v. 4, n. 22, 2024

Journal of Engineering Research

IMPROVING THE ANDROID RELEASE HOMOLOGATION PROCESS THROUGH AN AUTOMATED SYSTEM APPLIED TO SECURITY MAINTENANCE RELEASES

Heryck Michael Dos Santos Barbosa
Sidia Institute of Science and Technology - SIDIA, AM - Brazil

Pedro Ivo Pereira Lancellotta
Sidia Institute of Science and Technology - SIDIA, AM - Brazil

João Gabriel Castro dos Santos
Sidia Institute of Science and Technology - SIDIA, AM - Brazil

Abda Myrria De Albuquerque
Sidia Institute of Science and Technology - SIDIA, AM - Brazil

Janislley Oliveira de Sousa
Sidia Institute of Science and Technology - SIDIA, AM - Brazil

Alice Albuquerque Castro
Sidia Institute of Science and Technology - SIDIA, AM - Brazil

Abstract: Software testing is pivotal in ensuring the quality and reliability of technological products. This paper presents the development of PRIMA, an automated tool designed to enhance the homologation review process for Android releases. PRIMA systematically evaluates test artifacts produced by Google's trade-fed tools, utilizing a predefined set of rules to verify the accuracy and completeness of the results. To demonstrate the effectiveness of our tool, real-world Security Maintenance Release (SMR) tests were conducted within our company. The results underscore PRIMA's capacity to significantly streamline the Android release homologation process, reducing software release approval times by up to 54%. By automating these processes, PRIMA conserves time and resources for development teams and facilitates large-scale projects, such as Android, by efficiently managing multiple software artifacts in software release validation.

INTRODUCTION

Software testing is crucial for maintaining the integrity and quality of technological products, as it identifies potential issues early in the production process, saving time in later development phases [Zhao et al. 2021]. Google imposes stringent requirements on Original Equipment Manufacturers (OEMs) that wish to integrate the Android operating system into their mobile devices. Each device model released by OEMs must undergo a series of automated and manual tests to comply with Google's contractual requirements [AOSP 2023a]. Furthermore, OEMs must conduct additional tests tailored to their specific needs. After passing several compatibility tests, the results are submitted to Google for approval, and only the approved versions are released to the market.

In the realm of software release validation, automated tools are pivotal in ensuring quality assurance across the development life cycle [Pargaonkar 2023]. These tools are designed to automate the verification of code against predefined standards and to detect anomalies that could lead to future failures or security breaches [Thota et al. 2020]. For instance, continuous integration systems automatically build and test code with each change, facilitating early detection of integration issues. Additionally, automated regression testing tools rapidly assess the impact of code modifications on existing functionality, thereby ensuring that new developments do not disrupt the operational integrity of the software [Ali et al. 2020]. Performance testing tools simulate varying loads on the system to evaluate how changes might affect user experiences under different conditions. The strategic integration of these automated tools into the release process not only streamlines the validation phases but also significantly enhances the robustness and reliability of the software product [Enríquez et al. 2020]. Such automation is indispensable in today's fast-paced development to the annual release cycle of the Android system where the demand for rapid, yet reliable software deployment is ever-increasing [Mahmoudi and Nadi 2018]. By leveraging automated testing tools and processes, you can ensure rapid yet reliable software deployment while upholding stringent quality standards. This systematic approach not only aligns the final product with user expectations but also ensures Google regulatory compliance in the ever-evolving landscape of mobile development.

Maintaining quality control over Android release validation rules poses a significant challenge due to the large number of artifacts that must be analyzed at the end of the testing process [Lancellotta et al. 2022]. To enhance the review stage of these validations, we

proposed a methodology and developed an automated review tool named PRIMA (Primary Review in Mobile Android). Through the development of PRIMA, we aim to address the central research question: *Does the development and application of the proposed methodology improve the Android release homologation process?* Reviewing test artifacts before submitting them to Google, we anticipate a reduction in approval time and the elimination of additional fixes, thereby enhancing the validation process. We employed a quantitative approach to address our research question and test our hypothesis by analyzing our methodology in real-world Android release tests.

