Dear Sir/Madam,

We have carefully read the report of the editors and the reviewers regarding our article titled “Deriving a Usage-Independent Software Quality Metric”, and have extensively revised the paper to address the points they have raised.

We would like to thank the editors and the reviewers for their comments. We think the manuscript has improved greatly as a result of the reviewers’ constructive criticisms. We hope our latest response is satisfactory to you.

With this letter, we provide a document containing sentences form each report (*in italic*) that *pose specific questions or criticize our work*. Below these, we have added our own comments, and describe how each have been addressed in the paper. The annotations (e.g. S-3.3, pg 21) indicates that, e.g. section 3.3 on page 21 has been modified or extended.

Since the paper has been reorganized and extensive changes were made, we start by providing a summary of the updates to the paper, and then address the specific questions by the reviewers.

**Summary of update:**

* We have added three specific research questions that our paper it trying to answer, in accordance with the suggestion of Reviewer 1.
* The Sections of the paper have been reorganized as per the suggestion of Reviewer 1. A new “**Motivation**” section (S2) has been added. All sections related to data description and preprocessing have been consolidated into one section titled “**Data Description**” (S3). A “**Methodological Overview**” section (S4) has been added that discuss the methodologies used in the paper. Separate subsections have been added in the “**Results**” section (S5) that address each research question. The section “Implication of our Findings” has been repurposed into “**Discussion**” section (S6). The “Comparison with published results” section has been made into a subsection under the “**Related Works**” section (S7). The “**Limitations**” (S8) and “**Conclusion**” (S9) sections have also been modified according to the reviews.
* An Appendix has been provided with the article that discusses the relationship between exceptions and a code complexity measure (Lines of Code - LOC). We added this as an Appendix, since we do not believe this analysis is essential for addressing the research questions we posed.
* We have added timeline plots showing the difference in trends for the mobile applications when the number of exceptions is directly used as a predictor vs. when it is normalized by the number of users. Similar plots have been added for 4 popular NPM packages.

**Editor Comments**

*Thank you for your submission to EMSE. Based on the reviewers' feedback, we recommend "Major Revision" for your manuscript.*

*All the reviewers agree that the paper is generally easy-to-read, it deals with a relevant topic for the readers of this journal, and the difference from the previously published PROMISE paper is satisfying.*

*However, they also highlight that the paper needs to be improved regarding the following issues:*

*a) threats to the internal, construct and conclusion validity of the findings (all reviewers), b) missing control for potential confounding factors, and unclear implications of the findings (Reviewer 1), c) small/limited sample size (Reviewers 2 and 3), d) discussion on other internal quality variables (Reviewer 2), e) validity of applied techniques and tests --tests for normality, default use of R-package (Reviewer 3).*

We appreciate the editors’ and the reviewers’ positive and constructive comments. As shown below in this letter and in the revised manuscript, we have we have substantially revised it to address all the points raised by the reviewers.

**0)** We would like to address a common question raised by the reviewers: *the absence of any code complexity related measures in the paper*.

We believe this question was raised in response to us not being entirely clear about the research goals that are addressed in the article. We have added the research questions (**S-1**) we are addressing in this article and listed the motivations (**S-2**) for undertaking this problem. We believe these should be able to clarify why having code complexity related measures is not essential for the purpose of this work. To reiterate, the overarching goal of this paper is not to produce the best model of predicting or explaining exceptions nor to design the “best possible” (but impractical) quality metric. Our main objective here is to illustrate the interdependence of usage and observed number of crashes, and advocate the necessity for normalizing the observed number of software faults (like crashes) by the extent of usage to meaningfully compare quality of different software releases. Furthermore, the objective of the proposed measure is to highlight quality after adjusting for the effects of usage on exceptions. In other words, as intended, the proposed measure should reflect the impact of code, process, and coordination complexity on failures (see, e.g, [Cataldo et al, 2009](https://www.researchgate.net/profile/Jeffrey_Roberts2/publication/220070820_Software_Dependencies_Work_Dependencies_and_Their_Impact_on_Failures/links/0046352651c19d5aef000000/Software-Dependencies-Work-Dependencies-and-Their-Impact-on-Failures.pdf)[7]). These aspects ensure that a development team can utilize the quality measure to take appropriate action for quality improvement instead of being mislead by the effects of usage. As our findings show, the usage explains a large amount of variation in the number of observed exceptions which, conceptually, is easy to understand as the obvious mechanisms of how usage may translate into exceptions exist. While, similarly, it is clear why more complex code may contain defects, the relationship between latent defects and failures is much more complex, since not all defects may result in failures.

