An Exploratory Study of the Differences Between Bug Reports and Feature Requests Using the Stack Exchange Question and Answering Platform

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

Abstract—Intuitively, one would expect that feature requests and bug reports (and their management) would differ. Yet, no prior study has ever explored such differences systematically since there exists no large enough and high-quality dataset where issues are tagged as either bug reports or feature requests. In this paper, we investigate the differences between bug reports and feature requests in the issue management system of Stack Exchange. We focus our study on the Stack Exchange system since: 1) it contains a large number of high-quality issues that have been tagged as bug reports or feature requests by domain experts heightening the efficacy of our study); 2) the issue management is carried out through a question and answer (Q&A) platform, which provides us with a richer perspectives on the differences between the management of bug reports and feature requests. Our study finds that bug reports and feature requests differ significantly from each other along various dimensions, such as the amount of community contributions, the content of the issues, and the characteristics of the contributing users. Through constructing models, we observe that the question and answer (Q&A) platform specific features are important in issue management. *[Adi says: Using Q&A features, we are able to build classification models for distinguishing bug reports from feature requests with a higher AUC than models built on non-QA or traditional features]*. The designers of traditional issue management systems may wish to consider integrating such features to improve the management of issue reports.

Index Terms—Bug management systems, Stack Exchange, question & answering platform

1 Introduction

Issue management systems are an integral part of software development practices. Such systems track reported bugs and manage their resolution while also capturing feature requests, and tracking their progress. Intuitively, one would expect that the management of feature requests and bug reports would differ. A good understanding of such differences would help us better leverage this rich data for various tasks. For instance, the rich data available in issue management systems has been used to develop methods to avoid the occurrence of future bugs [9], [16], [28], to identify the most suitable developer to work on a new issue [2], [34], [37], [54], and to determine the needed time to address an issue [9], [20], [35], [52].

Yet, no study has ever explored such differences in a systematic manner. A key reason for the lack of such a study is that there does not exist large enough datasets, where issues are tagged by domain experts as bug reports versus feature requests. Many of the prior studies [1], [22], [56] tagged a small number of issues and such tagging was performed by researchers instead of domain experts. The small number of tagged data limits the generalization and robustness of any subsequent observation.

Thus, in this paper, we study the management of bug reports versus feature requests using a large number of high-quality issue reports from the issue management system of Stack Exchange. The issue reports of Stack Exchange websites are managed using its Q&A platform, namely Meta Stack Exchange[1]. We use the Meta Stack Exchange

- *Aaditya Bhatia, Gopi Krishnan Rajbahadur, 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, krishnan, 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://meta.stackexchange.com

to conduct our study for the following key reasons. First, the Meta Stack Exchange has accumulated a large number of high-quality issue reports that are tagged by Stack Exchange developers and community members (i.e., domain experts) instead of researchers. Using these issue reports to study the difference in management of bug reports and feature requests allows us to conduct a large scale study that might provide more robust and generalized insights. Second, unlike traditional issue management systems (such as Bugzilla[2] or GitHub[3]), Meta Stack Exchange uses a Q&A platform for issue management. These Q&A platform presents unique features that are not available in traditional issue reporting systems. For example, the response in the Q&A platform is divided into three different channels (question comments, answers, and answer comments). Any changes to a bug report or a feature request from the community (such as adding new information, providing clarifications or editing incorrect information) appear as a long thread of sequential comments in a traditional issue management system (e.g., Bugzilla). On the contrary, the Q&A platform allows its users to directly edit an issue report or any of its answers instead of performing a long list of sequential commenting (i.e., in-place editing). The reputation mechanism incentivizes users to participate in the issue management activities (e.g., submitting issue reports, providing replies in the form of answers and comments, editing issue reports to improve them). Furthermore, both questions and answers can be voted, both upvotes and downvotes are permitted. The voting features can help to maintain the quality of issue reports. This is because adding incorrect information can be penalized by downvotes and that negatively affects the reputation. *[Adi says: [8]]*. Thus, studying the difference between bug reports and feature requests along these unique features might provide us with a richer perspective on the differences between the bug reports and feature requests.

Therefore, in this study, we first compared bug reports and feature requests along various aspects in the context of Q&A platform based issue management systems. We found that:

1) **Bug reports and feature requests are significantly different from each other in various aspects.** Feature requests (290 words in median) are generally longer than bug reports (188 words in median), meanwhile feature requests receive longer comments and answers. Feature requests also receive more contributions through answers and votes, compared to bug reports. Feature requests are more likely to have higher sentiment scores than bug reports. In general, feature requests have more contributors than bug reports, while users who contribute to bug reports have a higher reputation than those for feature requests. In addition, Stack Exchange users are also active in managing issue reports. Within seven days of submission, issue reports collected 66.2% of the answers, 84.5% of the comments, 57.4% of the votes, and 52.2% of the edits that they received throughout the studied time period.
2) Later, *[Shaowei says: to verify whether Q&A platform specific features assist in improving the effectiveness of classifying issue reports and such features are significantly important in classifying issue reports when controlling other confounders]*, we construct several classifiers. Our study results show that: *[Shaowei says: update later]*
    a) *[Adi says: Addition of Q&A platform specific features to the traditional features improves the predictive power of classifiers as compared to classifiers built on traditional features. ~~Our results are consistent for different classes of classifiers.~~]*
    b) **Our constructed classifiers exhibit sufficient explanatory power.** The random forest classifier receives a median AUC of 87.8% when we combine unique factors pertaining to the Q&A platform along with the textual content even after just one day of the submission of issue reports.
    c) **Q&A platform specific factors are important factors for distinguishing bug reports and feature requests.** While text is the most important factor, we observe that Q&A platform specific factors (e.g., reputation of users and vote count) are also identified as the important factors in distinguishing bug reports from the feature requests.

[Adi says: re-written the entire thing here:] *[Shaowei says: updated]* In summary, our study shows that bug reports and feature requests are significantly different across various dimensions and they are managed differently in the Stack Exchange issue management system. Q&A platform specific factors (such as the reputation, voting, and different channels for providing response to an issue report) provide rich features that can be leveraged in classifying bug reports from feature requests. Moreover, our trained classifiers deemed Q&A features to be *[Shaowei says: significantly important]*the most important in distinguishing bug reports from feature requests. While our study explains the dynamics of bug reports and feature requests on Q&A based issue management system, the goal of this study is to help traditional issue management systems. Similarly, Q&A based features my help issue management systems in providing rich features for addressing other tasks like issue prioritization, triaging, cleaning, and benchmark creation. Traditional issue management systems, such as GitHub, may wish to consider integrating such features for better issue management (e.g., distinguishing bug reports and feature requests). The dataset we used in this study including

2. https://www.bugzilla.org
3. https://github.com

our results and scripts are publicly available[4] to support future replication.

The remainder of this paper is organized as follows. Section 2 provides the needed background about the issue management system of Stack Exchange. Section 3 introduces our studied dataset. The approaches and the results of our research questions are described in Sections 4 and 5. Section 6 discusses the implications based on our observations. We describe prior work related to our study in Section 8. Section 7 presents threats to the validity of the observations of our study. Finally, Section 9 concludes our study.

2 Background: Issue Management on Stack Exchange

Stack Exchange provides a Q&A platform for its users to share knowledge across various domains (e.g., programming, statistics, and mathematics). Stack Exchange uses its own Q&A platform, namely, Meta Stack Exchange[5] to manage its issue reports (i.e., bug reports and feature requests).

Issue reports are treated as regular questions in the Meta Stack Exchange, and are managed through the Q&A platform. All issue reports are labeled with the "bug" or "feature-request" tags. Thus, we consider all questions with the "bug" or "feature-request" tag as bug reports or feature requests, respectively, for our study. From here onwards, we refer to a Meta Stack Exchange question with the tag "bug" as a bug report and a question with the tag "feature-request" as a feature request, if not mentioned otherwise. For example, Figure 1 shows an example of a bug report[6]. The report describes a bug on displaying the close vote.

Stack Exchange expects its users to provide a title, body, and tags for each reported issue. There is a limit of 150 and 30,000 characters for the title and the body of an issue report, respectively. The title briefly describes the reported issue and the body provides a detailed description of the issue. Besides the tag "bug" or "feature-request", users are also encouraged to label an issue report with other tags to better organize issue reports. Each issue report also includes the reporter information, enabling any member of the community to visit the profile of the issue reporter.

All issue reports are open to the whole community to view, edit, vote on, leave comments, and provide answers. For instance, users can edit an issue report to fix grammar and spelling mistakes; clarify the description of the issue (without changing its original meaning); add additional information (i.e., related links)[7]. Users are allowed and encouraged to post answers to issue reports. Figure 2 shows an example of an answer to an issue 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. Stack Exchange also allows users to vote on issue reports. 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."[8] 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. Commenting can also be performed on an issue report (i.e., referred to as *issue-comment*) and its answer (i.e., referred to as *answer-comment*). Only upvotes are allowed on comments (in contrast to both upvotes and downvotes on issue reports and their answers). To ease presentation, we refer to an issue report along with all its associated answers and all their associated comments as an *issue thread*. We use the same terminology for bug reports and feature requests, and refer to a bug report or a feature request, its associated comments and answers as the *bug report thread* or *feature request thread*, respectively.

Once an issue is addressed, moderators will change the status of the issue to "status-completed". In this study, we only consider those issues that are addressed, since the "status-completed" tag of an issue report indicates that the issue report was validated by the community and developers, and we have a complete picture of the whole life cycle of that particular issue.

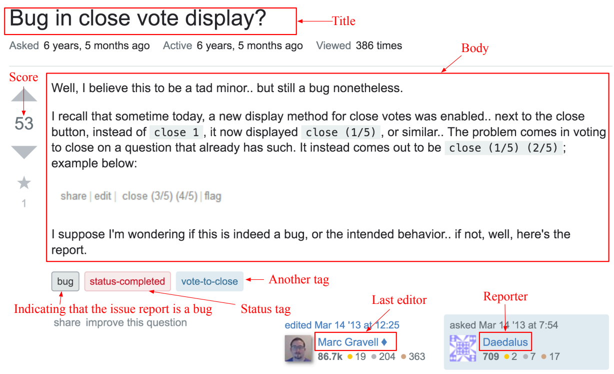
2.1 Comparison with traditional issue management systems

Q&A platform of Stack Exchange incorporates a set of features that are missing in traditional issue management systems (i.e., in-place editing, reputation, and voting). Table 1 compares the management of issue reports between Stack Exchange and GitHub issue management system–an example of a traditional issue management system. The in-place editing feature on Stack Exchange allows community members to contribute to issue reports without incorporating a long list of sequential comments. Whereas the GitHub issue management system allows only the issue reporter to perform the in-place editing. Other community members' made changes through commenting, which increases the difficulty of finding useful information from such a long list of comments [3]. In addition, Stack Exchange has the answering channel in addition to the commenting channel, which enables the users to have different channels to contribute to discussions about issues.