This paper is an extended version of a previously published article in an industry experience report [Barbosa et al. 2023]. It includes additional data comparing SMR tests conducted before and after the implementation of PRIMA during the second quarter of 2021 and the first quarters of 2023 and 2024. The goal is to assess the impact of PRIMA on the SMR testing process. This study provides valuable insights into how the proposed methodology and the PRIMA tool can enhance the Android release homologation process.

The following sections of this paper are organized as follows. Section 2 presents the background related to the Android life cycle product development. Section 3 outlines the methodology of the automated process developed through the creation of the PRIMA tool. Results will be presented in Section 4, highlighting the contributions of this work. Finally, Section 5 provides the conclusions regarding the automated system applied to security maintenance releases.

BACKGROUND

In this section, we will delve into the background of the Android release homologation process, focusing specifically on security maintenance releases. We will explore the Android product life cycle, the Google homologation process, and the importance of security maintenance releases in ensuring the safety and security of Android devices. By understanding these key components, we can better appreciate the need for an automated system to streamline and improve the Android release homologation process.

ANDROID PRODUCT LIFE CYCLE

To develop and release an Android-based device, Original Equipment Manufacturers (OEMs) must navigate various phases throughout the product life cycle. This intricate process necessitates a coordinated effort among multiple stakeholders, including Google, silicon manufacturers (SMs), device manufacturers (OEMs), and carriers. These parties must align their schedules to ensure seamless progression through each stage of development. Figure Figure 1 depicts the phases involved in launching an Android product, corresponding to each cycle of development [Proske et al. 2020].

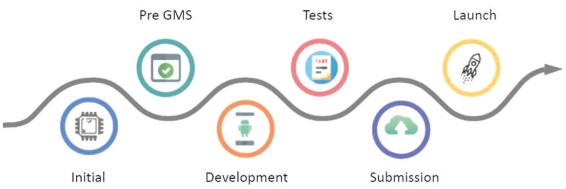

Figure 1 Android product cycles development

The OEMs must follow these phases in the Android product life cycle:

- **Initial cycle:** Product development begins with selecting the appropriate System on Chip (SoC), defining hardware specifications, and designing the device to integrate seamlessly with the Android

operating system [Yaghmour 2013].

- **Pre-GMS cycle:** This phase involves validating the product concept against the Compatibility Definition Document (CDD), which outlines the requirements that devices must meet to be compatible with the latest version of Android. The CDD acts as a central reference, linking to other resources such as SDK API documentation, to provide a framework for utilizing Android source code in creating compatible systems [AOSP 2024].

- **Development cycle:** To effectively run the latest Android OS, a device requires a compatible SoC supported by a Board Support Package (BSP). The BSP comprises vendor-specific implementations, key AOSP components, and additional framework elements for specific functionalities. During this phase, the operating system is developed based on the Android Open-Source Project (AOSP), which provides the necessary source code to build customized operating systems while adhering to CDD guidelines [AOSP 2023a, Yim et al. 2019]

- **Test cycle:** This phase is crucial for testing Android releases before their public deployment [Riccio et al. 2018]. Utilizing the Trade Federation (Tradefed) framework, the testing suite includes the Compatibility Test Suite (CTS), CTS Verifier for manual APIs, Vendor Test Suite (VTS) for reliability and system conformity, Google Test Suite (GTS) for Google Mobile Service (GMS) applications and Google contractual presets, Security Test Suite (STS) for CTS, and Closed Box Testing for OEM-specific applications. These tests form the automated component of our research focus [Lancellotta et al. 2022].

- **Submission cycle:** A mandatory process for obtaining Google's approval homologation before releasing Android software to the market. This phase ensures that all aspects adhere to Google's guidelines [Lancellotta et al. 2022].

- **Product launch cycle:** Following Google's approval and GMS certification, the product can be launched in the market through the Firmware Over-the-Air (FOTA) process [Blázquez et al. 2021].

GOOGLE HOMOLOGATION PROCESS

The Google homologation process refers to the rigorous testing and approval process followed by OEMs that Android devices must undergo in order to be certified by Google [Lancellotta et al. 2022]. This process ensures that devices meet Google's standards for performance, compatibility, and security. By successfully completing the homologation process, OEMs can ensure that their devices are ready for market release by FOTA and will provide a seamless user experience for consumers. Overall, the Google homologation process plays a crucial role in maintaining the quality and integrity of the Android ecosystem.

