Benefits and Challenges of a Physical Device Farm for Automated Software Testing: A Case Study of the STELA Tool Implementation

Lennon Chaves*
Sidia Institute of Technology
Manaus, Brazil
lennon.chaves@sidia.com

Flavia Oliveira*
Sidia Institute of Technology
Manaus, Brazil
flavia.oliveira@sidia.com

Leonardo Tiago
Sidia Institute of Technology
Manaus, Brazil
leonardo.albuquerque@sidia.com

Renata Castro
Sidia Institute of Technology
Manaus, Brazil
renata.castro@sidia.com

ABSTRACT

Software testing ensures seamless user experience that is achievable through manual or automated testing. Automated testing offers benefits, such as time reduction and parallel task execution. Physical device farms are used to conduct automated tests across different Android versions. The Software Test Execution Lab for Automation (STELA), designed as a Physical Device Farm, was recently implemented at a software institute by an automation team. This tool, housed in a lab with a capacity of 400 mobile devices, aided the testing team in executing automated tests. Interviews with 22 testing team members highlighted time optimization and a simplified interface as the primary benefits of STELA. Since its implementation, approximately 979 tests have been conducted, saving approximately 1915 minutes. This study examines the pros and cons of using a Physical Device Farm for software testing.

CCS CONCEPTS

• **Software and its engineering → Software verification and validation**; **Software development process management**; **Process validation**; **Software defect analysis**; **Software development methods**.

KEYWORDS

Software Testing, Physical Device Farm, Automated Testing, Experience Report

1 INTRODUCTION

In the field of software testing, it is evident that considerable effort has been invested in devising tests for mobile devices, including hardware and software, considering the multitude of mobile device manufacturers, operating systems, and available device types [1]. Specifically, Android operating systems present several unique aspects in application development, such as screen size, resolution, smartphone or tablet design, and diverse Android system versions [1]. Consequently, difficulties encountered in software testing have increased for various software platforms and mobile devices, making it imperative to preemptively address issues before end users discover them [1, 3].

In this context, a common strategy in the software industry is utilizing emulated Device Farms, which are cloud services emulating various mobile devices as if they were real [2]. This allows testing without physical devices using cloud services such as AWS Device Farm[1] and Google Firebase Test Lab[2]. Virtual mobile devices, essentially mobile devices emulated on servers using virtualization techniques, are frequently used [2]. However, a drawback is that these virtual devices may not accurately represent the behavior of physical devices, potentially leading to false-positive results [2].

Another approach is the concept of a Physical Device Farm [2], which uses the principle of using an infrastructure of real mobile devices to perform tests on different kinds of devices and custom software versions [1, 2]. The significant benefit of this strategy is the employment of mobile devices with the necessary configuration to perform tests in a specific scenario, collect log files from real devices, and reproduce and understand real behaviors in a mobile device that contrasts the simulation of scenarios through virtual devices [2].

In particular, the present study provides an exploration of the advantages and disadvantages of integrating a Physical Device Farm into the Institute of Research and Development, which comprises a testing team that conducts functional tests on mobile devices, specifically smartphones and tablets, using the Android operating system. To this end, the test automation team developed a software tool called the Software Test Execution Lab for Automation (STELA) which was implemented in a laboratory equipped with 20 servers and 400 mobile devices. Initial research was conducted using STELA through a survey administered to a sample of test team members, who represented a portion of the total testing team [4]. Although the findings were encouraging, this study identified areas for improvement and potential avenues for future research and development in STELA.

To achieve the goal of detailing the results of the developed work, Section 2 describes the investigation of the use of STELA. In Section 3, the results are discussed. Finally, in Section 4, the final considerations are detailed.

*Both authors contributed equally in this research.

[1] https://aws.amazon.com/device-farm/
[2] https://firebase.google.com/docs/test-lab

2 EXPLORATORY STUDY

2.1 Problem Context

To understand the context of projects validated by the test team, tests were conducted during the development phase if the Android operating system was for client projects, and the scope varied across different tests [3], such as sanity, interoperability, exploratory, and requirements tests from Latin American carriers. Each project represents a smartphone or tablet, and the Android version tested varies with the software release embedded in the device. The testing team handles all projects for clients in Latin American countries, encompassing a variety of devices that differ by type and operating system version. Clients also customize the operating system due to the advantages provided by the Android open-source project[3].

