Smart Test Execution of Android Compatibility Tests

Alice Castro, Heryck Barbosa, Abda Albuquerque, Roger Porty
{alice.castro,heryck.barbosa,abda.albuquerque,roger.porty}@sidia.com
Sidia Institute of Science and Technology - SIDIA
Manaus, AM, Brazil

ABSTRACT

With technology and internet expansion, the demand for software services has increased. As a result, these services have shifted from individual machines to being hosted on servers for remote access. This paper presents Smartgate, a web service that processes tasks from our request manager, accepting or rejecting them and starting Normal Exception (NE) scope tests. It provides early results to reduce the execution time of tests carried out by engineers and avoids human errors in the process, whether during system development or on the part of the test engineer. The data available in the task needs to be reviewed and validated for Smartgate to start correctly. Now, Smartgate has gained space in our software testing process after serving 53.16% of Normal Exception requests created and submitted from 2021 to 2023.

CCS CONCEPTS

• **Software and its engineering → Operational analysis**; **Software as a service orchestration system**.

KEYWORDS

Software Test, Test Automation, MoraySTF, Micro-services, Google Tests, Android Tests, Mobile Software

1 INTRODUCTION

The smartphone market has experienced exponential growth in recent years. In 2023, Samsung and Apple each produced a staggering 226.6 million and 234.6 million units, respectively [7]. With the constant emergence of new technologies, manufacturers regularly release and update new smartphone models. Consequently, software testing is crucial during system development to ensure the flawless performance of smartphones and to prevent the delivery of malfunctioning features. This testing plays a pivotal role in maintaining the high standard expected by consumers in the dynamic smartphone market [6].

When it comes to testing software for Android devices, it is essential to adhere to the requirements and guidelines set forth by Google, the owner of the system. These guidelines are detailed in the Compatibility Definition Document (CDD) [3]. The testing process encompasses a variety of approaches, including manual tests and automated tests. The Compatibility Test Suite (CTS) [4] is an automated test that works through scripts, and is one of the tests provided by Google. This test suite ensures that the software meets standards required for compatibility and performance on Android devices.

As technology has advanced and the internet has expanded, the demand for software services has increased. Consequently, these services have transitioned from being processed on individual users' machines to being hosted on servers, making them more easily accessible remotely. To address evolving demands, developers have employed new development methods and software architectures, such as the micro-services architecture. This modern approach emphasizes breaking down the system into small, independent services, thereby creating distributed applications that can communicate with each other through APIs (as noted in the [1]).

This paper presents our tool named Smartgate, a web service developed by the Google Approval team at the Sidia Institute of Science and Technology. Smartgate is a tool that handles tasks from our request manager, conducting acceptance or rejection processes and initiating NE scope tests. It comprises six distinct services: Kira, GA Web, Watcher, MoraySTF, DoligoRota, and Task Manager System. In essence, Smartgate validates test requests, triggers automated tests, and forwards the results to the analyst for further action. Notably, it has successfully processed 53.16% of NE requests submitted between 2021 and 2023, marking a significant advancement in our software testing process.

2 BACKGROUND

Our team at Sidia Institute encountered several challenges that needed to be resolved: the time required to set up the testing environment due to the multiple smartphone models, which sometimes required extra hours at night or on weekends; the requirement for human validation of test requests; the need for testers to initiate and execute automated tests; and the wasted human effort when tests were canceled. In addition to this daily challenge, we faced a larger one - how to conduct tests during the COVID-19 lockdown. At that time, we required a tool to automatically initiate tests without human intervention, as there were a limited number of people in the office. Therefore, since the end of 2020, Smartgate has been developed to meet our needs, whose purpose is to increase productivity by eliminating the need of engineers in the beginning of test process and, as outcome, providing results as clear as possible with the minimum number of failed test cases.

3 RELATED WORK

The article *Automatic Flash Tool for Mobile Devices* discusses a project where the authors designed an automated tool specifically for flashing Android OS images onto smartphones. To accomplish this, they implemented a system using a Raspberry Pi board and a hardware interface. This system is capable of running a service that can flash Android OS onto as many as 4 connected devices simultaneously.

The team achieved significant time savings using this automation tool, allowing them to redirect their efforts to other tasks. Additionally, they were able to schedule automated executions at the end of the day, resulting in improved efficiency [2].

3.1 Consideration

In contrast to the study by Correia et al. [2], Smartgate has the capability to interact with multiple smartphones on a daily basis. This system utilizes micro-services and does not rely on specific hardware, such as Raspberry Pi and JIG, to communicate with smartphones. Smartgate is designed to function within a laboratory environment (Figure 1), where it can access about 120 Linux computers and hubs to flash Android OS image about 2500 smartphones and test it. The only human involvement necessary is to connect models that have not yet been linked to the hubs.

