The influence of bug subscribers on the development of Android

Antonio Arias Losada
Master on Free Software 2012, Vigo Edition
Pontevedra, Spain
antonio.arias@gmail.com

Francisco Alberto Rocha Rivera
Master on Free Software 2012, Vigo Edition
Vigo, Spain
rocha.francisco.a@gmail.com

***Abstract*—The Android bug tracking system has a subscribing mechanism whose purpose is to let the developers learn about the importance of a particular bug for the users of the operating system. This mechanism scores a star to a bug each time a user subscribes to that particular bug. In this MSR challenge report we try to find how that mechanism influences the way the developers prioritize the resolution of a bug, either by giving the bug a higher priority or by resolving the bug faster. We study the time to close bugs reported in the public Android BTS, as well as the priority of the bugs, trying to relate each of those variables with the number of subscribers of the bugs. The results show that most of the bugs have the same priority, and that the resolution time is not influenced by the number of stars. We analyze why this is happening and conclude that there is no relation neither between the number of stars of a bug and the time it requires to be solved, nor between the number of stars of a bug and the priority the developers assign to the bug.**

***Keywords*-Android, MSR challenge, bug tracking system, BTS, subscribers, stars, priority, bug resolution time**

I. Introduction

Bug reporting is an essential component of the development model of an open source project. Testing and validation of the project is mainly performed by users, who can contribute back to the project by reporting bugs. The more users inspecting and testing the code of a project, the quickest any bug will be located and the more probable is that someone finds a fix for the bug (Linus' Law [10]). Therefore, bug tracking systems (BTS) are essential for the management and growth of open source projects, allowing direct and organized communication between users and developers of the project.

The bug tracker plays a crucial role on the development of the project as it centralizes development requests. For project planning, the tracker presents a big challenge because of the large amount of information gathered. Prioritizing all that information is fundamental for the definition of the roadmap of the project. The developers need to classify the bugs according to some criteria to generate a prioritized list of pending tasks, given that it is impossible to solve all the bugs as they are being reported.

To do that classification, the number of users interested in a particular bug should be a good measure of the relevance of the bug and, by extension, a measure of the importance of that bug for the developers of the project. To help cope with this issue, some bug tracking systems incorporate a mechanism which lets users subscribe to bugs, and thus enable a mechanism for prioritizing them. This is the case of Android. Like most open source projects do, the Android project maintains a public bug tracker where users can report bugs and request features for the Android software stack, as well as subscribe to bugs. As stated by the Android bug tracker [8], the number of users subscribed to a bug influences the way developers prioritize the bug. We would like to know if the mechanism is effective and working as expected.

From the moment a bug is reported and until it is closed, the bug may be in different states of progress. The current state is tracked through a status assigned to the bug. Bug reporters and members of the community are allowed to subscribe to any particular bug to be notified by mail each time the status of the bug is changed. For each bug, the tracker keeps a metric named stars. A star is scored to a bug each time a user subscribes to the state of the bug. Thus the number of stars of a bug is equivalent to the number of members of the community that have explicitly shown interest on the state of that particular bug. When reporting a bug or while browsing the reported issues, the users click on a star-shaped check box to subscribe to a bug. Hence the name of the metric. This information is reported in the tracker documentation to be considered by the developers to prioritize the resolution of the bugs, so the developers are expected to attend first those bugs with more subscribers, that is, the ones which have more relevance for the community.

We believe that if developers take into account the number of subscribers of a bug for the development of the project, then we should observe some relation between the number of stars of a bug and the time that bug takes to be solved. We should also expect to see a relation between the priority of a bug and the number of stars of that particular bug. The objective of the present paper is to evaluate these two statements using the bug reports provided in XML format by the MSR 2012 challenge [1].

The rest of the present paper is organized as follows: the next section describes work related to this project. Section III provides both the data source and the statistical analysis used

in this study, while Section IV presents our results. Section V presents a discussion about conditions that constrain our results. The last section includes conclusions and some considerations for further work.

II. Related Research

The influence of the subscribers of bugs on Open Source projects has not been studied thoroughly. However, the resolution time of bugs has been related with other parameters in some works. Herraiz et al [6] studied the mean time to close bugs reported in Eclipse, and related it to how the severity assigned by users affects this time. In particular, the study classifies bugs according to the time it takes to solve them, grouping the results in severity categories. Crowston et al [4] have also studied bug fixing time as a measure of FLOSS projects success, entering bug priority as a factor of the study. The paper presents a statistical analysis of the time to close bugs modeled as censored data.

III. Methodology

A. Data gathering

We have made the current study from the data provided for the Mining Challenge of the 9th International Working Conference on Mining Software Repositories (MSR) [1]. Specifically, we have extracted the needed information from the bugs reported on the tracker of the project, provided in XML format.