The testing team must meet significant daily demands to ensure that the test scope requirements are met. The test lifecycle comprises planning, specification, execution, and validation. In the planning stage, the team devises test cases based on the understood requirements by incorporating all the validation steps. These cases were executed and validated to ensure that they met the initial requirements. The scope of testing encompasses UI and functional testing, covering mobile application features and Android customization including app positioning, embedded versions, device settings, connections, and various screens.

SELBOT, a Device Farm pilot tool developed in 2020 for mobile device testing, has proven beneficial owing to its extensive test scope, numerous devices requiring validation, and a high demand for test execution on specific hardware and software versions. However, SELBOT's implementation faces maintenance difficulties, non-scalable infrastructure, incomplete test validation, and an inability to execute specific test suites, making it challenging to use and insufficient for the testing team's requirements.

2.2 STELA Device Farm

The Software Test Execution Lab for Automation (STELA) is an implementation that serves as a Physical Device Farm designed to ensure software quality through automated test execution. STELA addresses a series of problems by minimizing the possibility of human error, improving the efficiency of the test process, and reducing costs and execution time. The testing team implemented STELA in mid-August 2023. In particular, STELA was implemented using VueJS[4] and NodeJS[5] as front-end, while the back-end was developed using Python[6] and Flask[7]. These technologies were essential to ensure the scalability of STELA to attend a large number of mobile devices during the test execution.

In this way, STELA possesses an architecture based on three large modules:

(1) **Test Request Analysis**: consists in the validation of necessary information to start the test execution, in which it is determined if the test scope and the release are ready to be validated by the testing team;
(2) **Device Management**: combined with MoraySTF, which is a tool used to control access and use of the mobile devices that are connected to STELA. In this way, whenever a mobile device is connected to a STELA client station, MoraySTF manages access and uses it to perform tests.
(3) **Automated Execution**: performs the execution automatically according to the type of tests that are necessary to validate the software release. All automated test cases are executed, evidence of the execution is collected, and they are attached to the test management software (Jira[8]).

Regarding test execution, STELA initially prepares the device according to the software that will be validated and then starts the process of execution and validation of the tests. Soon after, the tests are finished, and a test report is generated along with evidence that will be attached to the test management system. Finally, STELA ends the test execution by updating the test status so that the tester knows that it has finished. Figure 1 illustrates how this process was performed.

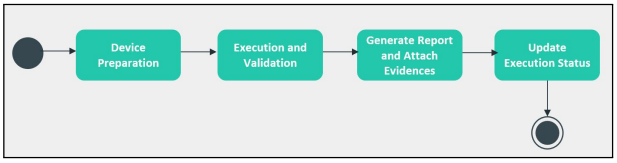

Figure 1: Test Execution Flow when using STELA

In this way, regarding the previous solution called SELBOT, STELA differs in its sample management, the flexibility of its use for test execution, and the monitoring of the tests' progression and scalability, as described in Table 1.

Table 1: How STELA differs from SELBOT

Feature	SELBOT	STELA
Analysis of test requests	Yes	Yes
Creation of test plans	Yes	Yes
Sample management	No	Yes
Flexible configuration	No	Yes
Automated test execution	Yes	Yes
Monitoring of test progress	No	Yes
Generation of test report	Yes	Yes
Scalability	No	Yes

It is important to highlight that this study does not consider the use of a Virtual Device Farm owing to the confidentiality issues surrounding the projects developed by the client; thus, it is not possible to perform tests in the cloud, which in turn makes STELA an adequate tool to perform tests in real devices developed by the client. In addition, due to many vendor's customization of Android system, the STELA was implemented to reproduce the particularities developed of the client's project.

[3]https://developer.android.com/
[4]https://vuejs.org/
[5]https://nodejs.org/
[6]https://www.python.org/
[7]https://flask.palletsprojects.com/
[8]https://www.atlassian.com/software/jira

2.3 Study Characterization

The goal of this study is to identify the main results of the use of STELA by a testing team that aims to validate different Android software releases. To support this study, a survey was developed to verify the viability of STELA use in a testing team with mobile projects from a global client. This research involved the participation of 22 testers of the test team and was designed with 10 questions, which involved the following topics: how is the tool used, what are its benefits, its disadvantages, the challenges faced, and the differences when compared to other approaches and suggestions for improvements. In addition, the number of validations performed by STELA's automated test suite was noted, that is, the number of tests performed on the Device Farm, with the goal of presenting its adherence to the team. Finally, the results were based on the lessons learned from the perceptions of the testing team.