In our process, the Key Person (KP) is responsible for analyzing the release request and distributing the releases between the tester squads. When the squad receives the release, they start executing the tests. When the tests are completed, they can start the manual Review, at this point in the process, we are trying to improve and implement our tool named PRIMA. If there is any inconsistency in the test result, an analysis is opened, and the GMS Support team is responsible for verifying if these tests failed. If the test is

failed, the release is canceled. If the manual review reveals any wrong result, the tester can fix the result before submission. In case any wrong result is caught in manual review, the tester can proceed with the submission. The Google Build Approval (GBA) team is responsible for reviewing the results before making the final submission to Google. If everything is correct, the release is approved and sent to Google. However, if the GBA team identifies any inconsistencies, feedback is sent to the tester squad with the results that need to be fixed. The tester squad fixes the results and replies to the feedback. GBA reviews the results again to approve the release. If the results cannot be fixed, the release is rejected and another release needs to be created. Figure 2 illustrates the steps involved in the Google homologation process.

Reviewing the literature related to the Google homologation process for Android releases, there is not much information available on using automation tools to improve the release process. The paper [Balachandran 2013] discusses Review Bot, an automated source code review tool that not only generates automatic reviews but also recommends the most suitable reviewers based on criteria such as their familiarity with the project. Review Bot significantly reduces the workload in peer code reviews by identifying common errors, thereby allowing human reviewers to concentrate on more complex and unusual errors or behaviors. The study [Lancellotta et al. 2022] also aimed to reduce human effort, specifically in validating review points to enhance the Android release homologation process. Building on these concepts, this paper advances the work by developing a tool and integrating it into our company's daily operations. These tests are conducted to approve specific mobile software releases and their customization for the global market. Once the Google homologation process is completed, the approved release becomes available for download by end-users [Myrria de Albuquerque et al. 2023].

SECURITY MAINTENANCE RELEASE

Our company's release testing process involves three distinct scopes. The first is the SMR tests, which focus on validating security issues by applying patches from the Google Security Bulletin. The NE (Normal Exception) test scope follows and is responsible for validating the customization made by each carrier within OEMs. Finally, the Full Submission scope serves as a base release for the SMR and NE scopes and comprises a set of tests aimed at validating all components of the Android system release [Alure and Puri 2021].

To illustrate the complexity of SMR, NE scope, and Full Submissions have approximately 1200, 105.000, and 3.400.000 test cases respectively, because of that PRIMA started validating only SMR before applying the tool in NE and Full scopes, but even SMR has less number of test cases, it has a certain complexity level with on average, reviewing 6 XML files, 3 TXT files, 62 JSON files, 20 PNG files, security patches, and two other extension files. In our company's daily routine, each human test analyst typically concludes about five SMR tests per day, involving a review of nearly 500 files.

The Android Security Bulletin [AOSP 2023b] offers patches that address Common Vulnerabilities and Exposures (CVE) issues, fixing security vulnerabilities including buffer overflows, use-after-free errors, and invalid pointer references [Andrade et al. 2023, de Sousa et al. 2023]. To ensure the security and safety required by Android, researchers developed and tested methodologies and tools that employ state-of-the-art models for verifying large software systems [Farhang et al. 2020, Sousa 2023]. This process involves pre-processing input source-code files and

