Taming App Reliability: Mobile Analytics 'in the wild'

Julian Harty
Commercetest Limited
High Wycombe, Buckinghamshire, UK
julianharty@gmail.com

Arosha K. Bandara
The Open University
Milton Keynes, UK
arosha.bandara@open.ac.uk

ABSTRACT

The importance of mobile apps in people's lives places a responsibility on app developers to ensure that their software is reliable. Despite the widespread use of mobile analytics tools to support this task, there is limited understanding of their use in industrial app development contexts. This paper reports on industry experiences of using mobile analytics tools through a case study based approach. The key findings highlight four main factors that affect developers' practice and identify ways of improving the effectiveness of mobile analytics to help developers make their apps more reliable.

CCS CONCEPTS

• **Software and its engineering → Software design tradeoffs**; **Software reliability**; **Extra-functional properties**.

KEYWORDS

App development, Mobile Analytics, SDK development, Software Tools

ACM Reference Format:
Julian Harty and Arosha K. Bandara. 2024. Taming App Reliability: Mobile Analytics 'in the wild'. In *28th International Conference on Evaluation and Assessment in Software Engineering (EASE 2024), June 18–21, 2024, Salerno, Italy*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3661167.3661169

1 INTRODUCTION

Billions of people use mobile apps on two main mobile platforms, Google Android and Apple iOS, and their related operating systems. These users rely on these apps for many aspects of their lives, including managing their finances, communicating, tracking their health and well-being, and education. The importance of mobile apps in the lives of so many people means app developers should ensure their software is of high quality, particularly in terms of reliability. This requires that they pay particular attention to problems that can cause their apps to crash or become nonresponsive [1] and users stop using apps that crash [11, 17].

Failures in production software are often transient [10] and logging failures in production can help devs address them [5]. While some quality issues can be identified through pre-release testing, many factors limit developers' ability to assure the reliability of their app when deployed on users' devices; *e.g.*, one limitation is the paucity of input and usage conditions that developers can establish, create, or recreate (even if they wanted to do so).

EASE 2024, 18–21 June, 2024, Salerno, Italy

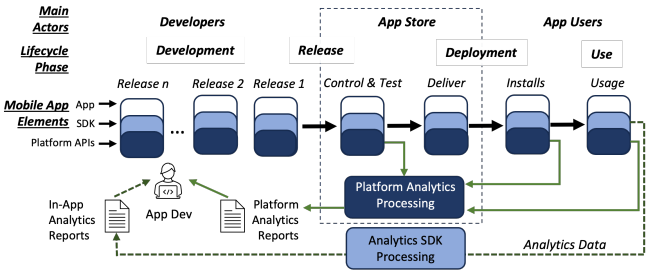

Figure 1: Mobile Analytics in Context

Therefore, measurements must be sourced from the field and *mobile analytics* has become an established and endemic approach to do this. App developers choose to incorporate mobile analytics into their apps, and most apps on Google Play include at least one in-app mobile analytics SDK; Firebase is the most popular and installed in over 70% of apps [2]. And yet, the accuracy and effectiveness of these analytics SDKs and tools for mobile app development teams is not well understood. Even recent research, *e.g.* when Zhu *et al.* studied app stores from a software engineering perspective, left software analytics as an area for future work [6].

A typical mobile app comprises the platform APIs, SDKs, and the app's source code, with each of these having a primary owner who is responsible for maintaining them. As shown in Figure 1, when an app moves from development to deployment and use, the main actors involved change from developers to owners of the app store and finally to users of the application. App developers often update the app and make new releases, here numbered 1, 2, ..., n, which percolate via the app store's approval and rollout processes and are eventually deployed on some of the userbase's devices [4].

When developers release apps to the app store platform analytics records activity generated by pre-launch reports; and when apps are installed on user's devices it tracks the installation, usage, and eventual uninstallation. In-app analytics can track usage of the app; and often does so in greater detail than the platform-level analytics.

This study discusses the use of mobile analytics tools in app development projects based on the experience of the lead author and doctoral research [14]. Through observations from various projects, it offers case studies that illustrate industrial practices, highlighting mobile analytics' strengths, weaknesses, and challenges, along with suggestions for enhancement.