4. https://github.com/SAILResearch/replication-18-Adi-bugvsfeature
5. https://meta.stackexchange.com/
6. https://meta.stackexchange.com/questions/171817
7. https://meta.stackexchange.com/help/editing
8. https://stackoverflow.com/help/whats-meta

Characteristics	Stack Exchange Issue Management System	GitHub Issue Management System
Voting mechanism	Upvoting can be done on issue reports, answers, and comments. Downvoting is only available on issue reports and their answers.	Voting is not supported.
In-place editing	Supports in-place editing. All users can make or suggest changes. Both issue reports and their answers can be edited.	Only the issue reporter can perform in-place editing on issue reports.
Augmentation of information	Issue reports can be answered or commented. Answers can be commented.	Responses to issue reports appear as sequential comments.
Reputation system	Users are encouraged to contribute to issue management activities by rewarding reputation points.	Reputation is not supported.

TABLE 1: Comparisons of issue management systems between Stack Exchange and GitHub.


Title
Body
Score
53
1
share | edit | close (3/5) (4/5) | flag
Another tag
Indicating that the issue report is a bug
Status tag
Last editor
Reporter
share improve this question
86.7k ● 19 ● 204 ● 363
709 ● 2 ● 7 ● 17

Bug in close vote display?

Asked 6 years, 5 months ago Active 6 years, 5 months ago Viewed 386 times

Well, I believe this to be a tad minor.. but still a bug nonetheless.

I recall that sometime today, a new display method for close votes was enabled.. next to the close button, instead of `close 1`, it now displayed `close (1/5)`, or similar.. The problem comes in voting to close on a question that already has such. It instead comes out to be `close (1/5) (2/5)`; example below:

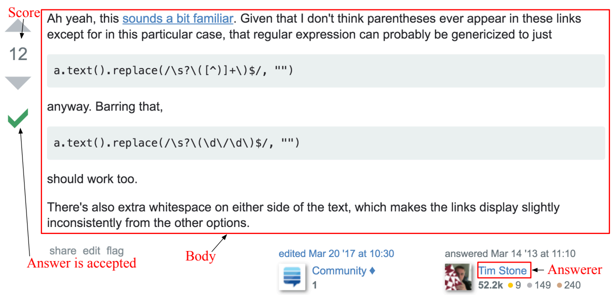
I suppose I'm wondering if this is indeed a bug, or the intended behavior.. if not, well, here's the report.

bug status-completed vote-to-close

edited Mar 14 '13 at 12:25 Marc Gravell ♦

asked Mar 14 '13 at 7:54 Daedalus

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



Ah yeah, this sounds a bit familiar. Given that I don't think parentheses ever appear in these links except for in this particular case, that regular expression can probably be genericized to just

```
a.text().replace(/\s?\([^)]+\)$/, "")
```

anyway. Barring that,

```
a.text().replace(/\s?\(\d\/\d\)$/, "")
```

should work too.

There's also extra whitespace on either side of the text, which makes the links display slightly inconsistently from the other options.

edited Mar 20 '17 at 10:30 Community ♦

answered Mar 14 '13 at 11:10 Tim Stone

Fig. 2: An example of a response in the form of an answer to the bug report that is shown in Figure 1. The answer contains a suggested change in the code to fix the problem.

The reputation mechanism promotes the active participation of community members when managing issue reports (e.g., submitting issue reports, providing explanations or solutions in the form of replies and comments, and editing issue reports in-place) [7]. However, such a reputation mechanism is missing in the GitHub issue management system.

Finally, the voting feature available in Stack Exchange issue management system ensures the quality of the contribution because adding incorrect information probably will be penalized by the community members through down voting [7]. GitHub issue management system does not support voting on issue reports. Such a voting mechanism can help distinguish good issue reports from bad ones. *[Adi says: added by prev paper citaitions to back our claims here!]*

3 Data Collection

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

For this study, we downloaded a publicly available data dump of Meta Stack Exchange from archive.org[9]. The data dump contains all the activities around posted questions 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. We use the following files from this set:

- **Posts.xml:** Contains all questions and answers. To collect bug reports and feature requests, we collected all questions with the tag “bug” or “feature-request”, respectively. We then collected the answers that are associated with those bug reports and feature requests.
- **Comments.xml**: Contains all comments that are associated with questions and answers. We collected the comments that are associated with bug reports and their answers. Similarly, we also collect the comments that are associated with feature requests and their answers.
- **PostHistory.xml**: Contains all edits that affect the content of any questions or answers. We only collected those edits that affect the title, body, and tags of bug reports and feature requests.
- **Votes.xml**: Contains all the votes that are associated with questions, answers, or their comments. We collected all the votes that are associated with bug reports, answers, and their comments. Similarly, we collected all the votes that are associated with the different parts of a feature request.

After collecting the data, we selected our studied issue reports based on the following criteria:

1) We removed issues that have no user id. We observed that some issue reports do not have a valid user id. Our study investigates the contributing users, which

9. https://archive.org/download/stackexchange

requires a valid user id for each issue report. We removed 941 such issue reports.

2) We removed all issue reports that have been shifted from "bug" to "feature-request", or vice-versa before. Such cases may indicate even the community members were confused about their types (bug report or feature request). To reduce the bias from such cases, we removed 786 feature requests that were reported as bugs, and 406 bugs that were reported as feature requests.
3) We removed all issue reports that were tagged both with "bug" and "feature-request". An issue report with both of the tags indicates that reporters might not sure about the type of the issue report. We removed 848 such reports.
4) We removed all issue reports that were not tagged with "status-completed". The "status-completed" tag indicated that an issue report was validated by the community members and Stack Exchange developers. This also ensured that we had a complete picture of the whole life cycle for such issue reports. In total, we removed 11,720 bug reports and 19,187 feature requests that were not tagged with "status-completed".

Finally, we ended up with 6,546 bug reports and 2,502 feature requests. *[Adi says: We collected various contributions of these bug reports; namely, answers, issue comments, answer comments, edits and votes. Table 2 shows an overview of our studied dataset. We capture these contribution even after the "status-completed" tag.]*

Category	Bug Reports	Feature-requests
Issues	6,546	2,502
Answers	6,057	2354
Issue Comments	4,606	1,817
Answer Comments	3,435	1,772
Edits	16,681	7,105
Votes	57,694	60,103

TABLE 2: Overview of our studied dataset.

4 RQ1: How does the management of bug reports differ from that of feature requests in the context of Q&A platform?

Intuitively, the management of bug reports and feature requests would differ. However, none of the prior studies systematically investigate how they differ from each other in the context of Q&A platforms. For example, closing a bug report might be much faster than a feature request; developers might be more likely to have more extensive discussions about feature requests compared to bug reports. *[Adi says: Understanding such systematic differences enables us to streamline the issue resolution process. For example, triaging of issue reports, automatic classification of issue reports into bugs or feature requests, and automatic flagging of important reports.]* Therefore, in this RQ, we investigate whether bug reports and feature requests are different from each other or not. Our empirical investigation offers a deeper understanding of the differences between bug reports and feature requests, not only from the perspective of their content but also based on how the community contributes to them and the characteristics of the contributing users.

To understand whether bug reports and feature requests differ from each other, we investigate them along three different dimensions to answer the following questions.

- **Do bug reports and feature requests differ in terms of community contributions?** We consider the following four avenues for contributing to bug reports or feature requests: a) answering, b) commenting c) editing, and d) voting.
- **Do bug reports and feature requests differ in terms of their content characteristics?** We consider the followings aspects of textual content: a) length in word count, b) readability score, and c) expressed sentiment.
- **Do bug reports and feature requests differ in terms of contributing users?** We check whether the number of contributing users and their reputation vary across bug reports and feature requests.

Below we describe in detail the approaches that we follow to perform our quantitative analysis and the results we obtain along the three above-mentioned dimensions.

4.1 Do bug reports and feature requests differ in terms of community contributions?

Approach: Stack Exchange issue management system offers different avenues for community contributions. We consider contributions through answering, commenting, editing, and voting issue reports (i.e., bug reports or feature requests), and their associated answers. It should be noted that Stack Exchange does not allow the downvoting on comments (i.e., issue-comment and answer-comment) but allow the upvoting on them. We compare such contributions (i.e., answers, comments, edits, and votes) between bug reports and feature requests. Furthermore, we consider those contributions across different time periods to understand whether community contributions vary across time or not. We not only calculate contributions that occurred within one, three, and seven days from the reporting of an issue report but also calculate the total number of contributions throughout the lifetime of an issue report (i.e., the time between reporting an issue and receiving the status-completed tag (closing time)).

In traditional issue management systems, all contributions to an issue report appear as a long thread of comments. Thus, we also merge all these contributions together to understand whether bug reports and feature requests differ when the issue management systems do not offer different avenues for contributions like the Stack Exchange

Q&A platform. We perform the Mann–Whitney U test to determine whether or not the differences between bug reports and feature requests (along the above-mentioned dimensions) are statistically significant and use the Cohen's d test to measure the magnitude of the differences. We use the thresholds provided by Cohen [29] to determine the effect size, where $|d| < 0.2$ indicates that the effect size is negligible, $0.2 \leq |d| < 0.5$ indicates that the effect size is small, $0.5 \leq |d| < 0.8$ indicates that the effect size is medium, otherwise the effect size is large.

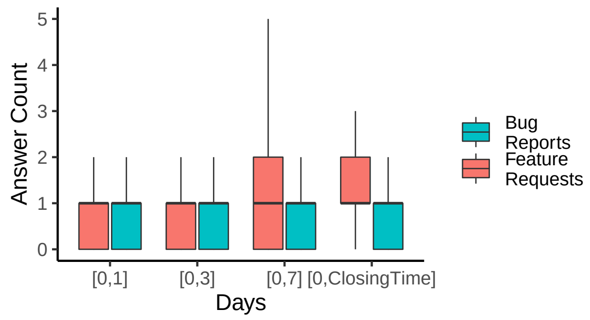


Fig. 3: The number of answers that were received by an issue report within one, three, and seven days since its reporting. We also consider the number of answers received throughout the lifetime of an issue report.

Findings: Feature requests receive more answers than bug reports when we consider seven or more days, as shown in Figure 3. Both bug reports and feature requests receive a median of one answer within seven days. Bug reports get a minimum of zero and maximum of 11 answers, while feature requests receive a minimum of zero and maximum of 15 answers within the same period of time. The difference between bug reports and feature requests in terms of the number of the received answers is statistically significant (p-value<0.05) with a small effect size (0.26). The difference becomes more pronounced when we count all the answers throughout the lifetime of a bug report and a feature request.

Bug reports and feature requests receive a median of two edits as of getting closed, considering edits in the title, body and tag parts. However, a bug report (i.e., title, body, and tags) receives its second edit within seven days (median), whereas a feature request receives its second edit after seven days of its reporting – highlighting that bug reports receive edits sooner than feature requests.

