Improving App Quality Despite Flawed Mobile Analytics

Conference or Workshop Item

How to cite:

© 2020 The Authors

https://creativecommons.org/licenses/by-nc-nd/4.0/

Version: Version of Record

Link(s) to article on publisher's website:
http://dx.doi.org/doi:10.1145/3387905.3388603
ABSTRACT
Analytics can help improve the quality of software; the improvements are affected by the fidelity of the analytics. The impact of poor fidelity may vary depending on the type of data being collected, for example, for crashes low fidelity may be sufficient.

The mobile ecosystem includes a platform where apps run and an app store that intermediates between developers and users. Google’s Android ecosystem provides all the developers with analytics about various qualities of their app through a service called Android Vitals that automatically collects data on how their app is performing.

My research found ways to improve app quality through using mobile analytics, including Android Vitals. It also found fidelity flaws in several analytics tools provided by Google. They confirmed and validated some flaws and chose not to discuss others.

KEYWORDS
Android-vitals, Crashlytics, Firebase, Mobile-analytics

1 INTRODUCTION
Development teams want users to use their apps in order to achieve various goals such as revenue growth, popularity, etc. They would like their apps to be of high quality both for their own satisfaction and to encourage users to use their apps more.

Inherently developers construct models of their software in order to design, implement and test their apps. However, no model survives unscathed on contact with reality (paraphrasing "no plan survives contact with the enemy" [14]).

As an example, the Catdroid project includes over 1,500 hand-crafted automated tests intended to test almost every eventuality [13] and is developed using clean code and a myriad of additional approaches intended to deliver excellent code [7, 16]. Yet, in-use, the crash rate averaged 3.91%, nearly four times the bad behaviour threshold of 1.09% Google considers excessive [3]; and the app was in the bottom 7% of education apps according to Android Vitals [2]. Reality trumped the model.

Software Analytics has been established for at least a decade, for instance through the work of Buse and Zimmermann [5, 6]. The value of applying analytics to improve software’s quality and fitness-for-purpose has also been established, e.g. by Microsoft where usage analytics were collected for 48,000 internal users of the Lync desktop application [17].

Mobile app ecosystems materially affect software development practices [1] by acting as an intermediary and a conduit between developers and end users. The relevance and importance of mining feedback for mobile apps is well established, e.g. understanding why users provide 1 and 2 star ratings [8]. Development teams can use this feedback to improve the fidelity of their models of their software. Among various feedback channels, user feedback, i.e. ratings and reviews, seems to be well studied [8, 18, 19].

Figure 1: Improvements in the crash rate for the Pocket Code Android app

Figure 2: Data Sources and Views

My research focuses on automated feedback gathered using analytics by the platform, using Google’s Android Vitals analytics service; and within an app using analytics libraries, e.g. Crashlytics and Firebase, that developers optionally include in their mobile app [4]. Figure 2 provides a model of these two forms of feedback:
essentially digital data is collected either by an app or by the platform, forwarded to servers for analysis, and then various analytics reports are made available to authorised team members.

2 APPROACH

We used mixed methods during our research aiming to complement hands-on work with interviews with developers of business critical apps [10]. As the analytics is restricted to authorised users and as it is only generated for actively used applications we collaborated with various development teams who provided access to their app’s analytics data. The research involved working with several mature open-source projects with a combined user-base of 400,000. In each case one Android app was selected with the highest crash rate as measured by Android Vitals. The reports and data from Android Vitals were used on an ongoing basis to identify crashes and other stability issues. We raised bugs in the respective bug tracking systems for the most common issues. These were triaged by the development teams who fixed issues and released newer versions of both the selected app and their other apps.

To record and make the reports and data available we developed and open-sourced Vitals Scraper and to correct some flaws in the reporting we also created Android Stability Analysis [4].

3 RESULTS

For the Kiwix application, the crash rate decreased from 4.05% to 0.39% while the sibling apps continued to have a similar crash rate for a 4 month period. The sibling apps were then updated based on the improved code base and the reported crash rates have reduced by at least 50% within a month. For Pocket Code, the crash rate reduced from 3.91% to 1.07% Note: the crash rate of the sibling app Pocket Paint also decreased during the experiment, from 1.66% to 0.82%, as the development team chose to fix a crash that adversely affected many users. [3] [12].

Aspects of the research were published at MobileSoft 2019 [9], WAMA 2019 [12], and in a jointly authored book in 2015/16 [11]. Extracts of Android Vitals data and reports have been made available to the research community. Various flaws were found in Android Vitals and Fabric Crashlytics, two of Google’s key analytics tools. The flaws include differences in the outputs in of over 10:1 in the crash rates they calculate. The flaws were reported to the relevant Google development team in 2019-2020 who accepted the bug reports and requested a comprehensive report so they could address the relevant issues. While the Google engineering team acknowledged the differences, they are defensive about the reasons why. What if they have similar flaws in their payment calculations in the WAMA 2019 [12], and in a jointly authored book in 2015/16 [11].

Independent development teams of a variety of Android apps (including Moonpig[12], LocalHalo, and Moodspace [12]) confirm they actively use analytics to monitor the quality of their apps and fix crashes reported in these mobile analytics tools. Trello’s Android team stress the importance of applying a similar approach [15].

4 CONCLUSION AND FUTURE WORK

Our research indicates developers can use mobile analytics to improve the reliability and performance of their mobile apps. The default Android Vitals analytics service provides adequate sources of information on crashes and ANRs and confirms the effects of fixes in newer releases of the respective apps. In-app analytics libraries provide finer-grained lower-latency feedback and developers tend to prefer these data sources to using Android Vitals.

Each of the tools we evaluated has fidelity flaws, the effect varies depending on the category of the data. Nonetheless, the developers improved the crash rate of their apps, by between $\frac{1}{2}$ and $\frac{1}{3}$, they also independently continue to apply the techniques I proposed. The engineering team at Google for Android Vitals follows our work. We will evaluate whether adding in-app analytics events, e.g. using Firebase, to apps improves their development and testing.

REFERENCES


[9] Julian Harty. 2019. Google Play Console: Insightful Development using Android Vitals. In MobileSoft 2019 [9], WAMA 2019 [12], and in a jointly authored book in 2015/16 [11]. Ex- tracts of Android Vitals data and reports have been made available to the research community. Various flaws were found in Android Vitals and Fabric Crashlytics, two of Google’s key analytics tools. The flaws include differences in the outputs in of over 10:1 in the crash rates they calculate. The flaws were reported to the relevant Google development team in 2019-2020 who accepted the bug reports and requested a comprehensive report so they could address the relevant issues. While the Google engineering team acknowledge the differences, they are defensive about the reasons why. What if they have similar flaws in their payment calculations in the same app store, which handled an estimated gross revenue of 29.3 Billion USD in 2019 [9].? Similar developers work on both.