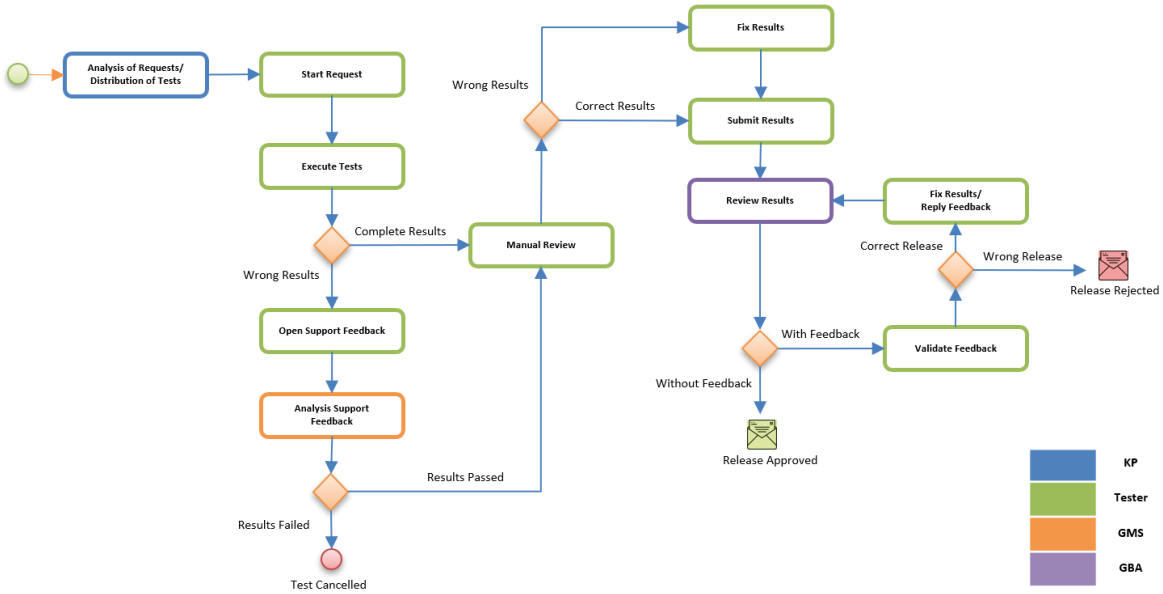

Figure 2 Google Homologation Process

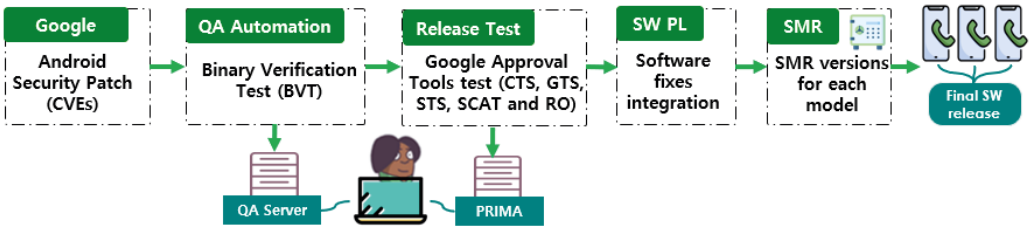

Figure 3 SMR process using PRIMA tool in Google Approval tests.

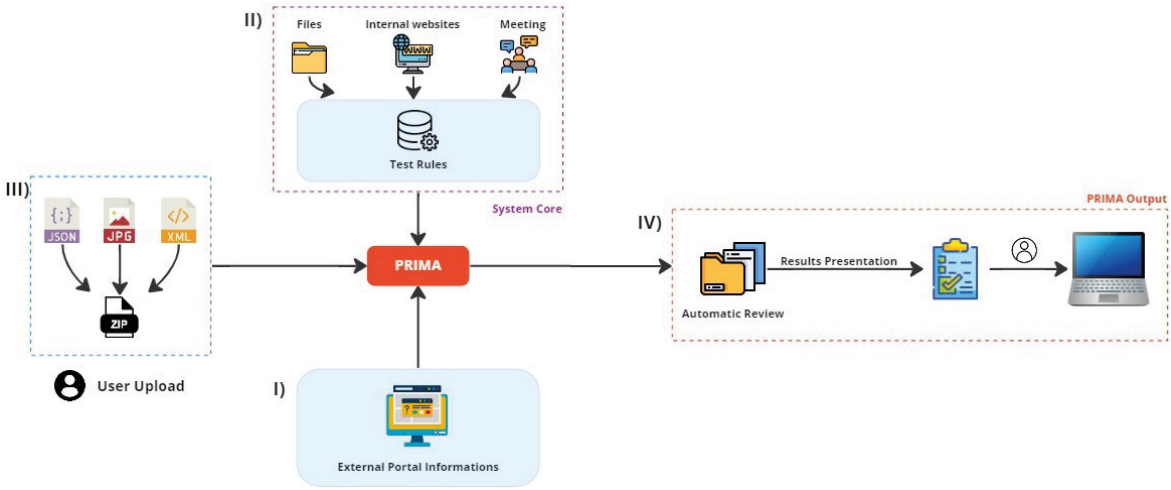

Figure 4PRIMA tool architecture and software artifacts

systematically guiding the model to analyze them efficiently and identify Android software vulnerabilities [Senanayake et al. 2023]. The release of these patches and maintenance updates for significant mobile device models is the responsibility of OEMs, who follow a monthly SMR process provided by Google, as shown in Figure 3.