Bug reports and feature requests receive similar number of comments. Both bug reports and feature requests receive a median of two comments throughout their lifetimes. The difference between the comment count of bug reports and feature requests is statistically significant (p-value<0.05), while with a negligible effect size (0.07).

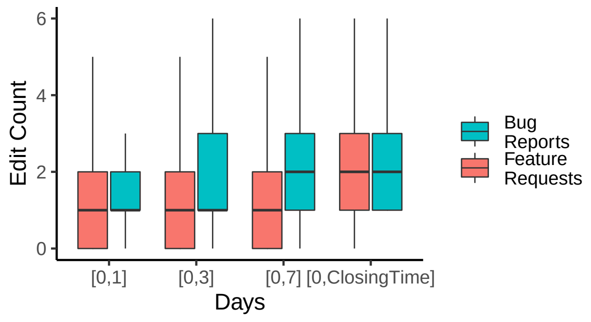


Fig. 4: The number of edits that were received by an issue report within one, three, and seven days since its reporting. We also consider all the edits received throughout the lifetime of an issue report.

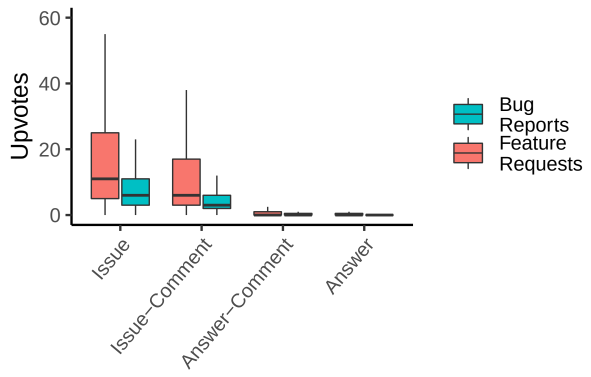


Fig. 5: Upvotes on different components of a bug report and a feature request.

Feature requests get more votes than bug reports. Figures 5 and 6 show the upvotes and the downvotes on different components of bug reports and feature requests, respectively. The number of received upvotes for feature requests (with a median of 11) is higher than that of bug reports (with a median of six). The differences are also statistically significant. All comparisons between bug reports

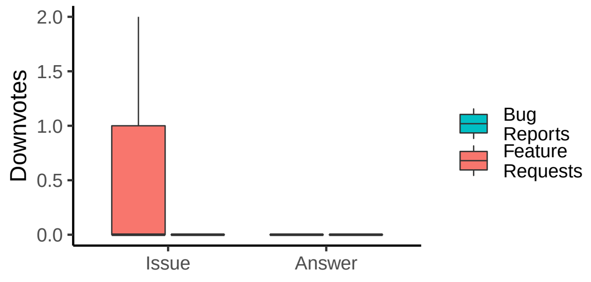


Fig. 6: Downvotes on issue reports and their answers. Please note that comments cannot be downvoted.

and feature requests in Figure 5 and Figure 6 are statistically significant (p-value<0.05). The effect sizes for upvotes on issue and answer are non-negligible (0.57 (medium) and 0.35 (small)) and negligible for answer-comment and issue-comment. The effect sizes for the comparison of downvotes of issue and answer between bug reports and feature requests are small (0.41 and 0.26).

Both bug reports and feature requests get a median of zero downvotes, however, **more feature requests (33.0%) received downvotes than bug reports (11.7%).** A closer investigation reveals that community members use voting to indicate whether they like the idea of a given bug report or a feature request. When community members do not like a requested feature, they would cast a downvote – explaining a large proportion of feature requests receive more downvotes than bug reports. While for bug reports, the downvoting is usually used to complain about the validity of a bug report. For example, in a bug report[10], one of the bug-comments mentioned: "Nope, don't see that kind of behaviour here at all", which indicates that the bug report was downvoted possibly due to that reported bug was not reproducible.

Results from our previous experiments show that feature requests tend to receive more votes and answers than bug reports. One explanation to our observation that feature requests often get more answers and votes than bug reports is that feature requests are more open ended and are discussion oriented. Stack Exchange users use votes to indicate if a feature-request should be implemented or not. Whereas in the case of bug reports, upvotes and downvotes still indicate the quality and clarity of the bug reports. Voting is also used to indicate the quality of posts in other Stack Exchange sites, though they still act as an indicator for inviting discussion and community participation[11].

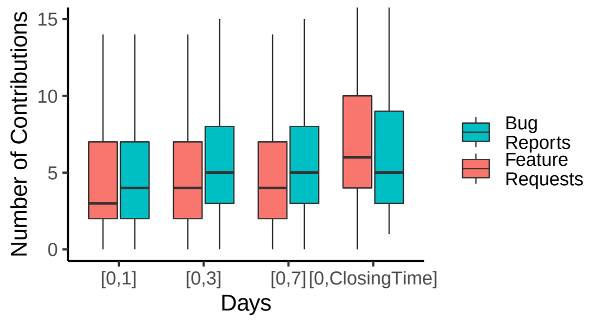


Fig. 7: Total number of contributions (i.e., answer count, comment count, edit count, and vote count) that were received by an issue report within one, three, and seven days since its reporting. We also consider all the contributions received throughout the lifetime of an issue report.

10. https://meta.stackexchange.com/questions/159376
11. https://stackoverflow.com/help/whats-meta

In total, feature requests get more contributions (i.e., the sum of answer count, comment count, edit count, and vote count) as compared to bug reports during the lifetime of issue reports, as shown in Figure 7. The median values are five and six contributions for bug reports and feature requests, respectively. Their differences are statistically significant (p-value<0.05) with a small effect size (0.24). However, when we consider the total number of contributions received within one, three and seven days, bug reports receive more contributions than feature requests. This is because bug reports get more edits than feature requests until seven days. After seven days, feature requests not only receive more edits but also more answers as well. In other words, bug reports are more likely to receive contributions at the first several days since its reporting, while feature requests are more likely to receive contribution continuously. One possible reason is that once a bug is fixed, the discussion on the bug stops, while a feature request is more likely to have extensive discussion about it.

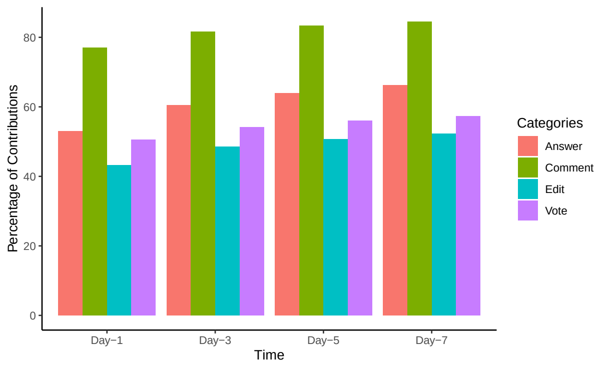


Fig. 8: The percentage of different categories of contributions after one, three, five and seven days of the submission of issue reports

Stack Exchange users are active in contributing on issue reports. Contributions (i.e., answers, comments, edits, votes) from Stack Exchange users take time to accumulate since such contributions are not available at the time of the submission of issue reports. However, Stack Exchange users are active in participating on issue reports, meaning a significant number of contributions are collected within a short period of time. For example, within seven days of the submission, issue reports collected 66.2% of the answers, 84.5% of the comments, 57.4% of the votes, and 52.2% of the edits that they received throughout the studied time period, as shown in Figure 8. Even after one day of submission, issue reports collected 52.9% of the answers, 77.0% of the comments, 43.2% of the votes, and 50.6% of the edits that they received throughout the studied time period. This perhaps because the inclusion of the reputation mechanism incentivises users to participate on issue management activities.

4.2 Do bug reports and feature requests differ in terms of their content characteristics?

Approach: We examine the length, readability score, and the strength of the expressed sentiment in textual content to understand whether these aspects vary between bug reports and feature requests. First, we calculate the length of an issue report by considering the number of words in it. Second, we calculate the readability score of an issue report using the Flesch Kincaid readability score [12] that indicates the difficulty of understanding a passage in English. A higher readability score indicates that the given issue report is easier to read. We use the Python package textstat[12] to calculate the Flesch–Kincaid readability score. We calculate the readability score for different components of an issue thread.

To check if users express different sentiments in bug reports versus feature requests, we perform a sentiment analysis on them. Sentiment analysis determines the polarity or the expressed emotion in a sentence or document (in our case, a bug report or a feature request). We use the python **Vader** package [24] to perform the sentiment analysis. Vader provides a compound score, a uni-dimensional measure of the sentiment, for a given piece of text. We use the term sentiment score to refer to the compound score we obtained through Vader and we compare bug reports and feature requests using that score. The compound score can be divided to obtain following sentiment categories: (a) positive sentiment: compound score $\geq$ 0.05 (b) neutral sentiment: -0.05 $<$ compound score $<$ 0.05 (c) negative sentiment: compound score $\leq$ -0.05.

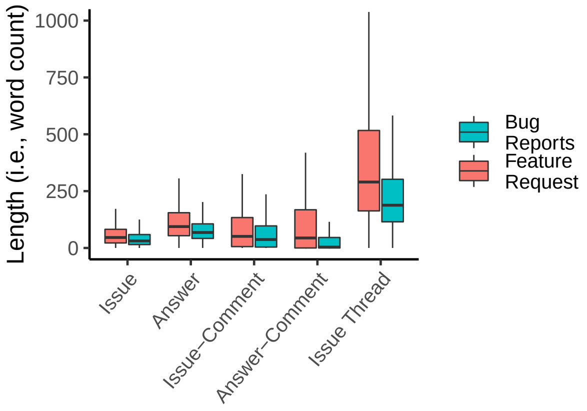


Fig. 9: Length of issue threads and their components.

Findings: Feature request threads (290 words in median) are significantly longer than bug report threads (188 words in median), as shown in Figure 9. We also observe that more words are used for the various components of a feature request thread than those of a bug report thread. The result of Mann–Whitney U test shows that the length difference between bug report threads and feature request threads is statistically significant (p-value<0.05), with a medium effect size (0.61). We hypothesize that this is likely due to feature requests being more open for discussion compared to bug reports. Besides, feature requests may need more explanations to justify the importance of the requested features.

12. https://pypi.org/project/textstat/

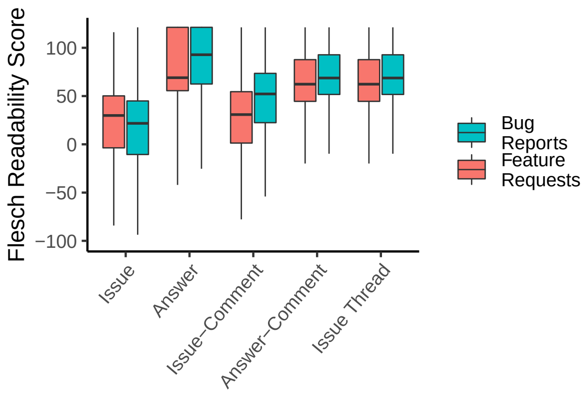


Fig. 10: Readability scores of issue threads and their components.

