

AUTOMADB

**Business Case**

Joaquin Miranda Castro

192493

Business Case for AutomADB

[Introduction 3](#_Toc40919533)

[Scope 3](#_Toc40919534)

[Framework Mechanism: 4](#_Toc40919535)

[SDK 4](#_Toc40919536)

[GUI 4](#_Toc40919537)

[Data integrity 4](#_Toc40919538)

[Auto reporting system 4](#_Toc40919539)

[Approach 5](#_Toc40919540)

[UI: 5](#_Toc40919541)

[Scripts – Test Suite: 5](#_Toc40919542)

[Framework: 5](#_Toc40919543)

[Libraries: 5](#_Toc40919544)

[Android: 5](#_Toc40919545)

[Applications under Test: 5](#_Toc40919546)

[Architecture 6](#_Toc40919547)

[Level of effort 8](#_Toc40919548)

[Return on Investment 9](#_Toc40919549)

[Benefits: 10](#_Toc40919550)

# Introduction

Automated testing is a crucial step in the current software development industry. With automation testing we would not require human intervention to run our tests which verify the integrity of our software. Thus, with an initial investment for the development of this automation framework we can increase the speed of test execution, test coverage, and ultimately significantly increase the return on investment for this software development project.

Furthermore, in these difficult times caused by the COVID pandemic, remote work has never been more important. The OpenSTF support allows the framework to test remote devices from an Android Device Farm, so the developers can share devices to evaluate their apps while keeping themselves safe.

The following document demonstrates how and where the automated framework will assist the development team and the stakeholders to be able to test, evaluate and certificate their apps.

# Scope

Android is in current development of the phone, calculator, and settings applications. The phone application has the functionality to call any emergency, national, and international phone numbers that the user may want to call. The calculator application offers the end user the ability to calculate with greater ease basic mathematical operations which include sum, subtraction, division and multiplication of any series of numbers including positive, negative and fractional numbers. The settings application offers the end user the ability to switch on and off the Wi-Fi for convenience purposes.

Our teams’ goal is to provide these three applications with a QA certification which guarantees the proper functionality requirements for the end user without any bugs or critical errors which could put at risk the experience for an android user.

Our certification is reached when 95% of all the test cases inside our test suite are passed without any outstanding bugs or issues.

It is also the goal to be able to run on an STF device farm, so the testing can be done remotely and on a more variety of devices.

The certification is based on android version 9 and include the following brands:

* Huawei Y9
* Motorola Moto G6

At the end of the development stage of our in-home framework we guarantee that it will run on the previously mentioned android versions and phone brands, and if an STF farm contains one or more of the mentioned devices, with the proper configuration the framework will be able to test them remotely.

# Framework Mechanism:

SDK**:** The current automation framework in development consists three external libraries, ADB shell, UI Automator and phone numbers. We also developed libraries which include a logger library to store the test execution results, a phone control library, validations library, a calculator library and an STF library. These work in conjunction with the scripts where they are called to be executed.

GUI**:** A GUI or tester interface was not developed due to the project’s time restrictions and to reduce the overall project costs.

Data integrity**:** This framework validates the phone numbers that are being called, the operations that are calculated and the phone processes that are executed such as turning on or off the Wi-Fi mode.

Auto reporting system**:** After the execution and validation of the scripts the framework stores the results in a test log to validate the state of the respective test case, thus concluding if it passed or failed.

# Approach

The framework now has scripts which allows the automated testing of android based cellphones, using the ADB shell, UI Automator, and phone number libraries in conjunction with the libraries we have created.

UI: the user interface used is only the Terminal and executes the framework either in local or STF.

Scripts – Test Suite: Dialer suite tests the phones ability to place calls to either emergency, national or international numbers. Secondly, the Wi-Fi suite tests the settings of the phone by switching on and off the Wi-Fi mode based on their current status. Lastly, the Calculator suite tests the calculator app where we calculate a variety of operations including sum, subtraction, division and multiplication. The suites are built of various test cases which have a common functionality and allow our testing to be more robust.

Framework: The framework defines and executes the test suites, each one of them contains test cases that are going to be run. There is no limit on the number of test cases per suite and can be customized accordingly. The framework also retrieves all the available STF devices and run the suites on each one of them.