The software under consideration being a closed-source commercial software, it was not easy to obtain information on the source code and even more difficult to associate what exact code was used in each release of the software. However, we were able to obtain a significant portion of the commit log and used that information to investigate if the adding of the number of lines of code as a measure for code complexity affects the number of exceptions. The result of the analysis is presented in **Appendix I.**

The concern about possible confounding by the code complexity variables are addressed in **S-2**.

**Reviewer 1 Comments**

1. *Moreover, when studying NPM packages, the authors find that the number of issues per download increases over time.*   
     
   The exact statement in the **abstract and S-5.3.2, (pg 1, 28)**, “the value of the quality variable (number of issues per download) increases with time for almost half of the packages (2030 out of 4430 packages, 45.8%)” In other words, for less than half of the packages *the number of issues per download increases over time.*
2. *In the first place, a very few details on how the model was built are reported: what type of logistic regression model is used? Were the assumptions made on the underlying data by the model verified? What is the selected family type (e.g., "binomial" or "quasi-binomial")? Why? How was the model implemented? All these questions would deserve an answer to enable/ease replicability.*  
     
   We used Ordinary Least Squares (OLS) regression method for the linear regression, the information has been added to **S-4, S-5 (pg 13, 17)**. The model was implemented using the “lm” function in R. Furthermore, we have checked the validity of the underlying assumptions for using an OLS estimator and performed usual diagnostic checks of the fitted model, which is discussed in **S-5.1.1 (pg 18)**.
3. *The second major problem with the statistical modeling is the lack of control variables. Besides the independent variables gathered from Section 4, there might be additional factors explaining the number of exceptions: as an example, the complexity of source code may play a role. Unfortunately, the authors do not take into account those control variables: this produces a serious threat to conclusion validity. Thus, I would strongly recommend to (i) reason on the possible factors impacting the number of exceptions and (ii) control for them in the statistical model.*  
     
   We tried to addressed this with our answer to **point 0)**. As noted in the response to 0), our goal is to produce a measure that is directly affected by code and process complexity, so that the development team can take appropriate remediation action, instead of being mislead by the effects of usage on the number of exceptions. We do provide the analysis including code complexity covariates presented in the Appendix. The key question for which a software project needs to know an answer is whether or not a specific release has better quality than the prior release. We believe that we demonstrate that it is important to adjust exceptions for usage in order to answer that question. The specific action (e.g., refactoring complex code, doing better code inspections, improving testing) pending on the circumstances can then be taken by the development team.
4. *When employing Random Forest, the authors run a 10-fold cross validation. By definition, this strategy randomly partitions the set of data to create training and test sets: such a randomness may bias the results, as it can create specific combinations of training/test sets leading to under- or over-estimate the importance of the exploited variables. To account for this aspect, a robustness analysis should be performed: the authors can run the validation multiple times (e.g., through a 10 times 10-fold cross validation) to assess how stable the results are over different runs. This would reinforce the validity of the conclusions given.*   
     