PRIMA ARCHITECTURE

Our methodology employs a REST API architecture, which enhances the system's flexibility, scalability, and portability [John and Siddique 2021]. This architecture enables the broad application of our automated review tool, facilitating communication with other company tools that can leverage accurate result inputs for enhanced functionality.

PRIMA is an API developed using the proposed methodology, designed to review test artifacts for Android release homologation. Figure 4 illustrates the resulting architecture of our approach. The process begins when a software tester enters the ID of the release request from the management system (I), which contains important information about the release being tested. The software tester then loads the test artifacts (III) into PRIMA. These artifacts consist of output files generated by trade-fed tools, with various file types such as XML, JSON, TXT, and image extensions. Before proceeding (II), a set of rules for each trade-fed tool was established as a prerequisite to verify each validation point on a specific topic and the anticipated value. Finally, PRIMA compares all information extracted from the request (I) and the artifacts (III) with the implemented predefined rules (II). The tool then displays an output to the user with found and expected values (IV) for each incorrect value detected. With this information, the quality assurance tester can verify the proper files and take necessary actions to fix any issues.

RESULTS

In our company environment, various tests are conducted to homologate Android releases. Our experiment focused on the SMR test type, which involves a high volume of releases with frequent changes and updates to security patches and software fixes, as shown in Table 1. Despite the volume of releases, SMR has a smaller number of test artifacts compared to other types of tests.

Release Period	Submissions	TTS (Days)	TTA (Days)
2021 1st quarter	605	5,40	6,60
2021 2nd quarter	**723**	**2,85**	**4,18**
2021 3rd quarter	380	1,98	3,32
2021 4th quarter	542	2,69	3,27
2022 1st quarter	396	2,26	3,34
2022 2nd quarter	514	2,69	3,27
2022 3rd quarter	269	1,55	3,00
2022 4th quarter	628	2,11	3,38
2023 1st quarter	**792**	**1,40**	**2,00**
2023 2nd quarter	497	0,88	1,97
2023 3rd quarter	601	1,12	2,61
2023 4th quarter	540	1,70	3,50
2024 1st quarter	**819**	**1,25**	**1,89**

Table1Submissions TTS and TTA.

Table 1 and Figure 5 illustrate all SMR submissions made by our company between the years 2021 and the first quarter of 2024. We observed the time to submission (TTS) and time to approval (TTA), both are measured in days. TTS corresponds to the average time between release test creation and when quality assurance tester submits to Google, TTA corresponds to the time between test creation and Google approval letter.

In Figure 5 the left side is the number of days, the right side is the number of submissions, each blue bar is the number of submissions for each quarter of the year from 2021 to 2024 first quarter, the orange line and gray line are the average time to submit (TTS) and the average time to approval (TTA) respectively. It is a visual way to represent the data presented in Table 1.

PRIMA was implemented in 2023 and we noticed the TTS and TTA were lower than other periods without our tool. Comparing the

