Experiences with Google Approval Process: an automated approach to enhancing efficiency in Android releases

Ewerton Andrade
ewerton.andrade@sidia.com
Federal University of Rondonia,
SIDIA Inst. of Science and Technology
Porto Velho, RO, Brazil

Janislley Sousa
janislley.sousa@sidia.com
SIDIA Inst. of Science and Technology
Manaus, AM, Brazil

Matheus Lopes
matheus.lopes@sidia.com
SIDIA Inst. of Science and Technology
Porto Velho, RO, Brazil

Davi Barbosa
davi.barbosa@sidia.com
SIDIA Inst. of Science and Technology
Porto Velho, RO, Brazil

Weslen Lima
weslen.lima@sidia.com
SIDIA Inst. of Science and Technology
Porto Velho, RO, Brazil

Jimmy Lacerda
jimmy.lacerda@sidia.com
SIDIA Inst. of Science and Technology
Porto Velho, RO, Brazil

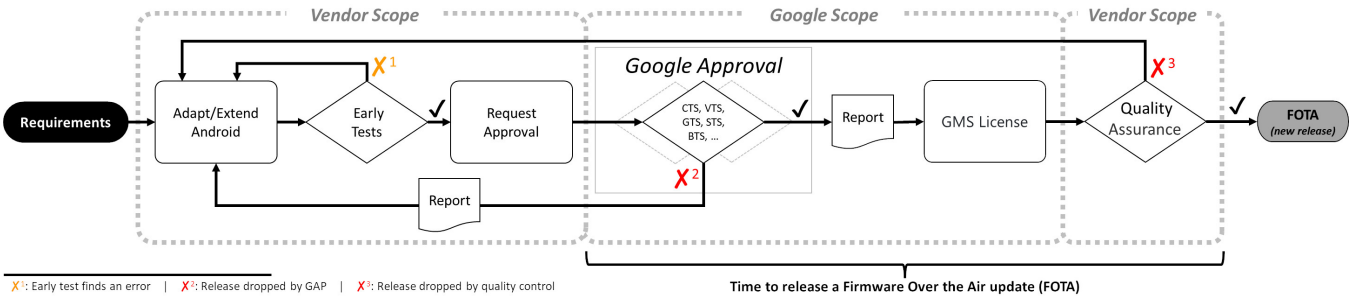

Figure 1: Flowchart for use Firmware-Over-the-Air (FOTA) updates to launch new Android releases approved by Google.

ABSTRACT

Google enforces a licensing requirement for Original Equipment Manufacturers (OEM) to market Android-powered mobile devices with Google Mobile Services (GMS) and to perform Firmware-Over-the-Air (FOTA) updates. This certification process is known as the Google Approval Process (GAP). Due to the potential for this process to be time-consuming and costly, it is common to implement strategies aimed at reducing failures and resource wastage. In this context, this paper presents the approach adopted by our institute to enhance its efficiency on GAP. Through the implementation of this approach, the following results were achieved: (I) a decrease of up to 92% in Android software release drops (an avg of ~55% compared to 2022); (II) a reduction of up to 86% in processing time (an avg of ~22% compared to 2022); and (III) an increase in fixed Common Vulnerabilities and Exposures (CVEs) over the years.

CCS CONCEPTS

• **Software and its engineering → Software verification**.

KEYWORDS

Android, GMS, FOTA, Google Approval, Software Release, CVE.

1 INTRODUCTION

Despite providing and maintaining Android under an open-source license, Google has several proprietary applications and services that run on devices powered by its operating system. This variety of Google applications and services is known as GMS [2], and encompasses Google Play Store and its essential apps. Most Android users expect these GMS apps to come pre-installed on their device and work flawlessly [4]. Another important user desire is simplicity, one-click updates on their devices, without the need for complex processes or physical connections. These simplified updates can be done by FOTA updates [6]. Consequently, these expectations must be pivotal in the software development for Android devices [4].

Therefore, Google requires Original Equipment Manufacturer – OEMs (as well-known vendors) to obtain a license to sell devices with GMS included and launch new releases to update its devices using FOTA. Thus, the responsibility of applying OS updates, system updates, and security patches with fixes for CVEs lies with the OEMs [12]. They are technically the ones who build the operating system by customizing the Android Open Source Project (AOSP) code maintained by Google.