In terms of the report itself, feature requests have significantly higher readability scores than bug reports, while other components of bug report threads have higher readability scores than those of feature request threads, as shown in Figure 10. The median readability score of the body of bug reports is 21, the score is 30 for that of feature requests. The differences are statistically significant (p-value<0.05) with a negligible effect size of (0.07). However, we observe that the median readability scores of other components (i.e, answers, answer-comments, and bug-comments) of bug report threads are higher than those of feature request threads. This is likely due to the simplicity of the followup comments/answers about a bug report versus a feature request where such followup comments/answers are parts of deeper discussions.

Feature request threads and their components have significantly higher sentiment scores than those of bug report threads and their components, as shown in Fig 11. In general, bug report and feature request threads have a median of zero and 0.32 compound sentiment scores, respectively. The differences are also statistically significant (p-value<0.05) and non-negligible (0.25). The median sentiment scores of all feature request components lie beyond the 0.05 threshold, which are significantly higher than those of bug report components (median values fall in the range of neutral), indicating that users are more likely to express positive sentiments in feature request components, while expressing neutral sentiments in bug report components.

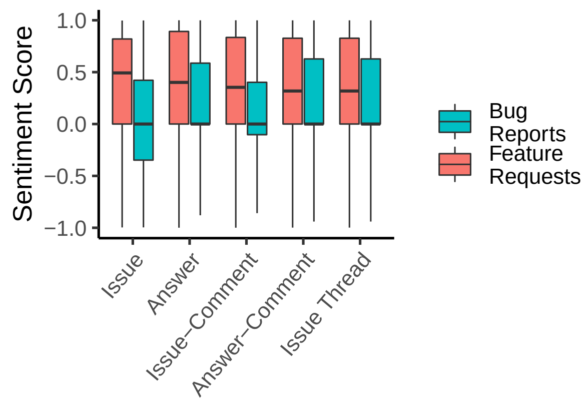


Fig. 11: Sentiment scores of issue threads and their components.

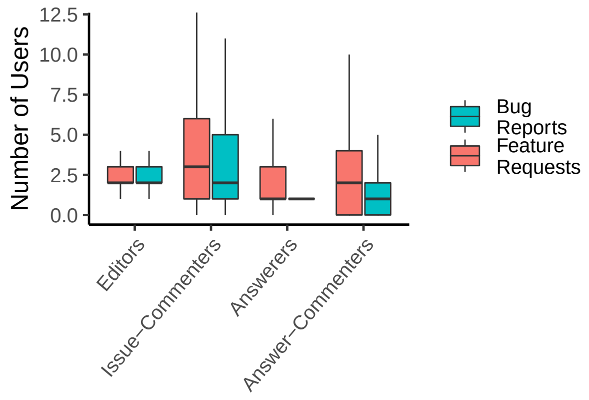


Fig. 12: Number of unique users who contribute to different components of bug report threads and feature request threads.

4.3 Do bug reports and feature requests differ in terms of contributing users?

Approach: We analyze the characteristics of the users associated with bug reports versus feature requests along two different aspects: a) characteristics of users who report bugs or request features b) characteristics of users who contribute to bug reports or feature requests through answering, commenting, or editing. We remove the contributions through voting from this analysis because Stack Exchange does not provide the user id for any upvotes or downvotes. We not only compare the number of users who participate in different activities but also compare their reputation on Meta Stack Exchange website at the time of reporting issue reports or contributing to those issue reports.

Stack Exchange data dump only provides the latest reputation points of any user. To mitigate this issue, we calculate the reputation point of a user (u) at any time (t) by considering the number of asked questions, the number of provided answers, and the number of received votes on those asked questions or provided answers prior to the time t. To do this, we follow the Stack Exchange's official guidelines for calculating reputation score[13].

Findings: In general, for all components of issue threads, feature requests tend to have more contributors compared to that of bug reports, as shown in Figure 12. In general, more unique users are involved in commenting on feature requests than on bug reports. Bug reports have a median of two commenters and feature requests have a median of three commenters. The difference between two groups is statistically significant (p-value<0.05) with a negligible effect size (0.11). We also observe similar scenario for answer-comments. The difference between two groups is statistically significant (p-value<0.05) with a medium effect size (0.61). Both bug reports and feature requests

13. https://stackoverflow.com/help/whats-reputation

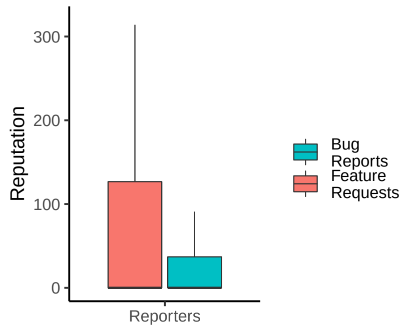


Fig. 13: Reputation of issue reporters at the time of reporting such issues.

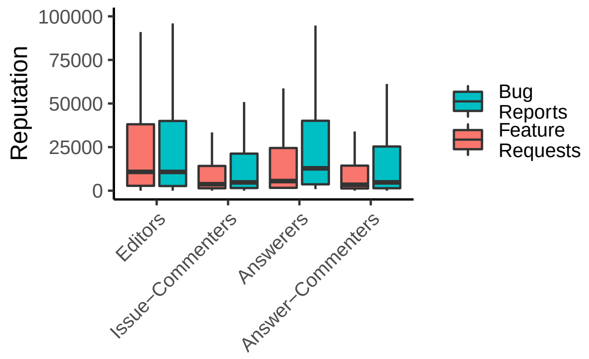


Fig. 14: Reputation of users who contribute to different components of bug report threads and feature request threads.

receive the contribution of a median of one answerer, and two editors as shown in Figure 12. The differences are also again statistically significant (p-value<0.05). The effect sizes are medium (0.67) for answerers and negligible (0.15) for editors of bug reports and feature requests. One possible reason of feature requests having more contributors than bug reports is that feature request are open and easy to raise extensive discussion. For instance, a user asked for a feature that allows users to retract a close vote[14]. This feature request raised extensive discussion, including 21 answers and more than 100 comments on the request itself and its associated answers. The community contributed in various aspects and expressed different opinions on it.

Although the median reputation of both bug reporters and feature requesters is zero as shown in Figure 13, feature requesters have a higher mean reputation (4,071) compared to that of bug reporters (2,795). When we manually validate we observe that many of the issues were contributed as the first time contribution in the Meta Stack Exchange site by the issue reporters. As a result, the reputation of those users at the time of reporting such issues is zero. The difference of reputation between two groups of users is statistically significant with a negligible effect size (0.07).

In general, contributors of bug reports tend to have significantly higher reputation than those of feature requests, as shown in Figure 14. Answerers of the bug reports (with a median value of 11,820) have higher reputation than those of feature requests (with a median value 4,521). The statistical test shows that the difference is statistically significant and effect size is negligible (0.11). We do not observe significant difference between the reputation of users who edit bug reports and feature requests (the median values are 9,775 and 9,791, respectively). When considering the reputation of users who contribute to bug-comments and feature-comments, we observe that the two distributions are significantly different from each other ($p - value < 0.05$) with a negligible effect size (0.16). The reputation of users who contribute to answer-comments of bug reports are significantly higher than that of feature requests with a negligible effect size of 0.16.

We observe that contributors (i.e., editors, issue commenters, answerers, and answer commenters) of bug reports tend to have a higher reputation than those of feature requests. One explanation to such observation is that those edits, comments, and answers of bug reports often contain information about the cause of the bug, bug-fix deployment information, ad-hoc solution, and clarification of system design that are expected to be answered by expert users (i.e, high reputation users). In the case of feature requests, anyone can express their opinion regarding whether a feature should be implemented or not, thus no in-depth knowledge of the system is needed nor expected.

14. https://meta.stackexchange.com/questions/915

Summary of RQ1

Bug reports and feature requests differ significantly from each other along various aspects. Feature requests receive more contributions through answers, and votes (both upvotes and downvotes) compared to bug reports. Feature request threads (290 words in median) are generally longer than bug report threads (188 words in median), meanwhile feature requests receive longer comments and answers. Feature requests threads and their components also have higher sentiment scores than those for bug reports. In general, feature requests have more contributors than bug reports, while users who contribute to bug reports have higher reputation than those for feature requests. In addition, Stack Exchange users are active in contributing on issue reports. Within seven days' of submission, issue reports collected 66.2% of the answers, 84.5% of the comments, 57.4% of the votes, and 28.9% of the edits that they received throughout the studied time period.

5 RQ2: Do Q&A factors help in distinguishing bug reports from feature requests?

[Adi says: old subsection striked:]~~RQ2: Are Q&A factors important in distinguishing bug reports from feature requests in the context of Q&A based issue management systems?~~

Motivation: In RQ1, we observe that bug reports and feature requests differ in terms of various Q&A platform specific dimensions (i.e., community contributions, content characteristics, and contributing users) in the Stack Exchange issue management system. The issue management process of Stack Exchange is performed using a Q&A platform that has several unique features that are not present in traditional issue management systems [8]. For example, the rich interaction data available in such a Q&A platform (i.e., community contributions and contributing users) are not present in traditional issue reporting systems (such as Bugzilla). In addition, users are active in contributing to the management of issue reports as we observed in RQ1 in the context of Q&A based issue management systems. Therefore, we wish to verify whether Q&A platform specific features *[Adi says: improve the performance of the classifiers and]* are important in the management of issue reports when controlling other confounders (e.g., the textual content of an issue report itself). Results from our study can help designers of traditional issue management systems to decide whether Q&A platform specific factors are indeed useful and whether such factors should be incorporated in their systems.