2 METHODOLOGY

The study focused on mobile analytics tools within industry. It adopted a 'knowledge-seeking' approach to understand their use and a 'solution-seeking' perspective to explore enhancements in

their functionality and in the ways in which these tools are used by app developers in real-world contexts [20, p. 11-4].

This research is intended to augment and complement more granular research such as investigations of various characteristics of mobile SDKs in terms of auditing and privacy [3] and security [7], as these studies did not investigate the experience of developers using analytics tools *in practice*.

Case studies were the primary mechanism for this research and were chosen because they have proven to be an effective method to understand software qualities and are recommended where the research includes multiple data sources. Also, case studies are well suited for exploratory research, for which they can offer insights not available with other approaches, especially when multiple data sources are involved and the investigator has limited control over the research context [19, pp. 132-133].

Case studies were selected using 'purposive sampling' as described by Flick [8, pp. 180-182]. Of necessity, the sample used in this research was opportunistic, which Flick described as "convenience" sampling. As Flick wryly noted: *"the problem of access may be one of the crucial barriers"* [p. 182], which applies particularly when seeking access to sensitive data and information on software failures for commercial mobile apps. Organisations and app development teams need to be willing to have the guts of their development practices investigated and published.

Data used to develop the case studies included development artefacts (*e.g.* source code, output of software tools, bugs, and issues) from the different mobile app project teams, pre-study interviews with key members of each team, direct communication with developers, and analytics tools together with associated analysis artefacts. When gathering these data the lead author had different roles in each app development team, namely as: a consultant and/or an embedded developer *(an active participant integrated into the project team)*, a coach *(to the app developers)*, an interviewer *(of development teams to learn about their practices and results)* and an analyst / observer *(performing static analysis of code repositories)*.

The analysis of the data to develop the case studies and draw out key findings used a combination of techniques which can be broadly categorised into sense-making, sense-building, feedback mechanisms, and action research, as summarised below:

- 'Sense-making' methods focused on understanding current practice. It includes inductive analysis of different forms of data to identify areas of interest - *beacon finding*. These findings were augmented by *drilling down* - further data collection & analysis to understand areas of interest in context.
- 'Sense-building' methods build on – and test – insights found through sense-making. The research methods included micro-experiments performed on local apps, FOSS analytics experiments, & across-case comparisons.
- 'Feedback mechanisms' were used to support and verify the other analyses, comparing the observations with other evidence, and / or asking developers for clarifications or reflections.
- 'Action research' methods were primarily concerned with understanding the use of analytics in context and evaluating the effect of improvements in the use of analytics, in terms of adoption into their use & into performance of the apps.

Table 1 summarises the mobile analytics tool case studies and the associated research methods. The findings are illustrated using four representative app case studies: Catrobat: a popular visual programming development environment, Kiwix: providing Wikipedia and other offline content to 10 million people globally, Moonpig a commercial e-commerce app [16], and finally, a video conferencing app with a userbase of millions, identified as C1 here.

3 FINDINGS

Limitations inherent in mobile analytics tools: The data comes from other people's apps, devices, and usage, and the data varies accordingly. Variety brings complexity and outliers. To evaluate the tools, the data must be representative of production; this is hard to establish *a priori* so they are challenging to evaluate outside production.

Four key themes: emerged that affected how developers use mobile analytics tools in practice. They are:

Design: encompassing SDK design and developer experience. Some SDKs also collect meta-data that helps developers find and fix the cause of failures - mobile analytics SDKs collect such data by design. There are numerous engineering challenges for developers of mobile analytics SDKs. *E.g.* focus on developer experience is vital for providers of mobile analytics. Iteratively [1] innovated by providing tools aimed at improving the coherence and consistency of mobile analytics. The choice of programming framework can have a major effect on the collection and reporting of failures. For example, the React Native UI framework encapsulates failures to the extent that the mobile platform (Android) did not record them.

Fitness-for-purpose: mobile analytics products need to provide a good product fit for app developers, including providing actionable reports and integration into developer workflows. Programmatic access to analytics outputs is particularly important for large organisations and development teams. Android Vitals, while ubiquitous, only helps developers once their apps generate sufficient data to pass unpublished thresholds within the product.

Utility: includes the efficacy of the mobile analytics tool(s), the benefits of combining tools, and bug-localisation. Mobile analytics needs to provide value to developers in terms of finding and addressing reliability issues.