During this development process to licensing the device with allowed FOTA updates and embeds GMS, Google carries out a series of tests and validations known as GAP [17]. These tests cover compatibility, drivers, hardware, security, and several other components [5, 13, 17]. Ensure that the software OEMs provide is reliable and compatible with the open-source version of Android, with new features and updates running smoothly without issues. A Android software release may be dropped by Google for not meeting its standards or at the end of the process by the OEM itself, if a not applicable requirement is identified that compromises its quality. Due its complexity and extensive nature, GAP can take weeks [10] and may cost from US$ 40,000.00 to US$ 75,000.00 [4].

Table 1: Data related to five smartphone models under our management.

	Model 1			Model 2			Model 3			Model 4			Model 5		
	2021	*2022*	*2023*	*2021*	*2022*	*2023*	*2021*	*2022*	*2023*	*2021*	*2022*	*2023*	*2021*	*2022*	*2023*
Dropped releases:	116	25	18	118	163	27	142	98	71	189	112	15	36	37	18
Releases with FOTA:	335	85	63	186	209	137	171	84	107	380	256	201	54	63	54
Sum of all releases:	451	110	81	304	372	164	313	182	178	569	368	216	90	100	72
Days to FOTA (*sum*):	10568	2037	1405	5768	5259	3432	6731	2667	3148	11030	7924	5935	1146	1569	983
Days to FOTA (*avg*):	31.55	23.96	22.30	31.01	25.16	25.05	39.36	31.75	29.42	29.03	30.95	29.53	21.22	24.90	18.20

To prevent wasting time and resources, Android's smartphone vendors must develop strategies to reduce the risk of failures and wasted time during the Google Approval Process to launch new Android releases in the market. Therefore, in this paper, we present our approach to carefully, efficiently manage and develop activities to: (I) decrease the number of releases dropped during GAP; (II) reduce the time spent during the release approval process; and (III) launch new releases covering more vulnerability fixes.

2 GOOGLE APPROVAL PROCESS

Figure 1 illustrates a generalized flow that encompasses specific details and processes from various vendors, where each step can be seen as follows:

Requirements: in this phase, all the stakeholders and their requirements are identified, analyzed, specified, validated, prioritized, and are compared with Google guidelines [1]. It may include new features, performance improvements, CVE fix, and OS upgrades.

Adapt/Extend Android: once requirements are established, developers start the modification of the core version of Android to incorporate the specified requirements. During this phase, applications that are not part of the open-source Android are integrated.

Early Tests: as previously discussed, the request for the GAP can result in significant implications. Hence, OEMs conduct early tests to anticipate and fix issues that may occur.

Request Approval: formal GAP request defined by Google.

GAP tests: mandatory tests that software releases undergo to ensure the security and quality standards (to details see [3, 10, 17]).

Report: Google generates a detailed report during the software release process, providing information on the software release. If errors occur, a report is sent to the OEM for rectification.

GMS License: its signature will be registered on the Google and can be accessed at a later time on the Play Store.

Quality Assurance: verify any new flaws or CVEs in GAP that have not yet been analyzed. Also, any changes in local telecommunications regulations or government requirements are monitored.

FOTA: a FOTA update is launched for the users.

3 OUR APPROACH

As discussed in Section 2, several steps are within the vendors scope and can be managed to mitigate the difficulties associated with GAP. We base our approach on four pillars: (P1) standardized and centralized tools for versioning and controlling software artifacts, (P2) extensive automated tests, (P3) continuous training, (P4) and guarantee the documentation of all software processes.

Firstly (P1), during the **Requirements** stage, we use a specialized system to handle both public and confidential requirements [11]. Once these requirements are identified, analyzed, specified, validated, and prioritized, they are registered in this specialized system to be implemented by a developer.

After these implementations are carried out, they will be recorded in another specialized system as Change Lists (CLs) and the **Adapt/Extend Android** begins. Hence, Software Project Leaders (Software PLs) need to apply and propagate these changes across each of the OEM's device models. This process must consider all the details of each model, such as varying hardware within the same commercial model and government restrictions.