Dimension Name	Factors	Description
Text	tf-idf_score_word	Tf-idf scores of words that are used across all the components of an issue report (Title, body, issue-comments, and answer-comments).
QAContent	ques_pos_body, **ques_pos_comment**, **ques_pos_answer**, **ques_pos_answerComment**	Positive sentiment that is expressed in the body, issue-comments, answers, and answer-comments of the issue report.
	ques_neg_body, **ques_neg_comment**, **ques_neg_answer**, **ques_neg_answerComment**	Negative sentiments expressed in the body, issue-comments, answers, and answer-comments of the issue report.
	ques_neu_body, **ques_neu_comment**, **ques_neu_answer**, **ques_neu_answerComment**	Neutral sentiments expressed in the body, issue-comments, answers, and answer-comments of the issue report.
	ques_comp_body, **ques_comp_comment**, **ques_comp_answer**, **ques_comp_answerComment**	Compound sentiments expressed in the body, issue-comments, answers, and answer-comments of the issue report.
	ques_FRE_body, **ques_FRE_comment**, **ques_FRE_answer**, **ques_FRE_answerComment**	Flesch Readability Ease score of the body, issue-comments, answers, and answer-comments of the issue report.
	ques_words_body, **ques_words_comment**, **ques_words_answer**, **ques_words_answerComment**	Number of words used to describe the body, issue-comments, answers, and answer-comments of an issue report.
	ques_img_body, ques_img_comment, ques_img_answer, ques_img_answerComment, ques_link_body, ques_link_comment, ques_link_answer, ques_link_answerComment, ques_code_body, ques_code_comment, ques_code_answer, ques_code_answerComment, ques_env_body, ques_env_comment, ques_env_answer, ques_env_answerComment	Whether the issue report body/comment/answer/answer comment has inline images, links, code or environment information (i.e, name of the operating system)
TradContent	trad_pos_body, trad_neg_body, trad_neu_body, trad_comp_body	Positive, negative, neutral, and compound sentiments of the body of the issue report.
	trad_pos_responseThread, **trad_neg_responseThread**, **trad_neu_responseThread**, trad_comp_responseThread	Positive, negative, neutral, and compound sentiments of response thread (i.e., the combination of all issue-comments, answers, and answer-comments).
	trad_FRE_responseThread, trad_FRE_body	Flesch Readability Ease for the ResponseThread and body of the issue report.
	trad_words_responseThread, trad_words_body	Number of words used to describe the ResponseThread and body of the issue report.
QAContributions	**contrib_ques_edits**, **contrib_answer_edits**	The number of edits performed on issue reports and their answers
	contrib_ques_answers	The number of answers received by the issue report.
	contrib_ques_comments, **contrib_ans_comments**	The number of comments received by the issue report and the answers.
	contrib_upvotes_ques, **contrib_downvotes_ques**	The number of upvotes and downvotes received by the issue report
	contrib_upvotes_ans, **contrib_downvotes_ans**	The number of upvotes and downvotes received by answers
QAUser	**user_editors_body**, **user_answerers**, **user_commenters**, **user_answerCommenters**	The number of editors, answerers, commenters, and answer-commenters that the issue thread has.
	user_rep_quesAsker, **user_rep_quesCommenter**, **user_rep_quesEditor**, **user_rep_answerer**,**user_rep_ansCommenter**, **user_rep_ansEditor**	Reputation of question askers, question commenters, question editors, answerers, answer commenters and answer editors.

TABLE 3: Description of the different factors across the studied dimensions. All the factors are collected after zero (i.e., at the time of the submission of an issue report), one, three, five and seven days of the submission of each issue report. Factors that are not available at the time of submission are highlighted in bold text.

Approach: To objectively identify the importance of the Q&A platform specific features in managing issue reports we consider the task of distinguishing bug reports from the feature requests. To do so, we use machine learning classifiers to examine the usefulness of factors derived from Q&A specific features in comparison with the factors derived from the traditional issue management systems for distinguishing bug reports from feature requests.

Before building the classifiers to evaluate the usefulness of Q&A specific features, we first collect factors pertaining to different dimensions of an issue management system. Some dimensions can be observed in both traditional and Q&A platform specific issue management systems (henceforth called the Q&A-backed system). On the contrary, some dimensions can be observed only in Q&A-backed systems. For instance, community contribution factors, such as downvotes, are not typically available in traditional issue management systems, whereas word count would be available on both traditional and Q&A-backed systems. We first collect the factors that are only available in Q&A-backed systems (Q&A platform specific factors), which are related to community contribution (i.e., Q&A Contribution Dimension (*QAContributions*)) and participating users (i.e., Q&A User Dimension (*QAUser*)). We next collect the factors that are related to the characteristics of the content in issue threads. Note that traditional issue management systems typically do not distinguish between comments and answers, while Q&A-based systems do. Therefore, we collect two dimensions of factors that are associated with characteristics of content: Q&A Content Characteristics Dimension (*QA-Content*) and Traditional Content Characteristics Dimension (*TradContent*). When collecting the factors of TradContent, we combine all content of comments and answers in an issue thread, while for the factors of QAContent, we collect the factor for each individual component of the issue thread. On top of these dimensions, we also collect the actual text presented in the issue reports and their associated answers and comments (i.e., Actual Text *Text*.

Actual text (i.e., *Text*) comprises all text in an issue thread (including the report title, report body, and the comments under the reports) in a traditional issue management system. Equivalently, we use the text of the title, body, issue comments, answers, and answer comments as factors in this dimension. To use these factors in our classifiers, we convert them into tf-idf vectors. We pre-process the text by removing stop words, punctuation, and other HTML tags. Next, we perform lemmatization followed by performing the stemming. We use the python package nltk[15] to remove the stop words and to perform stemming. Another python package Spacy[16] is used for lemmatization. Finally, we convert these words into tf-idf vectors using the python package scikit-learn[17].

Q&A Content Characteristics Dimension (*QAContent*) consists of 40 factors about the characteristics of the content (e.g., sentiment scores and readability scores) of each individual component of an issue thread. For instance, the ques_pos_answer records the positive sentiment score that is exhibited in the answer for a reported issue in a Q&A-backed platform. We consider all the factors that we study in Section 4.2. Table 2 lists all the factors considered in this dimension.

Traditional Content Characteristics Dimension (*Trad-Content*) comprises 12 factors about the various content characteristics of the body and the combined content (e.g., all comments and answers under an issue report) of an issue thread. For example, when collecting sentiment scores of the combined content of an issue thread, we combine all comments and answers in the issue thread to generate a ResponseThread and calculate sentiment scores for this ResponseThread, as mentioned in Table 3.

Q&A Contribution Dimension (*QAContributions*) is made up of nine factors that record the contributions to an issue report in Q&A-backed systems, in the form of activities like editing, commenting, answering, and voting as given shown in RQ1.

Q&A User Dimension (*QAUser*) is made up of 10 factors about the participating users in a Q&A-backed issue management platform. Such factors could not be observed on traditional systems, though the traditional issue management systems record the users who commented on an issue report similar to RQ1.

We present all the studied dimensions, the set of factors that make up a dimension, and their associated explanations in Table 3. We mark the features that are not available at the time of the submission of an issue reports in Table 3.

We describe the construction of classifiers and result analysis in the following paragraphs.

Classifier construction: We build two different classifiers to represent traditional and Q&A-backed issue management systems, respectively. The Q&A platform specific factors (i.e., factors related to QAContributions and QAUser dimensions) are not available in traditional issue management systems as those systems do not have separate channels (i.e., answer and comment channel) to provide a response to an issue report. Thus, we build a classifier using Text + TradContent dimensions to represent a traditional issue management system. Next, to verify the usefulness of incorporating Q&A platform specific factors, we consider Text + QAContributions + QAUser + QAContent dimensions to build a classifier representing Q&A-backed issue management systems. The studied QAContent dimension is different from the TradContent dimension because Q&A-backed issue management systems have question comment,

15. https://www.nltk.org
16. https://spacy.io
17. https://scikit-learn.org

answer, and answer comment channels to provide responses to submitted issue reports.

Moreover, to investigate the impact of the studied Q&A specific features over time, we also collected the data at different days – after one, three, five, and seven days of the submission of an issue report. We construct models using the data at different moments. In summary, we build two classifiers for each time point (one with factors available in a traditional issue management system and one with factors available in a Q&A-backed issue management system), and in total we build 10 classifiers. It should be noted that the data for some factors is not available at the time of the submission of an issue report. For example, QA contribution factors (e.g., answer count, edit count, and vote count) take time to accumulate. If the information is not available at the time of submission, we set the corresponding factor value to zero. For example, the answer count, edit count, comment count, and vote count are not available at the time of the submission of an issue report. We provide the factors available across different dimensions over time in Table 3.

When constructing classifiers, similar to the prior work of Ibrahim et al. [25], we use a composite model that consists of a two-steps process. We select a composite model to understand the importance of the textual content as a single entity instead of the importance of individual words. First, we construct a classifier on the textual content (i.e., *Text*) of the issue report to determine the relevance of that report to the issue report category. For each of the studied issue report the classifier provides the probability of the given issue report being a bug report based on the textual context of the issue report. Second, we combine the factors from other dimensions (i.e., QAContent, TradContent, QAContributions, and QAUser) with the probability score obtained for each issue report using the first classifier. The combined metrics are used as the input to the second classifier to determine the issue report category (i.e., either a bug report or a feature request).

We use random forest classifiers to study the usefulness of Q&A platform specific features because of their robust nature, and their ability to handle categorical and continuous explanatory factors [21]. We divide the dataset into training and test categories. In the training phase of the first step, we take the issue reports as input, divide the textual content into words, and calculate the tf-idf scores of those words for each issue report. We use these scores as input to the random forest classifier to determine the probability of the issue report being a bug report. The closer the value is to 1, the more likely that the issue report discusses a bug. We use the classifier that is constructed in the training phase to assign such probability scores for issue reports available in testing set too. Please note that we assign probability scores to issue reports of testing set using only with the classifier constructed on the training data thereby ensuring we do not look at the test labels. The second step uses another random forest classifier to determine the issue report category (either a bug report or a feature request). The classifier takes as input the probability values we obtain from the first step together with other factors from the remaining four dimensions as shown in Table 3.

Before building our classifiers, we remove the correlated and redundant factors using the R package AutoSpearman [26] as prior studies show that the presence of correlated and redundant features affect the interpretation of a classifier [36], [46]–[48]. The removed factors are highlighted in Table 3. Next, we train classifiers on the specified dimensions to distinguish between bug reports and feature requests. We also note down the performance (in terms of Area Under the Receiver-Operator Characteristic Curve (AUC)) and the factor importance scores following prior studies [40], [41]. Note that when building classifiers we did not apply any class rebalancing techniques to address the data imbalance issue since class rebalancing techniques would bias the interpretation of results and should be avoided when interpreting models [45].

To ensure the statistical robustness of the insights derived from our constructed classifiers, we use an out-of-sample bootstrap approach that relies on random sampling with replacement. The process splits the dataset into training and test datasets. The training dataset is created by sampling the whole dataset with replacement. Therefore the number of data points in the train datasets is the same as that of the original dataset (however some of the data points are repeated due to resampling). The test dataset is made up of data points that were not sampled during the creation of the training dataset. Typically, on average the test set consists of 36.8% of the data points found in the whole dataset [17]. We repeat the aforementioned process 100 times as the prior studies suggest [41], [49]. In each of the bootstrap iterations, the performance metrics and the factor importance scores are noted. For each iteration, a classifier is trained on data that is re-sampled with replacement (train dataset), then the classifier is tested using out-of-bag data points (test dataset i.e., data points that weren't used in the training of the classifier). We use the python package scikit-learn for building the Random Forest classifier [10], and set the number of trees to be 500. Finally, we use the permutation importance technique [10] that is available in the python package scikit-learn to generate the factor importance scores. We use the same setting across all different classifiers to avoid any biases.

Performance evaluation: We use the median AUC score to evaluate the performance of a classifier as prior studies [40]. We use the signed-rank test [53] to test if the 100 AUC scores that are generated by the different classifiers are significantly different. An AUC of 0.5 indicates random guessing and 1 indicates the perfect prediction. Furthermore, we used Cohen's d [14] to measure the effect size of the difference.