Dependability: considers the extent to which developers can rely on a mobile-analytics service and/or the underlying tool; and even a tool with few flaws cannot compensate for a heavily flawed service. There are numerous flaws in the mobile analytics tools and services; 17 were identified in Google Play Console with Android Vitals [14, p.207], the most studied of the offerings. Link rot and testability also adversely affect the dependability of the offerings.

We also identified cross-cutting issues, including fidelity and ethical considerations associated with the use of mobile analytics.

Flaws, quirks, and limitations were found in the mobile analytics tools used by the various apps. For example, reports from Microsoft's App Center, used by C1's app, often showed dates up to several days in the future [2], which made reported issues difficult to pinpoint.

[1] Since acquired by Amplitude

[2] Microsoft App Center accepts logs from 25 days ago to 3 days future, see learn.microsoft.com/en-us/appcenter/diagnostics/troubleshooting.

Case Study	Role of Researcher	Main Research Method	Research Opportunities	Research Purpose
Firebase Analytics	Interviewer	Ask the app devs	Insights into maintaining reliable apps and SDKs	Obtain expert users' views of the most popular mobile analytics tool
Fabric Crashlytics & Firebase Crashlytics	Analyst/Observer	Sense-making	Compare several Google Analytics tools	Triangulation
Microsoft App Center	Analyst/Observer	Sense-making	Crash & Error analytics	Blue-chip alternative
Sentry	Analyst/Observer	Sense-making	Mobile analytics for React Mobile cross-platform apps	Increase variety and coverage of tools
Sentry	Embedded developer	Hybrid/Mixed	Mobile Analytics for iOS Machine Learning SDK	Additional mobile platform
PostHog	Embedded developer	Hybrid/Mixed	Mobile Analytics for iOS Machine Learning SDK	Additional mobile platform; comparison with Sentry
Google Play Console with Android Vitals	Analyst/Observer	Ask the tool devs	Mutual symbiotic cross-pollination	Learn about the providers' perspectives
Iteratively	Consultant	Ask the tool devs	'Behind the curtain'	Discover state of the art approach to improving the rigour of mobile analytics
Iteratively->Amplitude	Interviewer & Informal Early Experience Program	Ask the tool devs	Explore state of the art novel tool	Insights into improving the tools

Table 1: Tool-centric cases: the research perspective

A more complete example are the 17 flaws found in Android Vitals and Google Play Console via the 4 illustrative apps. 14 are detailed in [12, 13, 16]. Table 2 lists all 17 for quick reference.

Despite flaws, all app developers accepted mobile analytics as useful and necessary *even if some ignored them for the most part*. Perhaps this is unsurprising given many people may consider good health to be useful, and yet ignore the state of their health?

One of the app case studies, Moonpig, exemplified the benefits of active engagement where they were able to consistently achieve and maintain high reliability of their app. Even when reliability decreased, the development team managed improvements and timed their releases to balance the overall end-user experience for their overall userbase [16, pp. 29–30]. Google Play does not allow developers to filter who receives newer releases, so the devs held off from releasing a new release to the most affected subset of users (a minority who were on newer releases of Android at the time) rather than disrupt the overall Android userbase.

Commercial teams generally incorporate at least one in-app mobile analytics SDK. Opensource teams tend not to; Catrobat used to before abandoning Crashlytics to protect the privacy of end users. Privacy may be a 'hidden' consideration in the choice of in-app mobile analytics and well worth further study.

Charges are a more visible consideration, as illustrated in a recent project in 2023 where default instrumentation of lifecycle events led to a monthly charge of between £2,000 and £3,000 by the commercial service. This charge occurred even though the Machine Learning SDK that incorporated this mobile analytics service was only being used by a few 10's of users out of an active user population of 10,000's [15]. The intended behaviour was to only record events when the Machine Learning SDK was in-use; however, the mobile analytics SDK's default setting meant it also recorded lifecycle events for the entire active user population [3].

[3]Thankfully, the service credited the first month's charge and helped the devs disable the recording of lifecycle events.

4 DISCUSSION

To date, four complementary directions of improvement have been identified for mobile analytics tools.

