Google Play Console: Insightful Development using Android Vitals and Pre-Launch Reports

Google Play Console: Insightful Development using Android Vitals and Pre-Launch Reports

Julian Harty
Commercetest Limited
julianharty@gmail.com

***Abstract*—Through a case study, I present several related software tools provided by the Google Play Console to Android developers. To help developers monitor and diagnose how their apps are performing in the real-world, Google Play Console includes *Android Vitals* to track technical performance, *Release Management* in particular automated pre-launch testing and analysis, and *User Feedback* for ratings and reviews. They enable developers to identify and address issues that affect active users of their Android apps. The (in-)stability reports include pertinent attributes such as the device model and Android version which can help correlate and triangulate when issues adversely affect customers. App developers can also use the findings to discover flaws in their development and testing practices. Ratings and reviews are already popular topics for researchers, perhaps *Release Management* and *Android Vitals* deserve equivalent interest and may offer several new rewarding research areas.**

***Index Terms*—Android Vitals, Google Dev Console, Kiwix, Metrics, Mobile Applications, Pre-Launch Report, Software Quality.**

I. Introduction

Android app developers predominantly use Google Play to publish their apps to users and over 1,000,000 developers currently use the Play Store in 2018 [1]. In recent years Google has launched several, practical, free tools to help these developers improve their Android apps.

For example, developers of the Busuu language app[1] reported how they used several of the tools and reports to find and address performance problems. By doing so they also improved their app rating from 4.1 to 4.5 stars [2].

A. Android Vitals

In July 2017, Google announced Android Vitals - "Android Vitals is designed to help you understand and analyze bad app behaviors, so you can improve your app's performance and reap the benefits of better performance" [3]. Since then Android Vitals has evolved with changes and new features being released on an ongoing basis [1].

B. Release Management

The Release Management tool provides a dashboard that includes various metrics together with time series graphs that compare: installs and uninstalls, crashes and Application Not Responding (ANRs)[2], and reviews and ratings for the current production release(s) against previous releases. The dashboard includes links to more detailed reports for each topic. These

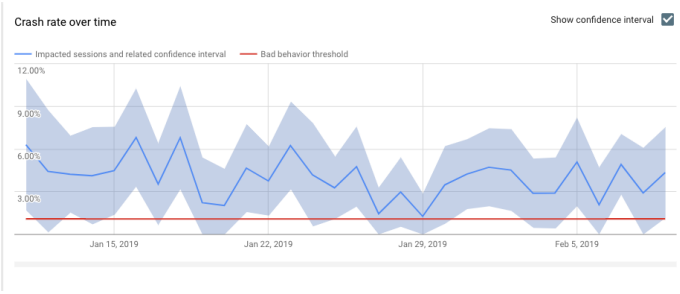

Fig. 1. 30-day Crash rate for Kiwix.

graphs and reports provide daily updates on how the various releases are performing and approximately how many devices each release is installed on. A final tool bundled with Release Management is the pre-launch report.

C. Pre-launch Reports

Google Play automatically generates pre-launch reports shortly after the binary file (known as an 'APK') for an Android app is uploaded. They are run for every non-production app *i.e.* for *internal test*, *closed* (alpha), and *open* (beta) tracks [4]. These reports were announced at Google IO in 2016 [5] and continue to be revised and enhanced [6].

The test automation software is called `Robo`. Robo starts and then interacts with the app; it records a video, screenshots as well as details of crashes and other potential flaws. By default, Robo explores and interacts with the app and seems to behave similarly to `Android Monkey`, one of the stalwarts of many research papers on test automation for Android . Robo also offers developers additional facilities, including `deep links`, `Robo script`, and `demo loops`[3]. Fig. 2 illustrates the test results for various releases of an app; in this example there were issues detected for the version code '182520'. The issue includes a crash and 17 issues related to accessibility.

The tests run on a range of devices with a variety of Android releases and locales – for some developers this may be the first time their apps are tested in a language other than English.

Up to five languages can be specified for testing.