Dimension Used	Time	AUC
Text + TradContent	D_0	80.4
Text + QAContributions + QAUser + QAContent	D_0	83.2
Text+TradContent	D_1	83.7
Text + QAContributions + QAUser + QAContent	D_1	87.8
Text + TradContent	D_3	84.4
Text + QAContributions + QAUser + QAContent	D_3	88.8
Text + TradContent	D_5	84.6
Text + QAContributions + QAUser + QAContent	D_5	88.9
Text + TradContent	D_7	84.6
Text + QAContributions + QAUser + QAContent	$D7$	88.7

TABLE 4: The performance of our random forest classifier using factors from different combination of dimensions. We use 100-out-of-sample bootstrap in our evaluation and take the median of the result. The time D_0 refers that the factors are collected at the time of the submission of a bug report or a feature request. On the other hand, D_1, D_3, D_5, and D_7 refers that factors are collected after one, three, five and seven days of the submission of a bug report or a feature request, respectively.

Factor importance evaluation: Finally, to ascertain the factors that are influential in identifying bug reports from feature requests, we compute the feature importance ranks. We do so by taking the 100 feature importance scores that are generated by the permutation importance technique [10]. We then use the Scott-KnottESD [49] to generate the feature importance ranks similar to [41], [51].

Results: *[Adi says: isn't the old heading of this para same as the heading of the next para?]* ~~Q&A platform specific features are important in distinguishing bug reports from feature requests.~~ *[Adi says: new heading:]* **The addition Q&A platform specific features better classify bug reports from feature requests as compared to classifiers built on traditional features. The classifier constructed on factors obtained from Q&A platform specific features to distinguish bug reports from feature requests have a median AUC of 87.8%. The above value is higher compared to the AUC of the classifier constructed on factors obtained from traditional issue management systems**. When considering the factors that are available at the time of the submission of the issue reports (D_0 as shown in Table 4), the baseline model (i.e., Text + TradContent) constructed on factors obtained from traditional issue management systems identifies bug reports from feature requests with a median AUC of 80.4% *[Adi says: , 7.4% lower than that of Q&A features.]* When combining all Q&A platform specific factors with the textual content of issue reports (i.e., Text + QAContributions + QAUser + QAContent), the AUC score increases to 83.2%. Considering the factors after only one day of the submission of issue reports (D_1), the baseline model obtains a median AUC of 83.7%. However, combining all the Q&A platform specific factors with the textual content increases the median AUC to 87.8%. The classifier significantly outperforms (p-value<0.05) the baseline classifier with a large effect size of 3.8. This shows the importance of using Q&A platform specific factors in distinguishing bug reports from feature requests, especially when more interaction data are available. The high AUC value also indicates that our classifier explains the data well. Moreover, such a result highlights the value of the rich interactions around issue reports on a Q&A platform to distinguish bug reports from feature requests. Our observation is consistent across the classifiers built using data of other moments.

Q&A platform specific factors (e.g., the reputation of contributors) are important in distinguishing bug reports from feature requests. The factor pertaining to the content characteristics (i.e., text) becomes the most important factor, showing the importance of considering the textual content of issue reports for their management. However, Q&A platform specific factors are also identified as important ones. Table 5 shows the factors that are ranked as the most important factor after the textual content of issue reports in the classifiers that combine Q&A platform specific factors with the textual content of issue reports (i.e., Text + QAContributions + QAUser + QAContent). For instance, the reputation of contributors is an important factor for distinguishing bug reports from feature requests across all the classifiers that are built using data of different moments. The user_rep_answerer is identified as the most important factor in all classifiers as shown in Table 5. Other factors that are related to the reputation of contributors, such as user_rep_answerer, user_rep_quesEditor, user_rep_ansCommenter, and user_rep_quesCommenter, are ranked as the second most important factor at the moment of D_1. The above factors are also related to separating replies to three different channels. Without such a separation, we can neither distinguish answerers from commenters nor calculate the reputation of those users. In accordance with our findings in RQ1, votes on issue reports are also identified as important factors at the moment of D_1 and D_3. Both reputation and vote count are Q&A platform specific features that are not available in the traditional issue management systems. Thus, Q&A platform specific features are important in issue report management, such as distinguishing bug reports from feature requests. Traditional issue management systems may wish to integrate such features in their systems.

Time	Factors
Day-0	user_rep_quesAsker, ques_words_body, ques_FRE_body, ques_comp_body
Day-1	ques_neu_body, ques_comp_body, user_rep_answerer, ques_words, user_rep_quesEditor, ques_upvote_-count, user_rep_ansCommenter, user_rep_quesCommenter, ques_comment_words, user_editors_body
Day-3	ques_senti_neu, user_rep_answerer, user_rep_ques_editor, ques_words_body, contrib_ques_upvotes, ques_senti_com, contrib_ans_downvotes
Day-5	ques_comp_body, user_rep_answerer, user_rep_quesEditor, ques_words, ques_FRE_comment, ques_-comment_words, user_rep_quesCommenter, ques_FRE_body, ques_neu_body
Day-7	user_rep_answerer, ques_comp_body

TABLE 5: The factors that are ranked as the second most important factors in the classifiers that integrating Q&A platform specific features. Note that the textual content of issue reports (i.e., text) identified as the most important factor (i.e., ranked first) in all classifiers, therefore we do not show here.

Summary of RQ2

Q&A platform specific factors (i.e., reputation of users and vote count) *[Adi says: help in better classifying bug reports and feature requests by improving the AUC of the predictive models. Moreover,]* Q&A features are important in distinguishing bug reports from feature requests. The random forest classifiers obtain a median AUC of 83.2% when the classifiers are constructed on factors obtained from traditional issue management systems. However, combining Q&A platform specific factors with the textual content increases the AUC to 87.8%. In both cases, factors are collected after one day of the submission of issue reports. Our observation is consistent even after collecting the factors after three, five, and seven days of the submission of issue reports.

6 Implications of our Findings

[Shaowei says: updated] **Traditional issue management systems should consider integrating features of Q&A platform based management systems which are missing in them (e.g., reputation, in-place editing and voting).** Results from RQ1 shows that Stack Exchange users are active in managing issue reports. Within seven days of submission, issue reports collected 66.2% of the answers, 84.5% of the comments, 57.4% of the votes, and 52.2% of the edits that they received throughout the studied time period. Our comparison of the management of issue reports between Stack Exchange and GitHub shows that some QA features are currently not available in GitHub. We also observe that bug reports and feature requests differ significantly from each other along various aspects. In RQ2, we observe that such Q&A platform specific factors can help improve in distinguishing bug reports from feature requests. Therefore, we encourage traditional issue management systems, in which such features are not available yet (as shown in Section 2.1), should consider integrating such Q&A platform specific features in them. Especially, in the future as traditional issue management systems' (e.g., GitHub) Q&A capability evolves to a position where Stack Exchange is at, then our approach could be leveraged.

Our study could facilitate software engineering research beyond distinguishing bug reports from feature requests. First, Stack Exchange's reputation mechanism can increase users' participation in managing issue reports (i.e., correcting incorrect tagging). Although we observed cases where bug reports were submitted as feature requests or vice versa, due to the active participation of community members those misclassifications were corrected within a median of 2.84 hours of submitting an issue report. Thus, integrating the reputation mechanism in the traditional issue reporting system can enable to fix tagging issues by the users. Second, prior studies show that the misclassification of bug reports can significantly impact the performance of bug prediction models. For instance, before building a bug prediction model to classify issues in a future version of a software, one could clean up the data available in issue management systems that integrate Q&A platform specific features, which would in turn help avoid bias in bug prediction models and improve their performance by ensuring the removal of noise (i.e., feature requests from actual bug reports in this case). Separating true bug reports from feature requests can facilitate research that leverages bug reports for software maintenance (e.g., developer recommendation [2], change impact analysis [19], and defect localization [30]). Third, feature location techniques [42] focus on finding source code that implements a specific functionality in a software system. Incorrect classification of issue reports can affect the performance of feature location techniques. In this regard, large well-labelled issue reports can act as a gold standard in improving the performance of feature localization and traceability techniques. Our study can be leveraged to improve the management of issue reports (i.e., discriminating bug reports from issue requests) and in turn improving the benchmark for feature localization and traceability techniques.

7 Threats to Validity

External validity: Threats to external validity are related to the generalizability of our findings. In this study, we performed a comparison between bug reports and feature requests. Our results may not generalize to other datasets. However, the leveraging of a Q&A platform for bug management is quite rare to the prevalence of traditional bug management systems. Our studied dataset contains a large number of high-quality issue reports that have been tagged as bug reports versus feature requests by Stack Exchange developers and community members, rather than researchers (ensuring the robustness of our findings). Thus, our studied dataset provides a rare opportunity to understand the differences between bug reports and feature requests, and to evaluate the effectiveness of using Q&A specific factors in classifying bug reports and feature requests.

Internal validity: Our studied dataset contains a large number of issue reports that have been tagged as bug reports versus feature requests by Stack Exchange developers and community members, rather than researchers. We cannot guarantee that all issues in the Stack Exchange issue management system are tagged carefully and some tagging errors may exist. However, due to the active participation of community members such errors are likely to be low (expected to be much lower than traditional issue management systems that do not incentivise users to participate in community activities, such fixing tags in issue reports) and such errors are very likely to be corrected. We manually examined 50 such re-tagged issue reports and they were tagged correctly eventually.

Stack Exchange issue management system uses a Q&A platform for issue management. To make a comparison between the issue management system of Stack Exchange and traditional issue management system, we analyze issue threads by grouping answers and comments. Such a grouping is done because we are not aware of any project similar to Stack Exchange that leverages a traditional issue reporting system. Thus, the results we obtain may not be truly representative. We suggest that future research should address this issue further when a similar web-application project with similar scale as of Stack Exchange is available in the traditional issue reporting systems (such as Bugzilla) or traditional issue reporting systems integrate the features available in Stack Exchange (i.e., in-place editing feature, voting, different channels for providing responses (answers and comments) and the reputation mechanism to encourage users to participate in community activities).

8 Related Work

This section discusses prior studies related to our work.

8.1 Automatic classification of Bug Reports and Feature Requests

To automatically discriminate bug reports from feature requests, various techniques have been proposed using text-based features [1], [18], [33], [39], [56]. For instance, Antonoil et al. used text-based features to identify bug reports from feature requests. To do so, the authors compared logistic regression and naive Bayes classifiers against decision trees [1]. Our study differs from these prior studies on two important aspects. First, the goal of our study is not to improve issue classification rather to determine the importance of Q&A platform specific features in managing issue reports that are missing in traditional issue reporting systems. To objectively identify the importance of the Q&A platform specific features in managing issue reports we consider the task of distinguishing bug reports from the feature requests. None of the prior studies has ever investigated a Q&A platform specific issue management system or use Q&A platform specific factors to distinguish bug reports from feature requests. Second, all the prior studies use small datasets that are tagged by researchers or software engineering, none of them performed a study on a large number of issue reports that are tagged by domain experts. For example, Antoniol et al. used 1,800 issues extracted from the issue reporting systems of Mozilla, Eclipse, and JBoss where the issues were manually tagged by five software engineers [1]. Herzig et al. manually classified 7,401 issues reported as bugs from the bug databases of five different projects [22]. However, in this study, we use a bigger dataset of 9,048 issue reports which have been curated by the domain experts. Table 6 summarizes the studies where issue reports are manually validated by the researchers or software engineers instead of the domain experts (i.e., developers).