We developed a Java application to parse the XML document with the StAX API (Streaming API for XML) [3]. Traditionally XML APIs are either DOM based, which support random access but are very resource consuming, or event based, which are very efficient but not very user-friendly. The StAX API is designed to close the gap between those two approaches. This API allowed us to cope with the big XML document provided for the challenge (2GB) efficiently and effectively.

We only included closed bugs. For each bug we evaluated the required time to be solved, computing it from the bug opening and closing timestamps provided in the bug document. We converted their timestamps into UTC seconds. Subtracting the values corresponding to the closing and opening timestamps we obtained the number of seconds invested to solve the bug. The resolution is the minimum we considered appropriate, since the bug that required less time took 17 seconds to be solved. The resulting application parses the bugs and exports the number of stars of that bug, its priority and the number of seconds that the bug required to be fixed to a text file. This file allowed us to import the data into a GNU R [9] environment to process it. GNU R is a free software environment for statistical computing and graphics. It provides a wide variety of statistical and graphical techniques, so it fulfilled our statistical and graphic needs for the present study [2]. Once we had the data we needed imported into GNU R, we applied the methodology presented on the next section.

B. Statistical Analysis

We are trying to validate two hypotheses, (1) the time required to solve a particular bug has some statistical dependence with the number of subscribers of the bug, and (2) the priority assigned by the developers to a particular bug has some statistical dependence with the number of subscribers of the bug.

To validate the first of our hypothesis, first we analyzed the boxplot of the times to close bugs and the number of subscribers, to have a quick evaluation of the ranges of the variables. Then we grouped the bugs by number of subscribers to study the relation between number of subscribers and resolution time. We discarded the possibility of evaluating the mean length of time required to close bugs with a particular number of subscribers, because not all the values for every number of subscribers have the same number of bugs, so the average values are skewed. Therefore we studied the resolution times for all the bugs instead of studying only the mean values for the bugs that share the same number of subscribers. In any case, we also computed the medians for the resolution times, which gave us a more robust view of the resolution times. Our study is based on the iteration over possible statistical relations to find the influence of the number of subscribers on the resolution time of the bugs.

We evaluate the density function of the logarithm of the number of subscribers to deduce a possible statistical distribution.

Then we analyzed the cumulative distribution function of the resolution time. In particular, we use the inverse of the cumulative distribution function, because cognitive psychology tells that it is easier to discern variations on descent curves [5].

We also tried to approximate the distribution of the samples with a Poisson distribution and repeated the process with an exponential distribution.

Finally we computed the value of the correlation between the resolution time and the number of subscribers of the bugs. We computed the Pearson coefficient, which measures linear correlation, as well as both Kendall and Spearman coefficients, which measure if there is any kind of strong relation between both variables, even if it is not linear.

To validate our second hypotheses, we have also tried to relate bug priority with number of subscribers in an analogous way as we did with the bug resolution time.

IV. Results

According to the Android BTS [8], the number of subscribers of a bug helps developers know which bugs are more relevant for the community. Figure 1 represents the time to solve bugs. The horizontal axis represents the number of

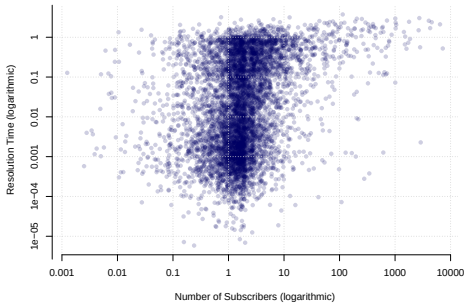

Figure 1. Bug resolution time

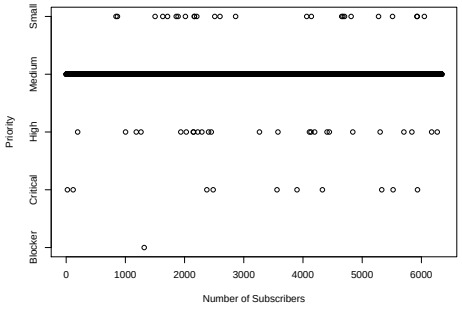

Figure 2. Bug priority

subscribers of a bug, while the values on the vertical axis represent the time to close bugs. To increase resolution, both axes present logarithmic scale. Given the variability on the vertical axis, we deduce that the time to close a bug is not influenced by the number of subscribers of the bug.

The existence of many bugs with a single star seems to reflect the fact that the community is not involved in the bug subscription mechanism. Moreover, there is a large amount of bugs with just a single subscriber. When a bug is reported, by default the reporter of the bug is subscribed to it (logically it is assumed the interest of the reporters on the bugs they report).