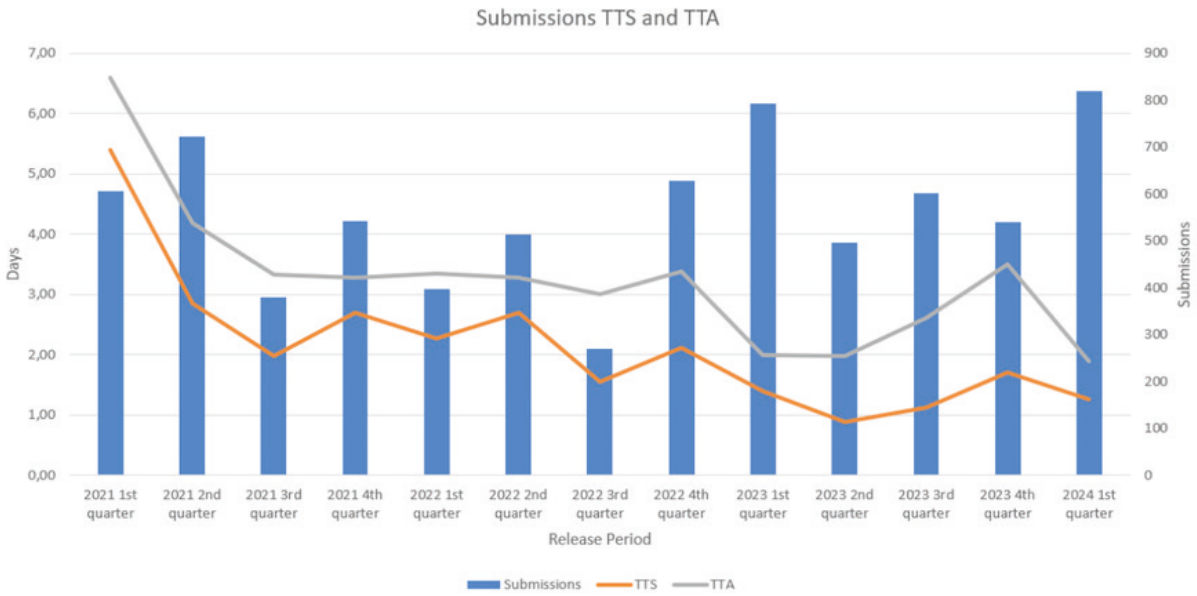

Figure 5. Submissions for each quarter related to TTS and TTA.

2023 1st quarter with the 2021 second quarter, which are periods with a similar number of submissions, we noticed approximately 50%-time reduction from both TTS and TTA. Comparing these two periods with the 2024 first quarter, even with more submissions this period has a time reduction from both TTS and TTA. Comparing only the 2021 second quarter with the 2024 first quarter we have a 56%-time reduction in TTS and a 54% gain in TTA.

In summary, our observations indicated an increase in the number of submissions and a reduction in the time required to approve a release. While acknowledging that factors such as team headcount and other test demands may influence these results, potential discrepancies, particularly in the second quarter of 2021, have been considered. However, analyzing the overall scenario, the implementation of PRIMA appears to enhance the efficiency of the Android release homologation process.

As we can observe using the PRIMA tool, automating the software release process homologation can save time and effort for development teams, speeding up the overall release process. Automation tools ensure consistent and accurate homologation, reducing errors and risks. Real-time feedback from automated tools helps teams address issues promptly, accelerating the release process. Scaling processes becomes easier with automation, benefiting larger projects such as Android and simultaneous software releases. By automating homologation, teams reduce manual work, cut costs, and improve overall efficiency.

CONCLUSION AND FUTURE WORKS

Our observation from the SMR type of real-world Android release tests demonstrated a significant improvement in the homologation process reducing until 54% time to release approval. The tool's effectiveness and seamless integration into daily workflows justify its expansion to Full Submission and Normal Exception release types, which have a larger scope and are more susceptible to manual errors, we propose to bring more rigorous experiments to test the effectiveness of PRIMA in those scopes to minimize external factors.

ACKNOWLEDGMENT

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

REFERENCES

Ali, S., Hafeez, Y., Hussain, S., and Yang, S. (2020). Enhanced regression testing technique for agile software development and continuous integration strategies. Software Quality Journal, 28:397–423.

Alure, S. and Puri, R. (2021). Firmware designing for android mobile. INTERNATIONAL JOURNAL, 5(12).

Andrade, E., Franca, H., Lima, W., and Barbosa, D. (2023). Sisyphus: um organizador de informações relacionadas a vulnerabilidades e correções para dispositivos android. In Anais da XX Escola Regional de Redes de Computadores, pages 109–114. SBC.

AOSP (2023a). Android open-source project. Available at https://source.android.com/ (accessed: 2023/04/13).

AOSP (2023b). Android security bulletin. Available at https://source.android.com/security/bulletin (accessed: 2023/04/13).