8.2 Studying and Improving the Bug Reporting Process

Several studies have been conducted to understand the challenges of 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 [27]. 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.

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

Study Name	Subject Systems	Dataset Size	Tagger Type
Antoniol et al. [1]	Mozilla, Eclipse, and JBoss	1,800	Software engineers
Herzig et al. [22]	HTTPClient, Jackrabbit, Lucene-Java, Rhino, Tomcat5	7401	Researchers
Malej and Nabil [33]	Apple app store and Google play store	4,400	Researchers
Zhou et al. [56]	Mozilla, Eclipse, JBoss, Firefox and OpenFOAM	3,200	Researchers
Zanaty et al. [18]	Shopify Core, Partners and Startscream	951	Researchers
Our Study	Stack Exchange	9,048	Developers and experienced users

TABLE 6: Examples of issue report datasets where the categories of those issue reports are manually validated.

that impacts its resolution time and they found that some core elements (e.g., screenshot and the stack trace) of a bug report do impact the resolution time [15]. Hooimeijer et al. modeled the quality of a bug report by analyzing their features [23]. Surprisingly, the self-reported severity of a bug report is not a reliable indicator of the importance of the bug. To improve bug descriptions quality by alerting reporters about the missing information when reporting a 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 [13]. Rejaul et al. proposed an approach to predict the missed key features (e.g., expected behavior) in a bug report [43]. 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 [50].

While prior studies focused on improving bugs by studying which types of information are important and how to detect those information automatically, we focus on investigating the factors that could be used to help identify bug reports and feature requests.

8.3 Leveraging Crowdsourcing to Support Bug Management Activities

Crowdsourcing could be leveraged to improve bug management, such as bug reporting, bug triaging, and bug fixing. Breu et al. observed that interacting with developers provides help in fixing bugs in term of shortening the resolution time [11]. 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 [55]. 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 [32]. Different from prior studies which leveraged crowdsourcing to support bug management activities, we investigate how to better distinguish bug reports and feature requests by using the features of Q&A platforms.

9 Conclusion

[Adi says: need to update! citations added here. write text.] [7] [38] [5] In this paper, we study the issue reports of Stack Exchange to analyze the differences between bug reports and feature requests along various dimensions (i.e., community contributions, content characteristics, and participating users). We find that bug reports and feature requests are significantly different across various dimensions and they are managed differently in the Stack Exchange issue management system. For instance, in general, feature requests have more contributors than bug reports, while users who report and contribute to bug reports have higher reputation than those for feature requests. Results from our study shows that Q&A platform-specific factors (such as the reputation, voting, and different channels for providing response to an issue report) *[Adi says: help better classification of issue reports and]* are important in distinguishing bug reports from feature requests. Traditional issue management systems, such as GitHub, may wish to consider integrating such features for better issue management (e.g., distinguishing bug reports and feature requests).

References

[1] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "Is it a bug or an enhancement?: A text-based approach to classify change requests," in *Proc. of the Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds*, 2008, pp. 304–318.

[2] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *Proc. of the 28th International Conference on Software Engineering*, 2006, pp. 361–370.

[3] 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.

[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] S. G. Baker, E. Schuit, E. W. Steyerberg, M. J. Pencina, A. Vickers, K. G. Moons, B. W. Mol, and K. S. Lindeman, "How to interpret a small increase in auc with an additional risk prediction marker: decision analysis comes through," *Statistics in medicine*, vol. 33, no. 22, pp. 3946–3959, 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] A. Bhatia, S. Wang, M. Asaduzzaman, and A. E. Hassan, "A study of bug management using the stack exchange question and answering platform," *IEEE Transactions on Software Engineering*, pp. 1–1, 2020.

[8] A. Bhatia, S. Wang, M. Asaduzzaman, and A. E. Hassan, "A study of bug management using the stack exchange question and answering platform," *IEEE Transactions on Software Engineering*, 2020.

[9] P. Bhattacharya and I. Neamtiu, "Bug-fix time prediction models: can we do better?" in *Proc. of the 8th Working Conference on Mining Software Repositories*, 2011, pp. 207–210.

[10] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, p. 5–32, Oct. 2001.

[11] 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.

[12] R. P. Buse and W. R. Weimer, "A metric for software readability," in *Proc. of the International Symposium on Software Testing and Analysis*, 2008, pp. 121–130.

[13] 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.

[14] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 1988.

[15] 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.

[16] M. D'Ambros, M. Lanza, and R. Robbes, "An extensive comparison of bug prediction approaches," in *Proc. of the 7th IEEE Working Conference on Mining Software Repositories*, 2010, pp. 31–41.

[17] B. Efron, "Estimating the error rate of a prediction rule: Improvement on cross-validation," *Journal of the American Statistical Association*, vol. 78, no. 382, pp. 316–331, 1983.

[18] F. El Zanaty, C. Rezk, S. Lijbrink, W. van Bergen, M. Côté, and S. McIntosh, "Automatic recovery of missing issue type labels," *IEEE Software*, pp. 0–0, 2020.

[19] M. Gethers, H. Kagdi, B. Dit, and D. Poshyvanyk, "An adaptive approach to impact analysis from change requests to source code," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, 2011, pp. 540–543.

[20] E. Giger, M. Pinzger, and H. Gall, "Predicting the fix time of bugs," in *Proc. of the 2Nd International Workshop on Recommendation Systems for Software Engineering*, 2010, pp. 52–56.

[21] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust prediction of fault-proneness by random forests," in *Proc. of the 15th International Symposium on Software Reliability Engineering*, 2004, pp. 417–428.

[22] K. Herzig, S. Just, and A. Zeller, "It's not a bug, it's a feature: How misclassification impacts bug prediction," in *Proc. of the International Conference on Software Engineering*, 2013, pp. 392–401.

[23] 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.

[24] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proc. of the Eighth International AAAI Conference on Weblogs and Social Media*, 2014, pp. 216–225.

[25] W. M. Ibrahim, N. Bettenburg, E. Shihab, B. Adams, and A. E. Hassan, "Should i contribute to this discussion?" in *Proc. of the 7th IEEE Working Conference on Mining Software Repositories*, 2010, pp. 181–190.

[26] J. Jiarpakdee, C. Tantithamthavorn, and C. Treude, "Autospearman: Automatically mitigating correlated software metrics for interpreting defect models," in *Proc. of the IEEE International Conference on Software Maintenance and Evolution*, 2018, pp. 92–103.

[27] 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.

[28] Y. Kamei, S. Matsumoto, A. Monden, K.-i. Matsumoto, B. Adams, and A. E. Hassan, "Revisiting common bug prediction findings using effort-aware models," in *Proc. of the IEEE International Conference on Software Maintenance*, 2010, pp. 1–10.

[29] K. Kelley and K. J. Preacher, "On effect size," *Psychological methods*, vol. 17, no. 2, p. 137, 2012.

[30] P. S. Kochhar, T.-D. B. Le, and D. Lo, "It's not a bug, it's a feature: Does misclassification affect bug localization?" in *Proc. of the 11th Working Conference on Mining Software Repositories*, 2014, p. 296–299.

[31] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 485–496, 2008.

[32] 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.

[33] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *Proc. of the 23rd International Conference on Requirements Engineering*, 2015, pp. 116–125.

[34] S. Mani, A. Sankaran, and R. Aralikatte, "Deeptriage: Exploring the effectiveness of deep learning for bug triaging," in *Proc. of the ACM India Joint International Conference on Data Science and Management of Data*, 2019, pp. 171–179.

[35] L. Marks, Y. Zou, and A. E. Hassan, "Studying the fix-time for bugs in large open source projects," in *Proc. of the 7th International Conference on Predictive Models in Software Engineering*, 2011, pp. 1–11.

[36] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "An empirical study of the impact of modern code review practices on software quality," *Empirical Software Engineering*, vol. 21, no. 5, pp. 2146–2189, 2016.

[37] T. T. Nguyen, A. T. Nguyen, and T. N. Nguyen, "Topic-based, time-aware bug assignment," *SIGSOFT Softw. Eng. Notes*, vol. 39, no. 1, pp. 1–4, 2014.

[38] M. Ohira, A. E. Hassan, N. Osawa, and K.-i. Matsumoto, "The impact of bug management patterns on bug fixing: A case study of eclipse projects," in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 2012, pp. 264–273.

[39] N. Pingclasai, H. Hata, and K.-i. Matsumoto, "Classifying bug reports to bugs and other requests using topic modeling," in *Proc. of the 20th Asia-Pacific Software Engineering Conference*, vol. 2, 2013, pp. 13–18.

[40] G. K. Rajbahadur, S. Wang, Y. Kamei, and A. E. Hassan, "The impact of using regression models to build defect classifiers," in *Proc. of the IEEE/ACM 14th International Conference on Mining Software Repositories*, 2017, pp. 135–145.

[41] ——, "Impact of discretization noise of the dependent variable on machine learning classifiers in software engineering," *IEEE Transactions on Software Engineering*, 2019.

[42] A. Razzaq, A. Wasala, C. Exton, and J. Buckley, "The state of empirical evaluation in static feature location," *ACM Trans. Softw. Eng. Methodol.*, vol. 28, no. 1, Dec. 2018.

[43] 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.

[44] E. Shihab, A. Ihara, Y. Kamei, W. M. Ibrahim, M. Ohira, B. Adams, A. E. Hassan, and K. Matsumoto, "Studying re-opened bugs in open source software," *Empirical Software Engineering*, vol. 18, no. 5, pp. 1005–1042, 2013.

[45] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The impact of class rebalancing techniques on the performance and interpretation of defect prediction models," *IEEE Transactions on Software Engineering*, pp. 1–1, 2018.

[46] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, A. Ihara, and K. Matsumoto, "The impact of mislabelling on the performance and interpretation of defect prediction models," in *Proc. of the IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, 2015, pp. 812–823.

[47] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "Comments on "researcher bias: The use of machine learning in software defect prediction"," *IEEE Transactions on Software Engineering*, vol. 42, no. 11, pp. 1092–1094, 2016.

[48] C. Tantithamthavorn and A. E. Hassan, "An experience report on defect modelling in practice: Pitfalls and challenges," in *Proc. of the 40th International Conference on Software Engineering: Software Engineering in Practice*, 2018, pp. 286–295.

[49] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering*, no. 1, 2017.

[50] 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.

[51] Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan, "What are the characteristics of high-rated apps? a case study on free android applications," in *Proc. of the IEEE International Conference on Software Maintenance and Evolution*, 2015, pp. 301–310.

