applied Psychology*, vol. 49, no. 3, pp. 550–565, 2000.

[20] S. Just, R. Premraj, and T. Zimmermann, "Towards the next generation of bug tracking systems," in *Proc. of the IEEE symposium on Visual languages and Human-Centric computing*, 2008, pp. 82–85.

[21] X. Liu and H. Zhong, "Mining stackoverflow for program repair," in *Proc. of the 25th IEEE International Conference on Software Analysis, Evolution and Reengineering*, 2018, pp. 118–129.

[22] N. Meng, S. Nagy, D. Yao, W. Zhuang, and G. Arango-Argoty, "Secure coding practices in java: challenges and vulnerabilities," in *Proc. of the 40th IEEE/ACM International Conference on Software Engineering*, 2018, pp. 372–383.

[23] D. Movshovitz-Attias and W. W. Cohen, "Natural language models for predicting programming comments," in *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013, pp. 35–40.

[24] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, and R. Oliveto, "Toxic code snippets on stack overflow," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[25] M. M. Rahman and C. Roy, "Effective reformulation of query for code search using crowdsourced knowledge and extra-large data analytics," in *Proc. of the IEEE International Conference on Software Maintenance and Evolution*, 2018, pp. 473–484.

[26] K. M. Rejaul, "Key features recommendation to improve bug reporting," in *Proc. of the International Conference on Software and System Processes*, 2019, pp. 1–4.

[27] R. K. Saha, S. Khurshid, and D. E. Perry, "An empirical study of long lived bugs," in *Proc. of the IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering*, 2014, pp. 144–153.

[28] R. Sirres, T. F. Bissyandé, D. Kim, D. Lo, J. Klein, K. Kim, and Y. Le Traon, "Augmenting and structuring user queries to support efficient free-form

code search," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2622–2654, 2018.

[29] Y. Tian, N. Ali, D. Lo, and A. E. Hassan, "On the unreliability of bug severity data," *Empirical Software Engineering*, vol. 21, no. 6, pp. 2298–2323, 2016.

[30] S. Wang, T.-H. Chen, and A. E. Hassan, "Understanding the factors for fast answers in technical Q&A websites," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1552–1593, 2018.

[31] S. Wang, T.-H. P. Chen, and A. E. Hassan, "How do users revise answers on technical Q&A websites? a case study on stack overflow," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.

[32] T. Wang, K. C. Wang, F. Erlandsson, S. F. Wu, and R. Faris, "The influence of feedback with different opinions on continued user participation in online newsgroups," in *Proc. of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2013, pp. 388–395.

[33] E. Wong, J. Yang, and L. Tan, "Autocomment: Mining question and answer sites for automatic comment generation," in *Proc. of the 28th IEEE/ACM International Conference on Automated Software Engineering*, 2013, pp. 562–567.

[34] Y. Wu, S. Wang, C.-P. Bezemer, and K. Inoue, "How do developers utilize source code from stack overflow?" *Empirical Software Engineering*, pp. 1–37, 2018.

[35] F. Zhang, F. Khomh, Y. Zou, and A. E. Hassan, "An empirical study on factors impacting bug fixing time," in *Proc. of the 19th Working Conference on Reverse Engineering*, 2012, pp. 225–234.

[36] H. Zhang, S. Wang, T. Chen, and A. E. Hassan, "Reading answers on stack overflow: Not enough!" *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[37] H. Zhang, S. Wang, T. P. Chen, Y. Zou, and A. E. Hassan, "An empirical study of obsolete answers on stack overflow," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[38] T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, and M. Kim, "Are code examples on an online q&a forum reliable?: a study of API misuse on stack overflow," in *Proc. of the 40th International Conference on Software Engineering*, 2018, pp. 886–896.

[39] M. Zhou and A. Mockus, "Who will stay in the floss community? modeling participant's initial behavior," *IEEE Transactions on Software Engineering*, vol. 41, no. 1, pp. 82–99, 2015.

[40] W. Zou, X. Xia, W. Zhang, Z. Chen, and D. Lo, "An empirical study of bug fixing rate," in *Proc. of the 39th Annual Computer Software and Applications Conference*, 2015, pp. 254–263.

Aaditya Bhatia Aaditya is currently working towards his Ph.D. and is a M.Sc. graduate from Software Analytics and Intelligence, Queen's University, Canada. His research interests include machine learning, software analytics, data warehousing, natural language processing, and mining software repositories.

Shaowei Wang Shaowei Wang is an assistant professor in the Department of Computer Science at University of Manitoba. He obtained his Ph.D. from Singapore Management University and his BSc from Zhejiang University. His research interests include software engineering, machine learning, data analytics for software engineering, automated debugging, and secure software development. His work has been published at premier software engineering venues such as the TSE, EMSE, ASE, and ICSME. He is one of four recipients of the 2018 distinguished reviewer award for the Springer EMSE (SE's highest impact journal). More information at: https://sites.google.com/site/wswshaoweiwang/.

Muhammad Asaduzzaman Muhammad Asaduzzaman is a postdoctoral research fellow in the Software Analysis and Intelligence Lab (SAIL) at Queen's University in Kingston, Canada. His research interests include software maintenance and evolution, mining software repositories, program comprehension, reverse engineering, and recommendation systems. His work has been published at premier software engineering venues such as the EMSE, ASE, ICSME, and MSR. Before moving to Canada, he studied at Khulna University in Bangladesh, where he received his BSc (2006). He completed his M.Sc. (2012) and Ph.D. (2018) from the Department of Computer Science at the University of Saskatchewan in Saskatoon, Canada. More about Muhammad can be read on his website: https://muhammad-asaduzzaman.com.

Ahmed E. Hassan Ahmed E. Hassan is an IEEE Fellow, an ACM SIGSOFT Influential Educator, an NSERC Steacie Fellow, the Canada Research Chair (CRC) in Software Analytics, and the NSERC/BlackBerry Software Engineering Chair at the School of Computing at Queen's University, Canada. His research interests include mining software repositories, empirical software engineering, load testing, and log mining. He received a PhD in Computer Science from the University of Waterloo. He spearheaded the creation of the Mining Software Repositories (MSR) conference and its research community. He also serves/d on the editorial boards of IEEE Transactions on Software Engineering, Springer Journal of Empirical Software Engineering, and PeerJ Computer Science. Contact ahmed@cs.queensu.ca. More information at: http://sail.cs.queensu.ca/.