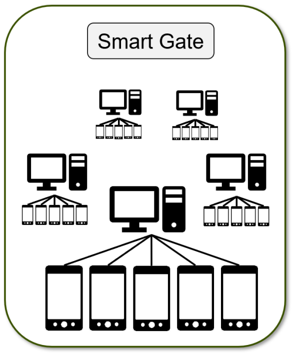

Figure 1: Smartgate Laboratory

Both projects share the same goal: to accelerate execution time, enabling testers to allocate their attention to other aspects of software testing.

4 SMARTGATE ARCHITECTURE

The proposed solution is related to the Android Compatibility Test Process for mobile devices. The goal is to ensure that all core functions of the system are working correctly and as expected, taking into account the Android version and the device's hardware specifications, in line with the predefined standards set by Google.

Creating a system that handles technical information related to the software of different models and contains a large amount of data requires modularization to better distribute activities. This is necessary due to the interactions between frameworks, which involve the main steps: Task Validation and Test Execution.

The testing process begins with a test requisition containing sensitive data about the software to be tested. It is crucial to carefully review this information. The micro-service (Kira) system will validate and either accept or reject the task. If rejected, the system will provide details on the issues with the request (Figure 2).

Once the task is accepted, the system will send information to the device laboratory. The laboratory manages all devices used for testing purposes, providing information on device availability and installing the required software for compatibility testing. If there are no available devices, the system must notify through a web service interface that additional devices need to be added. If devices are available, the software will be installed on the specific model, and the system should also provide progress updates. Once this step is completed, the system will commence test execution (micro-service DoligoRota) based on the test scope specified in the previously created task.

The web service (GA Web) interface should display all background processes, such as accepted tasks and their current status, and allow users to view sensitive information enabling them to monitor progress. It should also provide test plan configuration, including device allocation logic and the number of devices dedicated to each test scope, as well as the ability to view live test execution via transmission from a Linux agent, and to display error messages specifying the issue whenever they occur.

During the execution phase, the system consults the defined strategy on the web service to allocate the appropriate number of devices and perform tests using the latest tools available on the Linux agents, using the most current tools according to the type of test requested. Once the initial test results are generated, the system will automatically re-run them to reduce the number of failed and not executed tests, as the primary goal of compatibility validation is to pass all test cases. If some test cases does not pass, the system will provide the result anyway. The system must upload the results generated on Task Manager after the execution step.

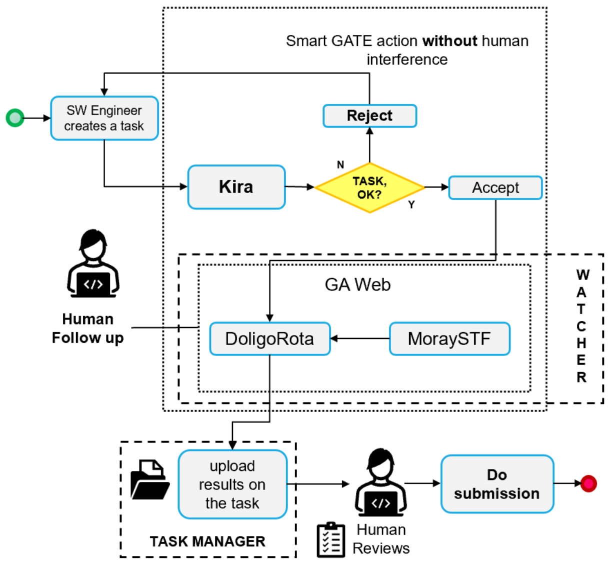

Figure 2: Smartgate Architecture

Smartgate consists of the following services:

- Kira: Service responsible for reviewing requests created on global management system, checking binary information, creating and updating manager task and registering task information into the web service to provide follow-up to testers.
- GA Web: It is a web service that shows status transitions of tasks as they pass by all the steps of acceptance, execution, retry, and submission. It also contains a test plan as a test view streamed from a Linux agent.
- Watcher: Service that provides communication between all system's modules. It is main purpose is to commute data as the process goes by.

- MoraySTF: Service that manages devices inside device laboratory. It is responsible to provide availability status and flashing procedure whenever a new task is added to the web service board [5].
- DoligoRota: Service responsible for running new test execution from Google tradefed (CTS, GTS and STS) and retrying generated results.
- Task Manager System: Global system that manages compatibility tasks. The proposed system gets initial data from it and concludes the process steps uploading and performing task transitions.

5 SMARTGATE TIMELINE

At the end of 2020 and in 2021, the Smartgate concept was developed to conduct testing during the COVID-19 pandemic. Initially, the Doligorota service carried out the first CTS and GTS execution of NE scope based on requests from the activities board using devices from a smaller version of the device laboratory. At that time, the validation and acceptance of the requests were done manually. The tests performed were then attached to the request so the engineer could proceed and complete the process.