We have found no statistical relation between the number of subscribers and the resolution time of the bugs by evaluating density functions and cumulative distributions. Correlation coefficients also show no statistical dependency between both variables.

We suppose that the number of subscribers of a bug is somehow used by the developers to assign the priority value of each bug. We have represented the priorities against the number of subscribers. Figure 2 shows the results. The horizontal axis represents the number of subscribers of a bug, while the values on the vertical axis represent the priority of the bugs. The figure shows that almost all bugs have "Medium" priority, which seems to portray the fact that developers do not pay special attention to this property.

Our results seem the true reflection of the closed contribution model of the Android development approach and the fact that Google takes all the decisions regarding the operating system unilaterally. The release cycle makes the source code public only after releasing binaries.

V. Threats to Validity

Our model is based on features that can easily be gathered from bug report submissions. Construct validity requires that we correctly identify resolution time, number of subscribers and priority for each bug of the project. These are straightforward values directly obtained by simple arithmetic calculations from the information provided by the tracker.

The main threat to the external validity of our study is the actual veracity of the values provided by the tracker. There are several systems keeping track of the bugs of the operating system, among which some are kept private by Google [7]. This means that the information of the public tracker is partial and incomplete, as reflected by the fact that most of the bugs present the same priority. That compromises the validity of our study, but in no case prevents our method to be applied to any other set of data.

We have not classified a sample from the data, but taken all the bugs stored in the database to make the analysis. This approach may suppose some threats to the validity of the analysis, because not all the components of the operating system have the same interest for the community. Furthermore, not all the components have the same number of users capable of detecting their bugs. For example, it is not as usual to spot a bug on the Dalvik virtual machine than it is to do it on the Gmail application. Moreover, in a mainstream project like Android it must be taken into account that not all the users of the operating system are reflected in the bug tracker, as most of them are not even aware of its existence and purpose. Our study may be repeated filtering the different bugs by component (Dalvik, Tools, Applications, etc.) thus selecting the size and properties of the sample to be more precise.

It is not possible to link the information present on the bug tracker with the information on the code repositories, so we could not apply any metric regarding the code (lines, comments, committers) to improve our study.

VI. Conclusions and Further Work

Almost all bugs registered in the public Android tracker have the same priority and there is no relation between the resolution time and the number of subscribers of the bugs.

As stated by the Android Open Source Project, there are several trackers for the bugs of the operating system, among which only the studied tracker is official and public. This is a major determinant for the results of our study. The information of the public tracker is distorted because

the actual development is driven by the private tracker, as reflected by the fact that almost all bugs share the same priority. We may deduce that the priorities of the bugs are tracked by the private tracker and never reflected back to the public tracker. We also believe that the number of subscribers of a bug is not a major factor in the prioritization made by the developers.

Regarding the relation between the resolution time and the number of subscribers of a bug, many other factors come into play. For example, a bug that might have more relevance for the community can be solved faster because it is not very complex or has high private priority, thus not spending enough time open to build up many stars. In contrast, a bug that is less relevant for the community may stay open more time, either because it is very complex or has low private priority, making it easier for the bug to accumulate more stars during its life.

We would like to have access to the private bug tracker databases of the project to evaluate their information, apply the presented statistical analysis and compare the results with the ones we have found here. However, bug tracking systems databases not always are available to researches and third parties, especially those related with OEMs. This issue is even more evident in the case of Google, which not only keeps a private bug tracking system, but also keeps the development private, making the source code available to the community only after releasing binaries. Making the information of the development available would help better understand the dynamics of the project.

REFERENCES

[1] 9th Working Conference on Mining Software Repositories. Mining Challenge. http://2012.msrconf.org/challenge.php.

[2] J. Adler. *R in a Nutshell*. O'Reilly, 2009.

[3] Codehaus. The Streaming API for XML (StAX). http://stax.codehaus.org/.

[4] Kevin Crowston, Hala Annabi, James Howison, and Chengetai Masango. Towards a portfolio of FLOSS project success measures. 2004.

[5] S. Few. *Now You See It: Simple Visualization Techniques for Quantitative Analysis*. Analytics Press, 2009.

[6] I. Herraiz, D. German, J. Gonzalez-Barahona, and G. Robles. Towards a Simplification of the Bug Report form in Eclipse. 1998.

[7] Android Open Source Project. Life of a Bug. http://source.android.com/source/life-of-a-bug.html.

[8] Android Open Source Project. Report Bugs. http://source.android.com/source/report-bugs.html.

[9] The R Project. What is R? http://www.r-project.org/.

[10] E. Raymond. *The Cathedral and the Bazaar*. O'Reilly, 1999.