   We thank the reviewer for the suggestion. Keeping this the suggestions by other reviewers in mind, we decided to employ a 10 times 2-fold cross validation because the sample of releases was quite small (11 in one case) and doing 10-fold cross-validation would, in such a case, predict a single observation. **S-4, S-5** updated.
5. *The structure of the paper should be substantially improved. In the first place, the paper contains several different analyses: a methodological overview that summarizes them would be highly beneficial for the reader. In the second place, the paper does not contain any research question, and this makes the reading experience really hard. Please, explicitly state what are the goals of the paper and the specific research questions that the paper wants to address. Third, the use of sections looks strange to me: in particular, I would recommend a more cohesive way of grouping related argumentations. For instance, Sections 2, 3, and 4 are all related to the dataset and would be better to have them in one section. Similarly, in the other sections (e.g., Sections 5 and 6), it is unclear what is part of the methodology and what is part of the results. I would strongly suggest to have a section for each research question with two subsections for methodology and results.*We thank the reviewer for the suggestion. The paper has been reorganized accordingly as described in the beginning of the response.
6. *In Section 9 the authors examine the results achieved on the NPM packages. While they may be interesting, I had hard times in understanding the real link between this study and the previous one. I think the paper should be more explicit in making clear what are the relations among the different sections of the paper and on the overall goal of the study. I believe that the paper can be strongly improved in this sense. Perhaps, the methodological overview and the addition of the research questions could help in reaching this point.*  
     
   We have added a research question addressing this point, (**S-1, pg 4**) and added further clarifications (**S-5.3**) addressing this point.
7. *The implications of the study are unclear. Section 10 is called "Implication of our Findings" but, unfortunately, no implications are discussed. This section indeed further discusses the results achieved, but does not provide any insight on how to make them actionable and what are the next steps to do by the research community. Thus, I believe that the section would require major changes to make it meaningful.*  
     
   The title of the section was corrected. We have also added the two major implications at the end of that section (**S-6, pg 31**), and further discussed the point in the Conclusion (**S-9, pg 35,36**).

**Reviewer 2 Comments**

1. *The study was only on 2 mobile apps (which actually had crash reports - the NPM dataset relied on issue reports). The number of releases were also very small. I'd suggest the authors to extend their dataset with more apps.*  
     
   Obtaining a crash report dataset with actual number of users, like the one we have proved extremely difficult as we investigated numerous options. A normal crash report dataset, like the one for Ubuntu linux distribution, does not contain any information on the actual number of users, making a study like this impossible. As we mentioned in limitations (**S-8, pg 34**), we had no control over the release cycle, and we were unable to get any more data for these two mobile applications as well. As a result, we were unable to implement this suggestion.
2. *Also, it would be worth exploring if issue reports contain crash reports.*  
     
   This is a good suggestion, and we’d like to investigate it further in later studies. However, the goal for including the study on NPM was to examine if our idea is extensible to other scenarios besides crash reports. Keeping that goal in mind, we chose to use the number of issues for NPM (since we are not aware of a crash report database for NPM). Our results indicate that even a measure that is not strictly a measure of software failures like crashes, is dependent on a usage parameter like the number of downloads for 99.2% of the packages examined, demonstrating the some extent of the generalizability of our idea of the need to adjust numbers of crashes by the number of users.
3. *The study also looked at only a few variables. How about internal variables (e.g. code complexity, lines of code, etc.)? These were known to correlate with software quality too. If they were included, would they have an impact on the correlation between the number of users and the quality outcome? Those internal variables are quite standard in the literature so I think such a study would be much needed.   
     
   The prediction models appear to have a poor predictive performance (as mentioned in the paper, but the specific results in terms of precision, recall and F-measure). This may be a major concern, suggesting that the number of users may not be a good predictor.*   
     
   We tried to address these points in our answer to point 0) and further in Section 2.
4. *Minor comments:  
   Fig 3: what does p=0 mean?  
   Page 18 - the paragraph at the end of section 6.2: what is the conclusion here?  
   Page 20 - "To obtain empirical ...": this sentence is quite confusing  
   Page 30 - subsection ???*  
     
   All of these points have been addressed in the paper. (**pg 18, 21, 22, 34**)