In 2022, we proudly introduced new features. The integration with the global management system allowed the creation of Kira, which accepts or rejects the requests, but we started validation in another scope. And with the expansion of the device laboratory, dynamic device allocation has been implemented.

The following year, improvements and new implementations brought more agility and robustness to the project. In 2023, Kira began to validate and accept or reject NE requests.

And now, in 2024, Smartgate starts to execute STS in NE following the rule when it is mandatory to run.

6 RESULTS

The main goal of Smartgate was to manage pending requests, reduce the time it takes to start these requests, and free up testers to focus on other aspects of software testing, like manual tests on mobile devices. Figure 3 demonstrates that Smartgate has been successful in saving testers time over the years. Although still in the prototype stage, we conducted tests at the end of 2020 to validate the project concept. In 2021, with a small laboratory comprising about 150 mobile devices, we were able to process 390 requests.

In September 2022, our project seized an opportunity to upgrade it is the laboratory and acquired roughly 2500 mobile devices of various models. It successfully fulfilled 1101 requests. In 2023, Smartgate managed to handle a little over half of the 864 requests. The decrease in the number of requests was due to the introduction of new models with new requirements for testing. From 2021 to 2023, Smartgate served 53.16% of the requests created and submitted by our team. With these results, Smartgate has gained space in our software testing process.

In addition to the numbers, Smartgate achieved other significant gains from the results: Initial human need to prepare the devices and start the tests removed, no prejudice if tasks is cancelled, results provided in advance, and reduced working time on weekends.

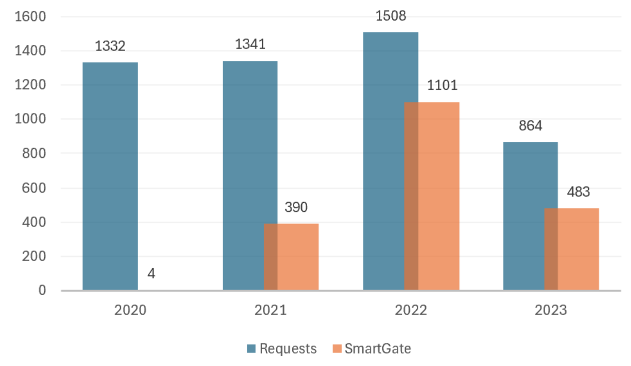

Figure 3: Number of requests executed by Smartgate over the years.

7 CONCLUSION

Over the years, we have focused our efforts and expertise on advanced activities in the realm of software quality. We are committed to implementing process improvements to ensure an exceptional experience for the end user. It is evident that Smartgate has effectively fulfilled it is purpose of enhancing productivity, directly influencing delivery times and result quality.

In future work, we intend to incorporate other test scopes into Smartgate, which also have details for review and require more devices to execute in less time. Just as we reach the end of the process for the NE scope, we will go step by step to reach the same end for the SMR and Full Submission scopes.

ACKNOWLEDGMENTS

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

[1] Nuha Alshuqayran, Nour Ali, and Roger Evans. 2016. A systematic mapping study in microservice architecture. In *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)*. IEEE, 44–51.
[2] Luiz Correia, Thales Silva, and Gabriel Villacrez. 2021. Automatic Flash Tool for Mobile Devices. In *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. IEEE, 1–5.
[3] Android Developer. 2024. *Compatibility Definition Document*. Retrieved Jun 06, 2024 from https://source.android.com/docs/compatibility/cdd
[4] Android Developer. 2024. *Compatibility Test Suite*. Retrieved Jun 06, 2024 from https://source.android.com/docs/compatibility/cts
[5] Klinsman M Gonçalves, Yasmine G Vaz, Eberth F Cruz, Rafael E Silva, Lineker Souza, Fábio M Azevedo, Eduardo D Sardinha, Paulo Fonseca, and Cícero AL Pahins. 2020. Using a tool-based approach to comply with smartphone user manual regulations in latin America countries. In *Proceedings of the 15th International Conference on Global Software Engineering*. 101–105.
[6] Mika V Mäntylä, Bram Adams, Foutse Khomh, Emelie Engström, and Kai Petersen. 2015. On rapid releases and software testing: a case study and a semi-systematic literature review. *Empirical Software Engineering* 20 (2015), 1384–1425.
[7] Fábio Matos. 2024. *Metrópoles: Apple bate Samsung e assume liderança na produção mundial de celulares*. Retrieved Jun 06, 2024 from https://www.metropoles.com/negocios/apple-bate-samsung-e-assume-lideranca-na-producao-mundial-de-celulares