After these changes are applied, project leaders can request or perform various **Early Tests** as needed, according to our second pillar (P2). Our organization conducts a wide range of tests to anticipate issues and avoid wasting time and money by requesting GAPs that would result in a dropped release. Therefore, we carry out Android tests provided by Google (e.g., CTS, VTS and STS), as well as tools developed within our institute, such as: BTS-Validator [16], BSA Tool [8], IMEI OFF Tool [7], PRIMA [5], and TSS Script [9]. Mainly, our approach uses a wide range of tests based on historical information provided by Google in GAPs previously carried out. These tests and the modification of Android occur iteratively. Only after receiving no fails will the project leader **Request the GAP**.

In parallel, following the third pillar (P3), updates, modifications, and lessons learned during this development and validation process are continuously carried out by stakeholders through training for improvements and reduce the time to release flow [14, 15]. Finally, following the fourth pillar (P4), a wiki is maintained to document all processes and improvements of this approach [14].

4 RESULTS

At our institute, we work on developing software implementation for Android devices (Smartphones and Tablets), upgrading operating systems, and maintaining releases for models produced by a global OEM. As previously mentioned, this paper aims to discuss the efficiency gains achieved through the approach described in Section 3. To present and discuss these results, Tables 1 and 2 provide data related to the five models (Smartphones) managed by Porto Velho office of our institute since the beginning of 2023. The following sections will analyze and discuss these results in detail.

4.1 Android Releases dropped

As shown in Table 1, Model 4 demonstrated the most favorable outcomes in terms of the reduction in releases dropped by GAP. With just 15 releases dropped in 2023, there was a decrease of approximately 92% compared to 2021, and about 86% in comparison to 2022. On the other hand, Model 3 yielded the least promising outcome among the five models, with 71 releases dropped in 2023 and the smallest reduction of ~27% when compared to 2022.

Despite these variations, the overall results achieved with our approach are promising. On average, there was a reduction of ~70% compared to 2021 and ~55% compared to 2022. These reductions translate into significant time and cost savings, as fewer dropped releases mean fewer rework cycles and faster time-to-market for new Android software releases. Additionally, it is noteworthy that a reduced number of maintenance releases for previously launched devices were required in 2023, indicating another improvement.

4.2 Time to approve a release on GAP

Model 1 demonstrated the most significant reduction when compared to 2021 (~86%), while Model 5 shows the most significant decrease compared to 2022 (~37%). Conversely, Model 3 once again presents the least favorable outcome among the models, with an increase of ~18% compared to 2022. However, when the total days are divided by the number of releases with FOTA, it is evident that there was an average reduction of ~7%, attributable to the increased number of releases launched for this model.

Overall, we observed an average reduction of approximately ~48% compared to 2021 and ~22% compared to 2022. These reductions not only enhance our operational efficiency but also reduce the resources and costs associated with prolonged approval processes.

4.3 Amount of fixed vulnerabilities

Complementing the data about the five models under consideration, Table 2 shows the consistent increase in the number of fixed CVEs over the years. This suggests that our approach not only enhances the efficiency of the process but also increases the security.

Table 2: Amount of fixed CVEs over the years.

	Critical	High	Moderate	*Total*
2021	42	346	62	450
2022	31	356	69	456
2023	45	496	5	546
Total	118	1198	136	1452

5 LESSONS LEARNED

Throughout the implementation of our approach, several critical lessons were learned that highlight the advantages and efficiencies achieved. Firstly, our approach resulted in a reduction in the number of dropped releases. We guarantee that each release was thoroughly tested before being submitted to GAP. This consistency in release quality minimized setbacks, leading to a more reliable software development cycle. Secondly, by using standardized tools for control software artifacts, extensive automated testing, continuous training, and updated documentation, we enhance team productivity and streamline the GAP. This efficiency translated into faster turnaround times and a streamlined development process.

Therefore, we observed a notable reduction in software development costs. For example, assuming an annual expenditure of US$1,000,000 on executing the GAP, maintaining the average reduction of ~55% would result in an expense reduction of US$550,000. By streamlining processes and more automated tests, we curtailed unnecessary expenditures and allocated resources effectively.

Lastly, we noted an enhancement in security. Our approach integrated rigorous testing and security protocols, significantly mitigating risks and vulnerabilities. Notably, each release launched under this approach included fixes for identified CVEs, ensuring the prompt resolution of known security issues.