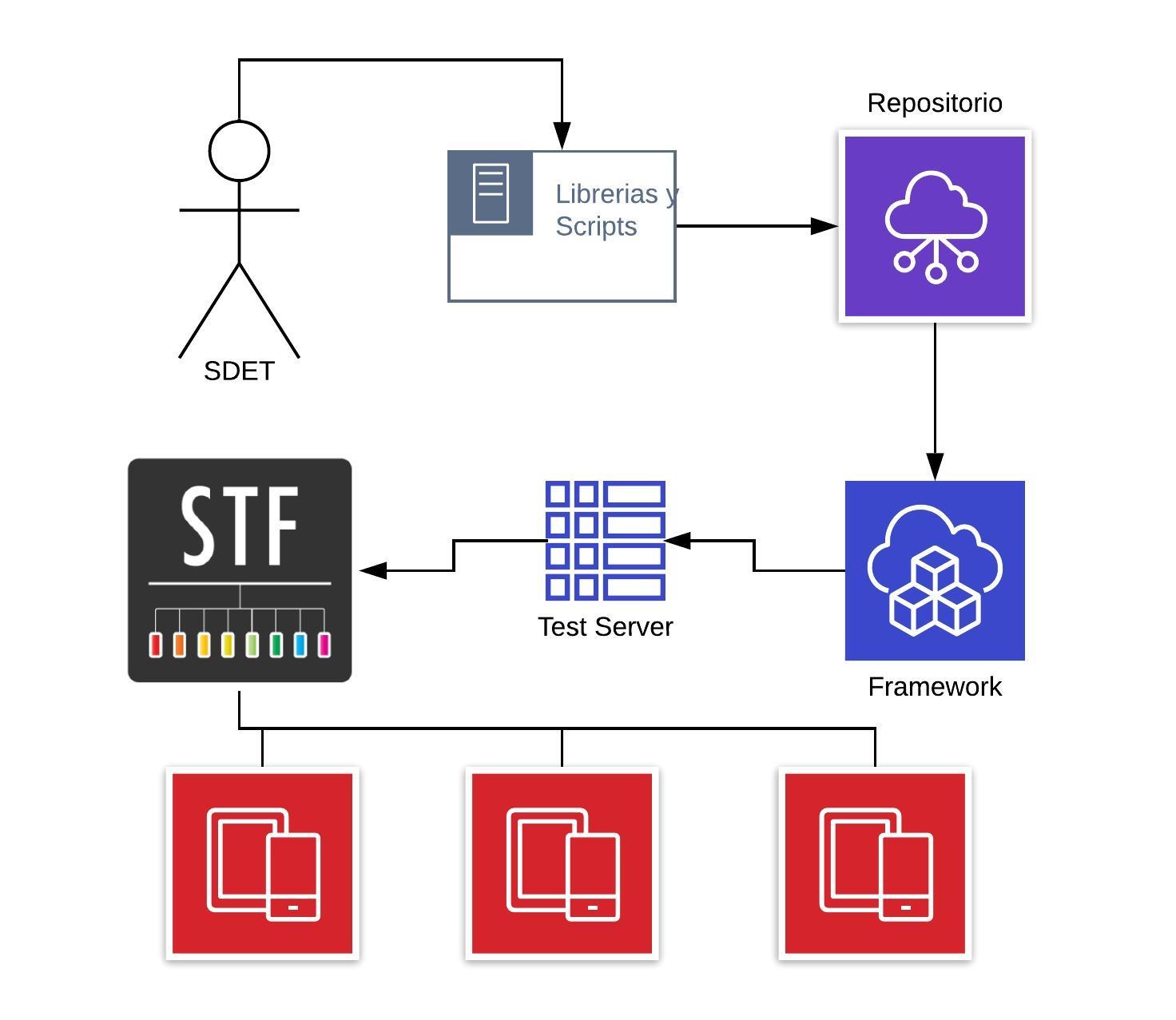
Libraries: The libraries are based on methods that are flexible and reusable which facilitate the development cycle. These libraries include but not limited to methods that control the phone, validate phone numbers, create a test logger, access the calculator app, etc.

Android: The android devices used are version 9 and can be local or remote on a STF Device Farm.

Applications under Test: Our framework finally calls the settings, phone and calculator apps to run the series of tests.

# Architecture

The framework which was developed by our team is based on scripts, libraries, and test suites which are stored on a git repository. This framework offers the possibility to run all of our 54 test cases in a faster and more precise manner. As previously stated, this framework uses external libraries such as ADB shell, UI Automator, and phone number. It is based on the python 2.7 language and is run on our test servers which is our local laptop and the remote devices connected to the STF instance.