(1) Fixing flaws in the tools, including those listed in Table 2 for Google's Android Vitals, would help to increase the dependability and encourage evaluation of the various mobile analytics services. An appetite to fix, test, and monitor such flaws is key to addressing them.
(2) Improving integration into other tools and into developer workflows becomes more important as the business grows. Integration also facilitates analysis and knowledge dissemination within a larger team, rather than requiring each developed to access the mobile analytics service. Microsoft's App Center's APIs demonstrate integration is possible [4].
(3) Improving auditability and verifiability: *e.g.* providing SDKs and analytics as opensource and open cross-tool benchmarks. Crashprobe [5] provided this service for several years, similar work should be viable to resurrect.
(4) Improving analytics capabilities: The ability to search and filter results varies significantly between tools. Recent work in 2023 with PostHog's HogQL helped identify and diagnose a production issue within hours and demonstrates such capabilities are feasible; sadly, other tools lack any such facility which limits analysis from a practical perspective.

Ownership of the collected data is a key concern, as is determining who has permission to use various aspects of the data.

4.1 Industrial implications of this research

For mobile analytic tool providers: Apply the four directions of improvements identified in the above discussion; and encourage public research and auditing of your service(s).

For app developers: One of the key insights from the work was the importance of development teams continuing to actively engage with mobile analytics on an ongoing basis. Moonpig's engineering team was able to achieve excellent results throughout

[4]openapi.appcenter.ms/
[5]crashprobe.com/ios

ID	Flaw	Consequences
01	Testing discouraged	Limits investigation into behaviours and their impacts.
02	Negative populations	Nonsensical and therefore untrustworthy statistics.
03	Repeated graphs	Poor UX, waste of space (waste of real estate).
04	Gaps in the data	'Flying blind', loss of confidence in the service.
05	Inconsistent data ranges for some reports	Poor UX, confusing, may lead to incorrect/flawed decisions.
06	Missing URL parameters	Results incorrectly filtered.
07	No updates for 10+ days	'Flying blind', loss of confidence in the service.
08	Incorrect ranges in reports	Off-by-one errors.
09	Unexplained negative headline rate	E.g. the combination of new users acquired and lost imply a subzero userbase.
10	Poor grouping of clusters	Inaccurate summaries, rank of failures skewed, suboptimal prioritisation.
11	No Service problem-reporting	Lack of transparency of historical service outages, *etc.*
12	Lack of reports (despite usage)	Unusable analytics for low to mid range usage by end users, 'Flying blind' after take-off.
13	Second country's data conflated with that of the first	Misleading report, poor UX.
14	10x mismatch with crashlytics	Lack of trust in at least one of the mobile analytics tools.
15	Incorrect date for last update	Misleading developer experience, loss of trust in the service.
16	Identical crash clusters in the paged list of ranked results	Adversely affects counts of matching crash clusters, confusing.
17	Stale data in some graphs	Data should be fresh and recent in each of the reports for the 'last' so many days *e.g.* 'Last 7 days'.

Table 2: Flaws discovered in Google Play Console with Android Vitals, based on [14, p.208, Table 8.1]

the case study. Hackathons boosted reliability for the Kiwix and Catrobat apps for as long as the development team paid attention to the problems reported by mobile analytics. However, *as interest waned poor reliability returned*, hence, to provide highly reliable apps developers need to actively use mobile analytics.

4.2 What challenges remain?

One of the hardest challenges in this area of research is obtaining access and then permission to publish results gleaned from commercial app development teams and their companies. This is particularly true, as studying the reliability and stability of mobile apps may lead to an intense focus on what goes wrong with those apps in the field for end users. Companies, particularly those with legal and/or compliance departments, sometimes prevent publication. One of our case studies terminated the case study shortly before listing for an Initial Public Offering (IPO). Similarly, Google's product and engineering teams for Android Vitals were prevented by their policies and practices from providing updates on the flaws that were shared with them. These restrictions make such research particularly challenging and may be a key reason why so little research has been done or published on the accuracy and effectiveness of mobile analytics for mobile app development teams. Of note: secrecy in industry can lead to longer-term problems (*e.g.* the British Post Office Horizon system cover-up) [18].

5 CONCLUSION

Mobile analytics tools are in widespread use by mobile app and SDK developers. Implicitly, they depend on the results being trustworthy. Although these tools are useful, many have flaws that may limit their utility and dependability. This research, grounded in industry, explores a variety of these tools with the aim of encouraging research and improvements. Investigating the effectiveness of these tools helps developers and, indirectly, their users. In turn, as more is known about the behaviours, limitations, and flaws of the tools, there are opportunities to improve these tools and how developers learn how to use them. Notably, Google now offers online training for Google Play Console, Android Vitals, and Prelaunch reports [9].