[52] C. Weiss, R. Premraj, T. Zimmermann, and A. Zeller, "How long will it take to fix this bug?" in *Proc. of the Fourth International Workshop on Mining Software Repositories*, 2007, pp. 1–1.

[53] F. Wilcoxon, "Individual comparisons by ranking methods," vol. 1, no. 6, 1945, pp. 80–83.

[54] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in *Proc. of the 34th International Conference on Software Engineering*, 2012, pp. 25–35.

[55] 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.

[56] Y. Zhou, Y. Tong, R. Gu, and H. Gall, "Combining text mining and data mining for bug report classification," in *Proc. of the IEEE International Conference on Software Maintenance and Evolution*, 2014, pp. 311–320.

Appendix A
Performance of different classifiers

In our study, we use the random forest classifier to understand the factors that are important in distinguishing bug reports from the feature requests. However, other classifiers can also be used. Thus, we investigate the use of Q&A platform specific features in potentially distinguishing bug reports and feature requests in other classifiers. We select Decision Tree, Naive Bayes, and Logistic Regression from different families of classifiers that have been used in many prior studies *[Adi says: in bug management]* [20], [44]. For all four classifiers, we also report the median precision, recall and F-measure values. We choose a threshold of 0.5 to compute these metrics.

[Adi says: Q&A features outperform the traditional features for all four classifiers.] Table 7 shows the results for all four classifiers. The AUC measure improves when we collect the factors after one or more days of the submission of issue reports compared to collecting the factors at the time of the submission of issue reports. When we collect the Q&A platform specific factors along with the textual content after one day of the submission of issue reports, the *[Adi says: logistic regression]* classifier outperforms all the other classifiers. The decision tree classifier obtains the second position with a median AUC of 75.8%, and the random forest classifier obtains the third position with a median AUC of 72.0%. However, there is not much difference in the AUC measure when we collect the factors after seven days of the submission of issue reports. Our study results are not consistent for classifiers in terms of precision, recall and F-measure values. This perhaps because our dataset is imbalanced. Although we include the results of different metrics for the sake of completeness, the AUC measure is the most appropriate to consider for our study. This is because AUC is insensitive to cost and class distributions so that the data imbalance issue of our dataset is automatically accounted for and provides a score that is objective [31], [40]. *[Adi says: we have not provided statisticcal analysis (wilcox + cliffs delta) in any of the comparisons here so I did not provide for the LRM model updated values.]*

B Factors Obtained Through Correlation and Redundancy Analysis

Table 8 shows the list of factors we obtained after performing correlation and redundancy analysis. Here, Text + QAContributions + QAUser + QAContent dimensions are used to build classifiers representing a Q&A-backed issue management system. Text + TradContent dimensions are used to build classifiers representing a traditional issue management system.

Dimension	Time	Random Forest							Decision Tree						
		AUC	PBR	RBR	FBR	PFR	RFR	FFR	AUC	PBR	RBR	FBR	PFR	RFR	FFR
Text + TradContent	D_0	80.4	86.6	95.5	90.8	83.9	61.3	70.8	71.2	84.0	84.4	84.2	58.7	58.3	58.4
Text + QAContributions + QAUser + QAContent	D_0	83.2	86.7	95.5	90.8	84.2	61.3	71.0	71.2	84.0	84.4	84.2	58.7	58.7	58.4
Text + TradContent	D_1	83.8	87.3	97.3	92.0	89.6	62.7	73.8	75.7	86.6	86.6	86.6	64.8	64.8	65.0
Text + QAContributions + QAUser + QAContent	D_1	87.8	87.1	97.2	91.9	89.5	61.9	73.4	75.7	86.6	86.6	86.7	65.2	64.8	65.0
Text + TradContent	D_3	84.4	87.8	97.2	92.2	89.8	64.3	74.9	76.0	86.7	86.6	86.7	65.3	65.2	65.3
Text + QAContributions + QAUser + QAContent	D_3	88.7	87.5	97.2	92.1	89.7	63.6	74.6	76.0	86.7	86.6	86.7	65.3	65.2	65.3
Text + sTradContent	D_5	84.5	87.8	97.2	92.2	89.8	64.4	74.1	76.2	86.8	87.0	86.9	65.9	65.6	65.8
Text + QAContributions + QAUser + QAContent	D_5	88.9	87.6	97.2	92.2	89.8	64.1	74.8	76.2	86.8	87.0	86.9	65.9	65.6	65.8
Text + TradContent	D_7	84.5	87.8	97.4	92.3	90.2	64.5	75.1	76.4	86.9	87.1	87.0	65.8	65.8	65.7
Text + QAContributions + QAUser + QAContent	D_7	88.7	87.7	97.3	92.2	90.0	64.2	74.8	76.4	86.9	87.1	87.0	65.8	65.8	65.7
Dimension	**Time**	**Naive Bayes**							**Logistic Regression**						
		AUC	**PBR**	**RBR**	**FBR**	**PFR**	**RFR**	**FFR**	**AUC**	**PBR**	**RBR**	**FBR**	**PFR**	**RFR**	**FFR**
Text + TradContent	D_0	57.1	79.1	46.1	58.2	32.7	68.1	44.1	92.3	89.7	93.2	91.5	80.2	72.0	75.9
Text + QAContributions + QAUser + QAContent	D_0	68.7	79.3	85.3	82.2	51.9	41.9	46.4	92.6	90.4	92.2	91.3	78.9	74.8	76.8
Text + TradContent	D_1	62.5	79.7	59.7	68.2	36.4	60.4	45.5	94.7	91.4	94.4	92.9	84.1	77.0	80.4
Text + QAContributions + QAUser + QAContent	D_1	72.1	77.5	90.8	83.5	56.0	30.8	40.3	94.7	91.2	94.8	92.9	84.9	76.2	80.2
Text + TradContent	D_3	62.2	79.6	59.7	68.3	36.3	59.8	45.2	95.0	91.7	94.6	93.1	84.3	77.6	80.9
Text + QAContributions + QAUser + QAContent	D_3	72.2	77.5	91.1	83.6	56.6	30.6	40.0	95.1	91.3	94.9	93.1	85.2	76.6	80.6
Text + TradContent	D_5	62.4	79.4	59.5	68.1	36.3	59.9	45.3	95.1	91.9	94.5	93.2	84.3	78.3	81.4
Text + QAContributions + QAUser + QAContent	D_5	72.6	77.4	91.5	83.9	57.9	30.8	40.0	95.2	91.5	95.1	93.3	85.7	76.9	81.3
Text + TradContent	D_7	62.5	79.6	59.1	67.9	36.1	60.3	45.3	95.1	91.9	94.5	93.3	84.5	78.3	81.3
Text + QAContributions + QAUser + QAContent	D_7	73.0	77.5	91.9	84.0	58.3	29.5	39.3	95.2	90.6	95.0	93.3	85.5	97.2	81.3

TABLE 7: The performance of all four classifiers using factors from different combination of dimensions. Here, PBR, RBR, and FBR refer to precision, recall and F-measures for the bug report category, respectively. PFR, RFR, and FFR refer to precision, recall, and F-measure for the feature request category. We use 100-out-of-sample bootstrap in our evaluation. For the AUC, we take the average of the result and for the other metrics we report the median of the result. The time D_0 refers that the factors are collected at the time of asking ~~a bug report or a feature request~~ an issue request. On the other hand, D_1, D_3, D_5, and D_7 refers that factors are collected after one, three, five and seven days of submitting a bug report or a feature request, respectively.

Time	Factors (Text + QAContributions + QAUser + QAContent)	Factors (Text + TradContent)
D_0	text signal, ques_FRE_body, ques_neu_body, ques_com_body, ques_code_body, ques_link_body, ques_img_body, ques_env_body, ques_words_body	text signal
D_1	text signal, ques_FRE_body, ques_FRE_comment, ques_neg_comment, ques_neu_-body, ques_neu_comment, ques_comp_body, ques_comp_commen, ques_code_-body, ques_link_body, ques_img_body, ques_env_body, ques_code_comment, ques_link_comment, ques_img_comment, ques_env_comment, ques_words_body, ques_words_comment, contrib_ques_comments, contrib_upvotes_ques, contrib_-downvotes_ques, contrib_downvotes_ans, user_rep_answerer, user_rep_quesCom-menter, user_rep_ansCommenter, user_editors_body, user_rep_quesEditor	text signal, trad_FRE_responseThread, trad_pos_responseThread, trad_neg_re-sponseThread, trad_neu_responseThread, trad_comp_responseThread, trad_words_-responseThread
D_3	text signal, ques_FRE_body, ques_FRE_comment, ques_neg_comment, ques_neu_-body, ques_neu_comment, ques_comp_body, ques_comp_commen, ques_code_-body, ques_link_body, ques_img_body, ques_env_body, ques_code_comment, ques_link_comment, ques_img_comment, ques_env_comment, ques_words_body, ques_words_comment, contrib_ques_comments, contrib_upvotes_ques, contrib_-downvotes_ques, contrib_downvotes_ans, user_rep_answerer, user_rep_quesCom-menter, user_rep_ansCommenter, user_editors_body, user_rep_quesEditor	text signal, trad_FRE_responseThread, trad_pos_responseThread, trad_neg_re-sponseThread, trad_neu_responseThread, trad_comp_responseThread, trad_words_-responseThread
D_5	text signal, ques_FRE_body, ques_FRE_comment, ques_neg_comment, ques_neu_-body, ques_neu_comment, ques_comp_body, ques_comp_commen, ques_code_-body, ques_link_body, ques_img_body, ques_env_body, ques_code_comment, ques_link_comment, ques_img_comment, ques_env_comment, ques_words_body, ques_words_comment, contrib_ques_comments, contrib_upvotes_ques, contrib_-downvotes_ques, contrib_downvotes_ans, user_rep_answerer, user_rep_quesCom-menter, user_rep_ansCommenter, user_editors_body, user_rep_quesEditor	text signal, trad_FRE_responseThread, trad_pos_responseThread, trad_neg_re-sponseThread, trad_neu_responseThread, trad_comp_responseThread, trad_words_-responseThread
D_7	text signal, ques_FRE_body, ques_FRE_comment, ques_neg_comment, ques_neu_-body, ques_neu_comment, ques_comp_body, ques_comp_commen, ques_code_-body, ques_link_body, ques_img_body, ques_env_body, ques_code_comment, ques_link_comment, ques_img_comment, ques_env_comment, ques_words_body, ques_words_comment, contrib_ques_comments, contrib_upvotes_ques, contrib_-downvotes_ques, contrib_downvotes_ans, user_rep_answerer, user_rep_quesCom-menter, user_rep_ansCommenter, user_editors_body, user_rep_quesEditor	text signal, trad_FRE_responseThread, trad_pos_responseThread, trad_neg_re-sponseThread, trad_neu_responseThread, trad_comp_responseThread, trad_words_-responseThread

TABLE 8: Factors obtained after performing correlation and redundancy analysis.