* Library / scripts: group of methods are called a library. The test cases use these libraries and the test suites group these test cases. The scripts allow for the end to end execution of the test cases.
* Repository: Framework is stored on GitHub where version control is handled.
* Framework: the entire wrap of the libraries, scripts, methods, documentation which present a solution to the testing problem.
* Test server: Where our framework is run, requires git to retrieve all the changes made in the framework repository, python 2.7 to execute the scripts and ADB shell and UI Automator to adequately connect to the mobile device.
* STF: the STF instance acts as a device provider, which allows the adb from the Test server to connect remotely to the devices from the Farm.
* Mobile devices: The target of the framework where the tests are executed, and its results are evaluated. These devices are connected to the STF Device farm.

# Level of effort

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Task Activity | Saturday | Sunday | Monday | Tuesday | Wednesday | Thursday |
| STF Installation | 4 |  |  |  |  |  |
| STF Implementation |  | 7 |  |  |  |  |
| Test Case Execution |  |  | 2 |  |  |  |
| Code Maintenance |  |  | 2 |  |  |  |
| QA: Release notes/ bug reports |  |  | 1 |  |  |  |
| DOCS: business case |  |  |  | 2 |  |  |
| DOCS: core |  |  |  | 1 |  |  |
| DOCS: environment, setup and execution |  |  |  | 1 |  |  |
| QA: Test Plan |  |  |  |  | 2 |  |
| QA: Test Strategy |  |  |  |  | 2 |  |
| Demo Preparation |  |  |  |  |  | 2 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Hours | 26 | |  |  |  |  |
|  | Sprint 3 | |  |  |  |  |
|  | Version 2.0 | |  |  |  |  |
|  |  |  |  |  |  |  |
|  | Total hours: | 28 |  |  |  |  |
|  | Total USD: | $1,248.00 |  |  |  |  |

Note: The total USD price is based on the average hourly wage of a Software Development Engineer Test or SDET in the United States which varies greatly by state but has a national median of 48 USD per hour.

# Return on Investment

In the current state of this automation framework and with the current scope and requirements a test suite was built which consists of 54 test cases, validating each of the functionalities of the current applications in development by android.

If manual testing were to be implemented an it is estimated, it would take approximately 2 hours per device to cover all the test cases. Automated it would take 15 minutes per device.

Each execution of the tests is planned daily, and the estimated time to develop the framework is 50 hours. The following chart and graph present the ROI with two android devices.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteration | Manual testing | Automated testing | Man hours saved with automation |  |
| 1 | 4 | 50.5 | -46.5 |  |
| 2 | 8 | 51 | -43 |  |
| 3 | 12 | 51.5 | -39.5 |  |
| 4 | 16 | 52 | -36 |  |
| 5 | 20 | 52.5 | -32.5 |  |
| 6 | 24 | 53 | -29 |  |
| 7 | 28 | 53.5 | -25.5 |  |
| 8 | 32 | 54 | -22 |  |
| 9 | 36 | 54.5 | -18.5 |  |
| 10 | 40 | 55 | -15 |  |
| 11 | 44 | 55.5 | -11.5 |  |
| 12 | 48 | 56 | -8 |  |
| 13 | 52 | 56.5 | -4.5 |  |
| 14 | 56 | 57 | -1 |  |
| 15 | 60 | 57.5 | 2.5 | $120.00 |
| 16 | 64 | 58 | 6 | $288.00 |
| 17 | 68 | 58.5 | 9.5 | $456.00 |
| 18 | 72 | 59 | 13 | $624.00 |
| 19 | 76 | 59.5 | 16.5 | $792.00 |
| 20 | 80 | 60 | 20 | $960.00 |
|  |  |  |  | $3,240.00 |

# Benefits:

After observing the chart, we can thoroughly conclude that automation offers a significant Roi, thus reducing the overall software development project costs. It can also offer a higher overall test coverage due to the greater number of tests that can be executed on the application. This increases the quality of the application as a whole.

Automation also offers the opportunity of reusability of the framework and methods that where created and can be utilized through different approaches. It also offers the benefits of scalability, since the more devices we run the test on, the more early the ROI. Also, if we increase the number of test cases, the running time gap will be significantly increased between the manual and automated runs.

As the automated framework can be executed unattended at any desired time of the day, manual testers can also focus on more complex tests which cannot be automated, increasing the possibility of finding a bug or issue earlier in the development lifecycle and overall improving the productivity of the employees.