AOSP (2024). Android compatibility definition document. Available at https://source.android.com/docs/compatibility/cdd (accessed: 2024/04/19).

Balachandran, V. (2013). Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation. In 2013 35th International Conference on Software Engineering (ICSE), pages 931–940. IEEE.

Barbosa, H., Lancellotta, P., Santos, J., Albuquerque, A., and Sousa, J. (2023). Prima: an automated tool for android releases homologation review. In Anais Estendidos do XIV Congresso Brasileiro de Software: Teoria e Pŕ atica, pages 1–4, Porto Alegre, RS, Brasil. SBC.

Blázquez, E., Pastrana, S., Feal, Á., Gamba, J., Kotzias, P., Vallina-Rodriguez, N., and Tapiador, J. (2021). Trouble over-the-air: An analysis of fota apps in the android ecosystem. In 2021 IEEE Symposium on Security and Privacy (SP), pages 1606–1622. IEEE.

de Sousa, J. O., de Farias, B. C., da Silva, T. A., Cordeiro, L. C., et al. (2023). Finding software vulnerabilities in open-source c projects via bounded model checking. arXiv preprint arXiv:2311.05281.

Enríquez, J. G., Jiménez-Ramírez, A., Domínguez-Mayo, F. J., and García-García, J. A. (2020). Robotic process automation: a scientific and industrial systematic mapping study. IEEE Access, 8:39113–39129.

Farhang, S., Kirdan, M. B., Laszka, A., and Grossklags, J. (2020). An empirical study of android security bulletins in different vendors. In Proceedings of The Web Conference 2020, pages 3063–3069.

John, E. and Siddique, M. (2021). Efficient semantic web services development approaches using rest and json. In 2021 International Conference on Decision Aid Sciences and Application (DASA), pages 231–235.

Lancellotta, P. I. P., Barbosa, H. M. D. S., Santos, J. G. C., Sahdo, K. M. I., and De Sousa, J. O. (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, pages 13–16. SBC.

Mahmoudi, M. and Nadi, S. (2018). The android update problem: An empirical study. In Proceedings of the 15th International Conference on Mining Software Repositories, pages 220–230.

Myrria de Albuquerque, A., Barbosa, H., Lancellotta, P., Santos, J., and Sousa, J. (2023). Automating android rotation vector testing in Google's compatibility test suite using a robotic arm. In Proceedings of the 8th Brazilian Symposium on Systematic and Automated Software Testing, pages 54–63.

Pargaonkar, S. (2023). Synergizing requirements engineering and quality assurance: A comprehensive exploration in software quality engineering. International Journal of Science and Research (IJSR), 12(8):2003–2007.

Proske, M., Poppe, E., and Jaeger-Erben, M. (2020). "the smartphone evolution an analysis of the design evolution and environmental impact of smartphones ". Fraunhofer Institut für Zuverl ässigkeit und Mikrointegration.

Riccio, V., Amalfitano, D., and Fasolino, A. R. (2018). Is this the lifecycle we really want? An automated black-box testing approach for Android activities. In Companion Proceedings for the ISSTA/ECOOP 2018 Workshops, pages 68–77.

Senanayake, J., Kalutarage, H., Al-Kadri, M. O., Petrovski, A., and Piras, L. (2023). Android source code vulnerability detection: a systematic literature review. ACM Computing Surveys, 55(9):1–37.

Sousa, J. O. d. (2023). Lsverifier: a bmc approach to identify security vulnerabilities in c open-source software projects.

Thota, M. K., Shajin, F. H., Rajesh, P., et al. (2020). Survey on Software Defect Prediction techniques. International Journal of Applied Science and Engineering, 17(4):331–344.

Yaghmour, K. (2013). Embedded Android: Porting, Extending, and Customizing. " O'Reilly Media, Inc.".

Yim, K. S., Malchev, I., Hsieh, A., and Burke, D. (2019). Treble: Fast software updates by creating an equilibrium in an active software ecosystem of globally distributed stakeholders. ACM Transactions on Embedded Computing Systems (TECS), 18(5s):1–23.

Zhao, Y., Hu, Y., and Gong, J. (2021). Research on international standardization of software quality and software testing. In 2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall), pages 56–62.