**Reviewer 3 Comments**

1. *The derived measure "Quality" is a function of two existing measures. It is the number of exceptions standardized by the number of new users. For this measure, the authors did not collect the data in a different manner, nor did they use a formula that semantically reflects different perspectives (e.g. process or product side)*Response  
   As stated in the paper, our objective was to obtain an actionable and easy-to-use measure that can more accurately compare quality of the release than simply the count of failures. We are not aware of how to measure software quality more directly. For example, issue reports are also influenced by usage as we show for the NPM packages. Some of the previous approaches attempted to model defects through code complexity, process complexity, and social complexity e.g, [7], so these and similar methods could be employed by the project to set up more specific action plans. Our point is that the more involved models for the most part do not adjust for the extent of usage with rare exceptions (e.g., observations by Caper Jones [26] and the work on Interval Quality [43], and Customer Quality [20]).
2. *The differences between the behavior of the new derived measure and other measures have not been well-investigated. The authors ignored potential product measures (e.g. Measures of code complexity) and process measures (e.g. mean time between failures).*As noted in the response to point 0), the goals of this study is not to investigate what product or process measures affect exceptions (as the topic has vast literature at least in regards to defects), but to correct the exception counts for the extent of usage. As stated in the paper, our objective was to obtain an actionable and easy-to-use measure that can more accurately compare quality of the releases.   
   We hope that the added research questions and Motivation Section clarify that it would not be necessary for us to achieve our goals.
3. *It is difficult to know whether this derived measure is better than the other measures in literature, as there is no sign for a baseline to compare with in this work.*We have added the timeline plots comparing the trends (last point in the update summary) that addresses the concerns about how this measure is different. We also added an explanation of the other similar measures used in the literature.
4. *Random forest is a wrapper-based FS technique, and is not the best ranker. The authors might consider using a hybrid-based ranker instead (e.g. AutoSpearman)*  
     
   Thank you for the suggestion. The AutoSpearman paper mentions that it handles the consistency issue in ranking, and indeed there is some variation in the relative ranking of the variables for different iterations of Random Forest, but the number of new users has consistently been ranked as the most important variable in different iterations of the Random Forest runs, and the usage intensity and frequency has consistently been ranked low. To establish robustness we are using the result of different models to reach our conclusion, and similar results were obtained by different models, so we do not feel it is essential to use the best available ranker for this task.
5. *The authors heavily relied on a minimal set of measures without explaining eg the reason behind including these measures and excluding others.*  
   We have added an explanation for that in **S-2**, The reason for dropping some of the observed measures in discussed in detail in **S-3.1.3.**
6. *Section 7 "The comparison with published results" is inadequate. The authors i) ignored several aspects in the comparative studies and ii) did not reproduce or replicate experiments to highlight the differences and similarities.*   
     
   We have added an explanation about the reason for including that section (**S-7.1**). The reason for highlighting the results of these studies it to compare the results of those studies and our study (please also see the issue 3 above), which we found was showing similar trends, for the purpose of complementing the lack of an extensive dataset in our study. There are important differences, however. First, other studies have not used data for mobile applications, hence we needed to replicate results using different measures (i.e., measures available from Google Analytics and similar data collection suites). Second, while defect data is widely available for open source projects, failure and, especially, usage data are not absent. Any type of data for commercial projects is confidential and can not be used for replication.
7. *The authors used a linear regression model. However, tests for normality (e.g. the Kolmogorov-Smirnov (K-S) or ShapiroñWilk test) have not been considered.*  
     
   We have added explanation about validating the assumptions for using OLS estimators (**S-5.1.1, pg 18**).
8. *The validity of the regression analysis is questionable as the number of subjects per variable is small. For iOS dataset, the authors used a sample size of only 11 observations for 6 then 4 variables.*  
     
   We agree with the comment. That is the reason why we use other models to validate our findings. Since a similar result was reported by all the three different approaches we took, that increases the confidence in our result.
9. *The construction process of Bayesian networks was a tool-driven experience. The authors applied and experimented most, if not all, of the algorithms included in the applied R-package, without a proper explanation on why they used those algorithms. The authors applied a continuous BN structure search and at the same time they applied a discrete version)*  
     
   We have added the reason for considering those methods in **S-4.1, pg 15**. However, given that we performed an extensive simulation study to choose the best performing method and used it, we have no reason to believe a tool driven method would, in any way, be less accurate than a model constructed by hand. In fact, by using this approach, we ensured an BN model construction process free from personal biases of the researchers.
10. *In the simulation, the authors created 1000 different datasets out of a small amount of observations (e.g. 173 and 11.)*  
      