[1]https://play.google.com/store/apps/details?id=com.busuu.android.enc
[2]https://developer.android.com/topic/performance/vitals/anr
[3]Details available at https://developer.android.com/distribute/best-practices/launch/pre-launch-crash-reports

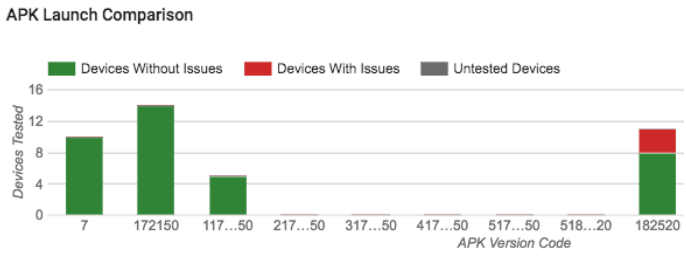

Fig. 2. Pre-launch report showing failures.

D. No app is an island

Some of the key strengths of these tools include comparisons, trends, and benchmarks that enable developers to learn how their app is performing in the field. The reports often include comparisons between the current and most recent previous period (often 30 days) and trends can be calculated, particularly when the timescales are set for longer periods. Apps are automatically compared against both top rated apps and apps in the same category to provide benchmarks; these include crashes (as illustrated in "Fig. 3"), ANRs, and ratings. "Fig. 1" shows Google provides a bad behavior threshold to help developers know what adequate looks like.

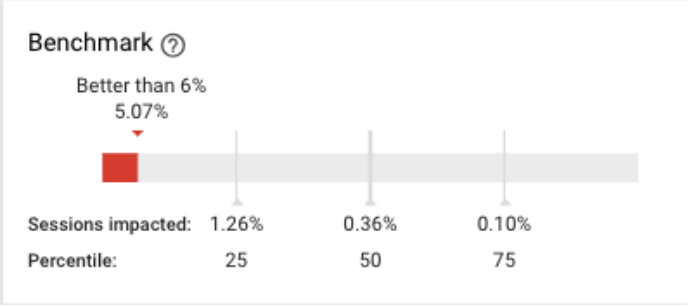

Fig. 3. Benchmark: crash rate for Kiwix compared to the Books and Reference Category.

E. Maturity and adoption

These tools are relatively new and, as yet, are used by a minority of the developers. This seems to be the first paper published on these tools.

Furthermore, they are of limited value for low-volume apps as several reports are only available once there are data volumes to enable the data to be anonymized sufficiently to protect users' privacy.

F. Data is available for download

Monthly reports are available to download[4] either interactively or using a command line tool `gsutil`[5]. The reports include crashes and ANRs, ratings, reviews, installations, and financial information.

[4]https://support.google.com/googleplay/android-developer/answer/6135870
[5]https://support.google.com/googleplay/android-developer/?p=stats_export

II. Exemplary Project

The author was provided access to the Google Play Console for a family of open-source[6] projects called Kiwix. Kiwix provides 18 active Android applications[7] that enable various Wikipedia and related contents to be used offline. There is a general purpose app which includes a download manager, the rest are packaged with specific content such as articles related to Medicine, STEM, and Travel.

A. How were the various apps performing?

Android Vitals considers 'Stability' in two aspects: crashes detected by Android *i.e.* not handled by the app, and ANRs. It provides 'Bad Behavior Threshold's for both, these are 1.09% for crashes and 0.47% for the ANR rate.

1) Bug Clusters: The Download Manager is unique to the main Kiwix app; and over 47% of the crashes are related to this feature. The next most common crashes are related to: incorrect Android Lifecycle management $\geq$ 14%, and processing contents of Wikimedia and other materials $\geq$ 8%.

As "Table. I" shows, the apps fail more often on newer releases of Android. Inexorably user populations will gradually migrate to newer versions of Android (at the time of writing Android 8.0 and 8.1 are the most popular) so these figures are particularly concerning and help developers to focus their efforts on addressing the causes of the increased crash rates on the newer releases of Android. Back in 2013 changes to the Android APIs (intrinsic to Google releasing new versions of Android) were identified as a key threat to the success of Android apps [7].