6 CONCLUSION

In conclusion, the implementation of our approach demonstrated tangible improvements in cost reduction, release reliability, efficiency, and security. These results indicate that our focus on standardized tools, extensive automated testing, continuous training, and thorough documentation has positively influenced our software development and maintenance processes.

However, the use of standardized tools and automated testing frameworks may not fully capture the intricacies of specific scenarios and custom software environments. Moreover, unforeseen security vulnerabilities could emerge, requiring ongoing updates.

Future research would prioritize the development of flexible testing frameworks that can adapt to various software environments. It is also crucial to investigate advanced security measures to address potential vulnerabilities. These enhancements and research directions will strengthen the robustness and adaptability of our approach, ensuring its long-term effectiveness and relevance in the academic sphere.

ACKNOWLEDGMENTS

The authors are grateful for the support offered by SIDIA R&D Institute in Mobile Cybersecurity project. This work was partially supported by Samsung, using resources of Informatics Law for Western Amazon (Federal Law No. 8.387/1991), in accordance as foreseen in article No. 39 of number decree 10.521/2020.

REFERENCES

[1] Android. 2024. CDD. https://source.android.com/docs/compatibility/cdd.
[2] Android. 2024. Google Mobile Services. https://www.android.com/gms.
[3] Android. 2024. Platform testing. https://source.android.com/docs/core/tests.
[4] Charles Arthur and Samuel Gibbs. 2014. TheGuardian: The hidden costs of building an Android device. https://www.theguardian.com/technology/technology.
[5] Heryck Barbosa and *et al.* 2023. PRIMA: an Automated Tool for Android Releases Homologation Review. In *Proc. of XIV CBSoft* (Campo Grande/MS). SBC, 1–4.
[6] Eduardo Blázquez and *et al.* 2021. Trouble over-the-air: An analysis of fota apps in the android ecosystem. In *Proc. of IEEE S&P*. 1606–1622.
[7] Antonio Brígido and *et al.* 2021. An Industrial Case Study on Applying Software Testing Automated in Global Software Development Environment. In *23rd ICGSET*. ACM, New York, USA, 1456–1459.
[8] Ana Carolina Chagas and *et al.* 2023. BSA Tool: An Experience Report of Software Automation to Perform Sanity Tests in a Global Software Development Environment. In *3rd ICICSE*. IEEE, Chongqing, China, 15–20.
[9] Antônio B. da Costa and *et al.* 2022. TSS Script: Automation Tool Applied in the Preparation of True Single SKU Testing Environment. In *2nd ICICSE*. IEEE.
[10] Saiyam Doshi. 2024. How to Obtain Google's GMS Certification for Latest Android Devices? https://www.einfochips.com/blog. Accessed: 2024-05-01.
[11] André José De Franca and *et al.* 2023. LinkDoc: An Automated Process in the Delivery of Doc in a Global Software Development Environment. In *5th WSSE*.
[12] Qinsheng Hou and *et al.* 2023. Can We Trust the Phone Vendors? Comprehensive Security Measurements on the Android Firmware Ecosystem. *IEEE TSE* (2023).
[13] Pedro Lancellotta and *et al.* 2022. An Industry Case Study: Methodology Application to the Reviewing Process on Android Releases Homologation. In *Anais Estendidos do XIII Congresso Brasileiro de Software: Teoria e Prática*. SBC, 13–16.
[14] Rayfran Rocha Lima and *et al.* 2021. Overcoming Knowledge-Sharing Barriers that Affect Software Quality: An Experience Report. In *Proc. XX SBQS '21*.
[15] Maria Meireles and *et al.* 2022. The Employment of Testing DOJO as a Collaborative Learning Methodology for Teaching Failure Analysis. In *Proc. CSTE*.
[16] Lucas Sousa and *et al.* 2023. BTS-Validator: identificando Aplicações Potencialmente Prejudiciais embarcadas no Android. In *Proc. ERRC/WRSeg*. SBC.
[17] Maddie Stone. 2019. Securing the System: A Deep Dive into Reversing Android Pre-Installed Apps. https://www.blackhat.com/us-19/briefings/schedule/.