    A simulated dataset is different from a bootstrapped dataset, and we used a custom BN model for creating the simulated dataset. The conditional parameters were calculated using our original data, but other than that, our data had no other role in the creation of the simulated dataset. We used the “rbn” function from the “bnlearn” package for creating the dataset. We invite the reviewer to check the manual page for that function: <http://www.bnlearn.com/documentation/man/rbn.html>
11. *The authors did not investigate or consider the false positives that may occur by constructing a belief network.*   
      
    The belief network was created by averaging bootstrap results with a custom threshold. However, we performed no additional checks. This point has been added to the “Limitations” section.
12. *The 10-fold cross validation has been applied to a dataset that consists of only 4 variables and 11 observations. Also, I could not know whether the authors split the datasets before or after applying this technique.*  
      
    Keeping the suggestions from other reviewers in mind, we decided to perform a 10 times 2-fold cross validation instead (See answer to R1: Q4).   
    For the second point, we have given the link to the public GitHub repository that includes the code we used for the study.
13. *The selection criteria behind Choosing LR, BN and RF*   
      
    Explanation added in **S4, pg 13.**
    1. We started by using the OLS estimator since it is one of the simplest modeling methods that gives good models in a lot of situations and the result is easy to interpret.
    2. However, we were unsure about the accuracy of the result due to the presence to moderate to high correlation between some of the predictor variables (e.g. the Release Date and Release Duration variables had a correlation of -0.88 for the iOS application data). Therefore, we decided to use Bayesian Network (BN) for modeling the interrelationship among these variables, since the accuracy of this model is unaffected by the presence of high correlation among the predictors. Variables with high correlation simply appear as connected nodes in the final model. Since the use of Bayesian Network models is not very common in this context and it is one of the main contributions of this paper, we discuss BN models in greater detail later in this section.
    3. The third and last modeling approach we used is Random Forest regression method. Random Forest is one of the best off-the-shelf models that work well with almost all types of data and it is easy to get the relative importance of the predictor variables from a fitted model. These two factors led us to use Random Forest regression as a third modeling technique to identify the most impactful predictors explaining the number of exceptions. Another reason for using the Random Forest model is that it is a predictive model, while the other two are explanatory models and they serve different purposes. We wanted to examine the effect of post-deployment measures for the purpose of prediction and observing which variables play an important role.

1. *Choosing these specific 6 measures.*  
   Explanation added in **S-3 (pg 6), S-3.1.3**.
2. *Choosing the datasets.*  
   Explanation added in **S-1, S-3, S-3.2**. Given the lack of publicly available datasets that has both the post-release software failure (like defects or crashes) numbers and software usage measures, having access to this dataset gave us an opportunity to perform a study like this and investigate the interrelationship between the number of software failures and software usage.
3. *Inadequate justification on the aggregation process used to combine/sum the observations collected from the commercial datasets.*  
   Explanation added in **S-3.1.3 (pg 8).**We had two main reasons for aggregating the data:
   1. The goals of our study are concerned with identifying the relationship between exceptions and other post-deployment variables for different releases, and defining a quality measure to compare the qualities of different releases. Therefore, having the measures aggregated at per-release granularity is essential.
   2. We would have been able to take a time-series based approach and still work out our goals if the releases were cleanly separated in time, i.e. if there were no overlap between releases. Unfortunately, we observed from the data that users continue to use one release long after the subsequent releases are available, and there is no clear pattern about how long a release is used. Therefore, we had to aggregate the data to a per-release level to be able to achieve the goals of this study.
4. *Why do we need to build a predictor (i.e. RF)?*  
   Explanation added in **S-4 (pg 13), S-9 (pg 36).** Also, see answer to Q-13.
5. *The authors used 4430 popular NPM packages. How did they measure popularity?*  
   By the number of downloads, it has been mentioned in the paper. (**S 3.2, pg 11**)
6. *The methods that were employed in constructing the belief networks are broad.*  
   That is by design. We wanted to perform the simulation study on a broad range to techniques to identify the best performing heuristic method in this context (i.e., for the software failure data).