TABLE I
Crashes of Kiwix by Android Version

Version	*Impacted sessions*	*Crash-free sessions*	*#Sessions*	*Bottom quartile*
9	8.28%	91.72%	3k	1.70%
8.1.0	7.39%	92.61%	6k	1.29%
8.0.0	4.89%	95.11%	13k	1.19%
7.0	2.08%	97.92%	4k	0.75%
6.0.1	1.40%	98.60%	3k	0.75%

B. Surprising discoveries

The crash rates vary significantly for seemingly similar apps. 'Chemistry & Physics simulations' has the highest crash percentage at 4.08%, more than twice that of any of the other seemingly similar packaged apps. The simulations use lots of JavaScript to provide rich and highly-interactive web content[8] and the higher crash rate may well be related to flaws in the code that reads in and interprets this much richer content than the relatively static material from Wikimedia sites, *etc*.

[6]https://github.com/kiwix/kiwix-android/
[7]https://play.google.com/store/apps/developer?id=Kiwix+Team
[8]Sourced from https://phet.colorado.edu/en/simulations/category/html

C. Know your apps

Android Vitals rates Kiwix as being in the bottom 6% of similar apps for 'excessive network usage'; while none of the other apps have *any* network usage reported. Why does the Kiwix app differ? It is designed and intended to enable users to download various content from Wikimedia and other sources, including TED Talks. All these are downloaded over the network connection, as is the latest catalog of the content that is also hosted online. Therefore it is unsurprising that it is an extensive user of the network compared to other apps in the 'Books and Reference' category.

III. Advancing the state of practice

Crash reporting libraries have been available for years, as have mobile analytics. Automated monkey testing has been provided as a core test automation tool since the early days of Android; and there are numerous other automated app crawler tools available currently. Nonetheless, Google Play Console offer the opportunity for development teams to receive relevant feedback without having to explicitly use special libraries or testing tools. Google also reduces 'friction' as the data is gathered from many millions of users who have opted-in to providing crash reports, *etc*. automatically[9]. A key factor is that Google Play has incredible breadth as it collects data for the millions of apps in the Play Store and from all the Android devices where users have opted-in to providing usage and diagnostics information. Nothing else has the combination of depth and breadth in the world of software app stores.

IV. From analysis to action

Although Google have made great strides in reducing the friction for developers by collecting the data and making it freely available to the app developers, the developers still need to firstly notice and - as necessary - take corrective action in order to actually improve the qualities of their apps and increase the users' perceptions, ratings and reviews.

For Android Vitals, here is a suggested checklist:

- Identify and group common clusters; deal with any obfuscation by uploading mapping files[10];
- Interpret the stacktrace to find where and when the exception was raised;
- Match the crash to the version(s) of the relevant app: this is key when an app has multiple releases active in production;
- Review the source code to establish code paths that could lead to the crash and experiment by testing the app on one or more devices.

Developers can then decide what action to take both immediately and longer-term. Their choices include:

- Exclude the flawed code: change the code paths so it does not reach the vulnerable code;
- Hide it: catch the exception and hide the crash;
- Deal with the symptoms: especially if the cause is unknown or outside the developer's control *e.g.* it may be related to a flaw on particular devices or responses from a third-party API;
- Report the problem to the user: An error well-controlled where the user is informed may improve the trust users place in the app and thereby increase their satisfaction with the app;
- Identify and address causes of the problem(s): sometimes the most challenging to achieve immediately is to work out the causes of the problem and write code that deals with these causes appropriately and correctly.

Generally developers will need to create, upload and release newer versions of their app in Google Play where they can then monitor the effects using the tools and reports mentioned here.

V. Related Work

As mentioned earlier, elements of the tools are already available from various sources. Furthermore researchers have published in various related areas. Gómez and coauthors - perhaps coincidentally - envisaged an App Store 2.0 [8] that offers similar capabilities to those Google were releasing around the same period (2017).