REFERENCES

[1] Android Project. 2024. *Monitor your app's technical quality with Android vitals.* https://support.google.com/googleplay/android-developer/answer/9844486

[2] AppBrain. 2024. Android analytics libraries. https://www.appbrain.com/stats/libraries/tag/analytics/android-analytics-libraries Retrieved 2024-Jan-17.

[3] Álvaro Feal et al. 2021. *Don't accept candy from strangers: An analysis of third-party mobile sdks.* Vol. 13. Bloomsbury Publishing, 1–27.

[4] Chuck Rossi et al. 2016. Continuous deployment of mobile software at facebook (showcase). In *24th ACM SIGSOFT Intl. Symposium on Foundations of Software Engineering*. 12–23. https://doi.org/10.1145/2950290.2994157

[5] Jeanderson Cândido et al. 2021. Log-based software monitoring: a systematic mapping study. *PeerJ Computer Science* 7 (5 2021), 38. https://doi.org/10.7717/peerj-cs.489

[6] Wenhan Zhu et al. 2024. What is an app store? The software engineering perspective. *ESE.* 29, 1 (2024), 35. https://doi.org/10.1007/s10664-023-10362-3

[7] Xian Zhan et al. 2022. Research on Third-Party Libraries in Android Apps: A Taxonomy and Systematic Literature Review. *IEEE Trans.on Software Engineering* 48, 10 (2022), 4181–4213. https://doi.org/10.1109/TSE.2021.3114381

[8] Uwe Flick. 2018. *An Introduction to Qualitative Research* (6 ed.). SAGE Publishing. 696 pages.

[9] Google Inc. 2024. Play Console Features: Google Play Academy. https://playacademy.exceedlms.com/student/path/265136-play-console-features

[10] Jim Gray. 1986. Why Do Computers Stop and What Can Be Done About It?. In *Fifth Symposium on Reliability in Distributed Software and Database Systems*. IEEE Computer Society, 3–12.

[11] Quinn Grundy. 2022. A Review of the Quality and Impact of Mobile Health Apps. *Annual Review of Public Health* 43, Volume 43, 2022 (2022), 117–134. https://doi.org/10.1146/annurev-publhealth-052020-103738

[12] Julian Harty. 2019. Google Play Console: Insightful Development Using Android Vitals and Pre-Launch Reports *(MOBILESoft '19).* IEEE Press, 62–65. https://doi.org/10.1109/MOBILESoft.2019.00019

[13] Julian Harty. 2020. Improving App Quality Despite Flawed Mobile Analytics *(MOBILESoft '20).* IEEE Press, 2 pages. https://doi.org/10.1145/3387905.3388603

[14] Julian Harty. 2023. *Improving Application Quality using Mobile Analytics.* Ph. D. Dissertation. The Open University. https://oro.open.ac.uk/90629/

[15] Julian Harty. 2024. Experiences Developing a Computer Vision SDK for Mobile Apps. In *IEEE/ACM 9th International Conference on Mobile Software Engineering and Systems (MOBILESoft '24).* 3. https://doi.org/10.1145/3647632.3651393

[16] Julian Harty and Matthias Müller. 2019. Better Android Apps Using Android Vitals. In *3rd ACM SIGSOFT Intl. Workshop on App Market Analytics (WAMA 2019).* ACM., 26–32. https://doi.org/10.1145/3340496.3342761

[17] Nishant Jha and Anas Mahmoud. 2019. Mining non-functional requirements from App store reviews. *ESE.* 24, 6 (01 Dec 2019), 3659–3695. https://doi.org/10.1007/s10664-019-09716-7

[18] Stephen Mason. 2021. The Post Office Horizon Scandal: a brief chronology. *Digital Evidence and Electronic Signature Law Review* (2021).

[19] Per Runeson and Martin Höst. 2008. Guidelines for conducting and reporting case study research in software engineering. *ESE.* 14, 2 (2008), 131. https://doi.org/10.1007/s10664-008-9102-8

[20] Klaas-Jan Stol and Brian Fitzgerald. 2018. The ABC of Software Engineering Research. *ACM Trans. Softw. Eng. Methodol.* 27, 3 (2018), 51 pages. https://doi.org/10.1145/3241743