3 FINDINGS AND LESSONS

The main paper contribution is on physical device farms, and for this reason, the lessons learned were performed regarding the usage of STELA by software testers. The interpretation and discussion of the results are summarized below, divided by topics:

Time Optimization: The interviewees highlighted that parallel execution improves efficiency. They valued the ability of the tool to manage test requests and validation processes. This tool expedited evidence collection, enabling the completion of more test requests. A significant finding was the comparison of the duration of manual versus automated tests using STELA to identify the potential advantages.

Simplified Interface: The team members highlighted that one of STELA's key advantages is its streamlined interface with integrated filters, providing users with greater flexibility. Users can decide whether to configure the device themselves or have STELA conduct the entire testing process, including validating test cases, simplifying its usage. The layout of the tool was reportedly easy to navigate for overseeing tests.

Dependence on the Automation: The participants in the surveys also mentioned the potential to become dependent on the tool, as when the system is offline, the entire process must be carried out manually, which takes significantly more time and may necessitate consulting tutorials if the tester does not recall how to perform the task or has never done so without the aid of automation.

Dependence on External Systems: According to the survey, the participants reported that STELA is reliant on external systems (Jira), and that when the system is not operational, the tool cannot be utilized. Furthermore, it was mentioned that due to this reliance on other systems, the tool needs updates that require it to be taken offline, and the test must be performed manually.

SELBOT versus STELA: The team employed SELBOT to manage and execute test requests, with 77.3% of the interviewees utilizing it; however, only 41.2% performed test executions. The remaining users employ SELBOT solely to validate and manage the test requests. Conversely, 81.8% of the 22 participants used STELA for test execution, whereas 18.2% used it only to validate the test requests. STELA offers many improvements over SELBOT, including advanced filters, customizable test sets, clearer error comments, and enhanced device preparation and test execution.

Test Execution: Since STELA's implementation on the testing team in August 2023, around 979 test executions have been conducted using a Physical Device Farm for test automation. This method has reduced test execution time by 1915 minutes, underscoring STELA's significant benefit in decreasing testing duration and enhancing the team's efficiency in managing client projects.

4 CONCLUSIONS

This study examined the implementation of a Physical Device Farm named STELA by a software testing team at the Institute of Research and Development. The team manages various projects and test scopes daily, utilizing STELA to execute tests across multiple devices. STELA was deployed in a lab with 20 servers and a capacity for 400 devices, automating device preparation, test execution, evidence collection, and test request management. Introduced in mid-August 2023, STELA's effectiveness was assessed through online surveys with 22 team members. Results indicate that STELA optimizes time by parallelizing tasks and offers a more user-friendly interface compared to the previous tool, SELBOT. Notably, even team members who did not use SELBOT are now using STELA. Since its implementation, STELA has conducted approximately 979 tests, reducing execution time by about 1915 minutes. This paper highlights STELA's achievements, including reduced testing time, with further claims to be explored in future research. Future work should expand automatic validation to include more test cases and automate test request handling.

ACKNOWLEDGMENTS

This paper is a result of the Research, Development & Innovation Project (ASTRO) performed at Sidia Institute of Science and Technology sponsored by Samsung Eletrônica da Amazônia Ltda., using resources under terms of Federal Law No. 8.387/1991, by having its disclosure and publicity in accordance with art. 39 of Decree No. 10.521/2020.

REFERENCES

[1] Haipeng Cai, Ziyi Zhang, Li Li, and Xiaoqin Fu. 2019. A large-scale study of application incompatibilities in android. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 216–227.

[2] Hao Lin, Jiaxing Qiu, Hongyi Wang, Zhenhua Li, Liangyi Gong, Di Gao, Yunhao Liu, Feng Qian, Zhao Zhang, Ping Yang, and Tianyin Xu. 2023. Virtual Device Farms for Mobile App Testing at Scale: A Pursuit for Fidelity, Efficiency, and Accessibility. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. Association for Computing Machinery, New York, NY, USA, Article 45, 17 pages. https://doi.org/10.1145/3570361.3613259

[3] Goutam Kumar Saha. 2008. Understanding software testing concepts. *Ubiquity* 2008, February, Article 2 (feb 2008), 1 pages. https://doi.org/10.1145/1361367.1348484

[4] Janice Singer, Susan E. Sim, and Timothy C. Lethbridge. 2008. *Software Engineering Data Collection for Field Studies*. Springer London, London, 9–34. https://doi.org/10.1007/978-1-84800-044-5_1