Autonomous Automated Testing Tools: Android Monkey has an established pedigree where it finds crashes in many Android apps even though it applies simple concepts and has few "smarts". It has become a benchmark for many researchers who use it as a comparison with whatever automated testing they are assessing. In recent years many commercial entities have launched autonomous automated testing tools, including: Google Firebase Test Lab, Amazon's AWS Device Farm, Appachhi, Bitbar, Monkop, and `test.ai`.

Distributed testing of mobile apps: Vikomir and co-authors evaluated the effectiveness of testing on sets of Android devices and claimed an effectiveness of 90% with five devices, where the most successful approach used different types (versions) of Android operating systems [9].

Analysis of Reviews: There are well-established streams of research that mine ratings and reviews from mobile app stores, for instance [10], [11].

Analysis of Crashes: Various researchers have investigated ways to automatically reproduce crashes. These include the work of Gómez and coauthors [12] where a client library is installed and activated explicitly when users consent; it identifies patterns of crashes from crash logs gathered from the field. Their approach is unlikely to be practical for general Android users who are not likely to accept and enable third-party software to monitor their use and also run tests remotely on their devices.

Release Readiness: Nayebi, *et al* [13] introduce the concept of marketability established using various release criteria, including quality and feedback from users.

VI. Limitations, bugs and flaws in the tools

"With Great Power Comes Great Responsibility." These tools are intended to help us improve our apps, therefore they

[9]https://support.google.com/googleplay/android-developer/answer/6083203?hl=en-GB

[10]https://support.google.com/googleplay/android-developer/answer/6295281

are also subject to scrutiny. I found various flaws in the tools; these have been reported directly to Google, here is a summary of most of them:

Counts: The Dashboard shows `11.99K` Crashes & ANRs, yet the detailed reports only had 11653. The dashboard includes an extra day than the detailed report.

Crash rates: the value is pre-filtered to the "Production" release with the highest crash rate (5.89%) on the `AppHealthOverviewPlace` page, yet clicking-through to see the details shows an overall crash rate of 4.48% (across all releases) for the main Kiwix app. The 5.89% rate is, however, for the *second-most* popular app version (`182160`) with `5K` sessions, and ignores the most popular app version (`1182160`) with `23K` sessions with a 4.58% crash rate.

More *crash clusters* are reported than really exist, for example Android Vitals shows 12 clusters for `(LibraryFragment.java:156)` rather than one.

Session counts differ depending on the grouping used, *e.g.* `28K` sessions by app version and `27K` sessions by Android version.

Figure 4 shows the heading for Top countries is consistently for the *second* ranked country not the first. Only the first 10 apps are listed in various dropdown menus, the others need to be searched for explicitly e.g. by Java package name.

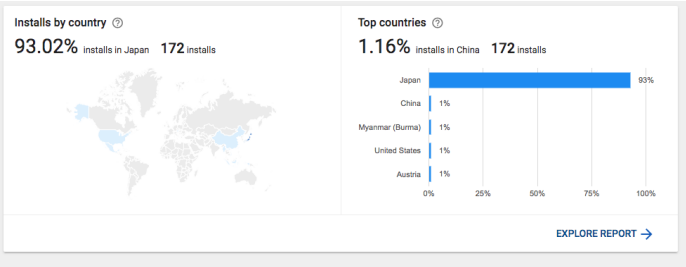

Fig. 4. Bug discovered in the Acquisition Report of Google Play.

VII. Conclusion and Future Work

The tools I have presented enable developers to monitor key aspects of how their Android apps are performing on end-user devices. They complement and integrate with the User Feedback tools also provided to the developers.

Developers do not need to change their applications to benefit from these tools (although they are encouraged by Google to add crash reporting and mobile analytics). The data and reports enable them to identify actual and potential flaws quickly so they can mitigate and address them. The integrated pre-launch testing and analysis catches some problems before the app is released; while the stability and performance reports provide de-facto feedback on their work including design, development and testing of various releases of their apps. Because of the volume thresholds, Google imposes the reports are currently more relevant to apps with 10,000+ active users.

Researchers are encouraged to use, experiment and evaluate the tools provided by Google Play Console. For example: on the efficacy of the pre-launch testing and reports; to compare and contrast the efficacy with other techniques such as textual analysis of reviews; to determine the sweetspot for actionable data while protecting privacy of users; *i.e.* the conundrum of how much data to collect and provide in the reports; and on ways to reproduce crashes on-demand for easier fault diagnosis using the crash clusters, based on [14] for instance.

A. Acknowledgements

Thank you Arosha Bandara, Yijun Yu (Open University), and Fergus Hurley (Product Manager, Google), for their help.

References

[1] P. Correa, Android Developers Blog: Wrapping up for 2018 with Google Play and Android, 18-Dec-2018. [Online]. Available: https://android-developers.googleblog.com/2018/12/wrapping-up-for-2018-with-google-play.html. [Accessed: 12-Feb-2019].

[2] Android Developers, Android Developer Story: Busuus performance improvements yield jump in user rating. 2017 https://www.youtube.com/watch?v=KS3EdZ6TETY.

[3] F. Hurley, Android Developers Blog: Android Vitals: Increase engagement and installs through improved app performance, 10-Jul-2017. [Online]. Available: https://android-developers.googleblog.com/2017/07/android-vitals-increase-engagement-and.html. [Accessed: 12-Feb-2019].

[4] Android Developers, Ensure You're Launching a High-Quality App or Game with the Pre-Launch Report. 2019 [Online]. https://www.youtube.com/watch?v=jSR_1sPvckU [Accessed 11 Mar. 2019].

[5] P. Kochikar, Android Developers Blog: What's new in Google Play at I/O 2016: better betas, the pre-launch report, benchmarks, a new Play Console app, and more, 18-May-2016. [Online]. Available: https://android-developers.googleblog.com/2016/05/whats-new-in-google-play-at-io-2016.html. [Accessed: 12-Feb-2019].

[6] Android Developers, Use pre-launch and crash reports to improve your app. 2019 [Online]. https://developer.android.com/distribute/best-practices/launch/pre-launch-crash-reports [Accessed 11 Mar. 2019].

[7] M. Linares-Vsquez, G. Bavota, C. Bernal-Crdenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk, API change and fault proneness: a threat to the success of Android apps, presented at the Proceedings of the 2013 9th joint meeting on foundations of software engineering, 2013, pp. 477487.

[8] M. Gomez, B. Adams, W. Maalej, M. Monperrus, and R. Rouvoy, App Store 2.0: From Crowdsourced Information to Actionable Feedback in Mobile Ecosystems, IEEE Software, vol. 34, no. 2, pp. 8189, Mar. 2017.

[9] S. Vilkomir, K. Marszalkowski, C. Perry, and S. Mahendrakar, Effectiveness of Multi-device Testing Mobile Applications, in 2015 2nd ACM International Conference on Mobile Software Engineering and Systems, Florence, Italy, 2015, pp. 4447.

[10] A. Al-Subaihin et al., App store mining and analysis, in Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile - DeMobile 2015, Bergamo, Italy, 2015, pp. 12.

[11] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, Why people hate your app: making sense of user feedback in a mobile app store, in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 13, Chicago, Illinois, USA, 2013, p. 1276.

[12] M. Gómez, R. Rouvoy, B. Adams, and L. Seinturier, Reproducing context-sensitive crashes of mobile apps using crowdsourced monitoring, in Proceedings of the International Workshop on Mobile Software Engineering and Systems - MOBILESoft 16, Austin, Texas, 2016, pp. 8899.

[13] M. Nayebi, H. Farahi, and G. Ruhe, Which Version Should Be Released to App Store?, in 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Toronto, ON, 2017, pp. 324333.

[14] White, M., Linares-Vsquez, M., Johnson, P., Bernal-Crdenas, C. and Poshyvanyk, D., 2015, May. Generating reproducible and replayable bug reports from android application crashes. In Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension (pp. 48-59). IEEE Press.